

UNIVERSITY OF KWAZULU-NATAL



DOCTORAL THESIS

Evolving Dynamic Fitness Measures for Genetic Programming

Author:
Anisa RAGALO

Supervisor:
Prof. Nelishia PILLAY

*A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy*

in the

School of Mathematics, Statistics and Computer Science

August 21, 2018

As the candidate's supervisor, I have approved this thesis for submission.

Signed:

Name: Professor Nelishia Pillay

Preface

The experimental work described in this thesis was carried out in the School of Computer Science, University of KwaZulu-Natal, Pietermaritzburg, from January 2012 to May 2018, under the supervision of Professor Nelishia Pillay.

The studies are original work by the author and have not been submitted in any form to any tertiary institution for any tertiary qualification such as degree or diploma. Where use has been made of the work of others it is duly acknowledged in the text.

Signed:

Professor Nelishia Pillay - Supervisor

Signed:

Anisa Ragalo - Candidate (Student number: 204505443)

Declaration of Authorship

I, Anisa RAGALO, declare that this thesis titled, 'Evolving Dynamic Fitness Measures for Genetic Programming' and the work presented in it are my own. I confirm that:

1. The research reported in this thesis, except where otherwise indicated, and is my original research.
2. This thesis has not been submitted for any degree or examination at any other university.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
 - (a) Their words have been re-written but the general information attributed to them has been referenced.
 - (b) Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the references section.

Signed:

Date:

Declaration of Publications

This research has produced the following publications.

1. Ragalo, A., and Pillay, N. "Evolving dynamic fitness measures for genetic programming". In: Expert Systems with Applications, (In Press, Accepted Manuscript). doi: 10.1016/j.eswa.2018.03.060.
2. Ragalo, A., and Pillay, N. "An investigation of dynamic fitness measures for genetic programming". In: Expert Systems with Applications, 92 (2018), pp. 52-72. doi: 10.1016/j.eswa.2017.08.022.
3. Ragalo, A., and Pillay, N. "Improved evolvability in genetic programming with polyandry". In: South African Computer Journal, 51 (2013), pp. 22-43. doi: 10.18489/sacj.v51i0.170.
4. Ragalo, A., and Pillay, N. "A hyper-heuristic approach towards mitigating premature convergence caused by the objective fitness function in GP". In: *Proceedings of the 14th International Conference on Intelligent Systems Design and Applications - ISDA 2014 (Okinawa, Japan)*. Ed. by Ali F. et. al. IEEE, 2014, pp. 68-75. doi: 10.1109/ISDA.2014.7066272.
5. Ragalo, A., and Pillay, N. "A building block conservation and extension mechanism for improved performance in polynomial symbolic regression tree-based genetic programming". In: *Proceedings of the Fourth World Congress on Nature and Biologically Inspired Computing - NaBIC 2012 (Mexico City, Mexico)*. Ed. by A. Gelbukh et. al. IEEE, 2012, pp. 123-129. doi: 10.1109/NaBIC.2012.6402250.

Signed:

Professor Nelishia Pillay - Supervisor

Signed:

Anisa Ragalo - Candidate (Student number: 204505443)

“Fitness is the driving force of Darwinian natural selection and, likewise, of both conventional genetic algorithms and genetic programming.”

John R. Koza

Abstract

Evolving Dynamic Fitness Measures for Genetic Programming

by Anisa RAGALO

This research proposes dynamic fitness measure genetic programming (DFMGP). DFMGP modifies the conventional genetic programming (GP) approach: rather than applying a single fitness measure individually throughout GP, a different fitness measure (or combination of fitness measures) is applied on each GP generation. A detailed review of the fitness measures used in GP is presented. The review demonstrates that different fitness measures were introduced to overcome different shortcomings, e.g. escaping local optima, reducing bloat, thereby improving on the performance of the GP algorithm. A subsequent analysis of the fitness measures shows that there is no universal “best” fitness measure; rather, different fitness measures are appropriate for different problems. The literature also anticipates that applying different fitness measures at different points of the GP problem solving process should be more effective than applying a single fitness measure throughout the algorithm. Hence the case for DFMGP.

Selecting the fitness measures to apply on each GP generation is in itself a combinatorial optimization problem: the study investigates two approaches to serve this purpose, namely, a genetic algorithm and genetic programming. The genetic algorithm (GA) derives a sequence of fitness measures to be applied, while GP produces an arithmetic function combining the fitness measures. The performance of DFMGP applying the evolved fitness measure sequences and DFMGP applying the evolved fitness measure combinations is compared to the conventional GP approach on a number of benchmark and complex, real-world problems.

DFMGP is found to be more effective than standard GP. The study also reveals that both the sequences and arithmetic combinations of the fitness measures are effective when applied to problem instances different from those used to derive them. Hence, the sequences and arithmetic combinations are reusable, whereby simpler problems are used for derivation, and DFMGP applying the derived fitness measures is then used to solve more complex problems. Therefore the time necessary for the derivations is reduced. An analysis of the evolved sequences and arithmetic combinations of the fitness measures shows that fitness measures applied in the preliminary DFMGP generations support exploration while those applied in later DFMGP generations support exploitation. GP search is a constant balance between exploration and exploitation, with the former being more suited to the preliminary generations, and the latter, later generations. DFMGP’s performance advantage over standard GP is therefore justified by the premise that the fitness measure used on each generation supports the more suitable search in the on-going phase of GP. DFMGP applying the fitness measure combinations derived by GP is also found to perform better than DFMGP applying the fitness measure sequences derived by the GA. The former approach facilitates combining explorative and exploitative fitness measures on some of the DFMGP generations, whereby rather than simply switching between exploration and exploitation, the fitness measure can drive the two processes to occur simultaneously when required. Hence it follows that GP searching the space of fitness measure combinations is the preferred approach to generating dynamic fitness measures for DFMGP.

Overall, the study reveals the effectiveness of DFMGP when applied to benchmark and real-world problems. Future work will look at a priori detecting the properties of complex problems, such that simpler problems with similar properties can be used to derive better dynamic fitness measures for DFMGP.

Acknowledgements

The financial assistance of the National Research Foundation, South Africa (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

I would like to thank the Centre for High Performance Computing, South Africa, for access to their resources and assistance. I would also like to thank my supervisor, Professor Nelishia Pillay, for her guidance and expertise in the fields of genetic algorithms and genetic programming.

Finally I would like to thank my dear husband Andrew and my daughter Azariah for their patience and support, as well as my dad and mum for always inspiring me to reach for my dreams.

Contents

Preface	ii
Declaration of Authorship	iii
Declaration of Publications	iv
Abstract	vi
Acknowledgements	vii
1 Introduction	1
1.1 Purpose of the study	1
1.2 Research perspective	2
1.3 Scope	3
1.4 Objectives	3
1.5 Contributions	3
1.6 Thesis layout	4
2 An Introduction to Genetic Algorithms and Genetic Programming	6
2.1 Introduction	6
2.2 Genetic algorithms	6
2.2.1 The genetic algorithm	7
2.2.2 Representation scheme	8
2.2.3 Initial population creation	9
2.2.4 Fitness evaluation and selection	9
2.2.5 Genetic operators	10
Crossover	10
Mutation	11
2.2.6 A critical analysis of genetic algorithms	12
2.3 Genetic programming	13
2.3.1 The genetic programming algorithm	13
2.3.2 Representation scheme	14
2.3.3 Initial population creation	15
Full	16
Grow	16
Ramped half-and-half	17
2.3.4 Fitness evaluation and selection	17
2.3.5 Genetic operators	18
Crossover	18
Mutation	20
2.3.6 Reproduction	21

2.3.7	A critical analysis of genetic programming	21
2.4	Summary	22
3	A Survey of Fitness Measures in GP	23
3.1	Introduction	23
3.2	Objective fitness	24
	Motivation	24
	Implementation	24
	Advantages	25
	Disadvantages	25
	Discussion	27
3.3	Divide-and-conquer fitness	28
3.3.1	Layered learning	28
	Motivation	28
	Implementation	29
	Variants	29
	Advantages	30
	Disadvantages	30
	Discussion	31
3.3.2	Behavioral programming	32
	Motivation	32
	Implementation	32
	Variants	33
	Advantages	34
	Disadvantages	35
	Discussion	35
3.4	Fitness sharing	35
	Motivation	36
	Implementation	36
	Variants	36
	Advantages	38
	Disadvantages	38
	Discussion	39
3.5	Dynamic fitness	39
3.5.1	Dynamic subset measures	40
	Motivation	40
	Implementation	40
	Variants	40
	Advantages	42
	Disadvantages	42
	Discussion	43
3.5.2	Stepwise adaptation of weights	43
	Motivation	43
	Implementation	44
	Variants	44
	Advantages	44
	Disadvantages	45
	Discussion	45

3.6	Subjective fitness	45
3.6.1	Competitive fitness	46
	Motivation	46
	Implementation	46
	Variants	47
	Advantages	48
	Disadvantages	48
	Discussion	49
3.6.2	Cooperative fitness	50
	Motivation	50
	Implementation	50
	Variants	51
	Advantages	51
	Disadvantages	51
	Discussion	52
3.7	Novelty search	52
	Motivation	53
	Implementation	53
	Variants	54
	Advantages	54
	Disadvantages	55
	Discussion	55
3.8	Summary	56
4	A Comparison of Fitness Measures in GP	57
4.1	Introduction	57
4.2	Experimental methodology	57
4.2.1	Fitness measures	58
4.2.2	Criteria for comparison	61
	Solution quality	61
	Generalization	62
	Population diversity	63
	Structural complexity	65
	Time taken	65
4.2.3	Benchmark suite	65
	Symbolic regression benchmarks	65
	Supervised classification benchmarks	67
	Boolean function synthesis benchmarks	69
	Path-finding benchmarks	70
4.2.4	Experiment set-up	72
4.2.5	Technical specifications	72
4.3	Results and discussion	72
4.3.1	Solution quality	73
	Symbolic regression benchmarks	73
	Supervised classification benchmarks	80
	Boolean function synthesis benchmarks	87
	Path-finding benchmarks	94
	Summary of the solution quality results	99

4.3.2	Generalization	100
	Symbolic regression benchmarks	101
	Supervised classification benchmarks	106
	Boolean function synthesis benchmarks	112
	Path-finding benchmarks	118
	Summary of the generalization results	122
4.3.3	Population diversity	123
	Symbolic regression benchmarks	123
	Supervised classification benchmarks	131
	Boolean function synthesis benchmarks	139
	Path-finding benchmarks	147
	Summary of the diversity results	153
4.3.4	Structural complexity	154
	Symbolic regression benchmarks	154
	Supervised classification benchmarks	159
	Boolean function synthesis benchmarks	165
	Path-finding benchmarks	171
	Summary of the structural complexity results	175
4.3.5	Time taken	176
	Symbolic regression benchmarks	176
	Supervised classification benchmarks	176
	Boolean function synthesis benchmarks	177
	Path-finding benchmarks	177
	Summary of the time taken results	177
4.4	Summary	178
5	Methodology	180
5.1	Introduction	180
5.2	Critical analysis of related literature	180
5.2.1	Justification of the GA approach for evolving DFMs for DFMGP	181
5.2.2	Justification of the GP approach for evolving DFMs for DFMGP	182
5.3	Research methodology	184
5.4	Objectives	184
5.4.1	Objectives one and two	184
	Overview of the objectives	184
	Measurements for analysis of the objectives	184
5.4.2	Objectives three, four and five	185
	Overview of the objectives	185
	Measurements for analysis of the objectives	186
	Hypothesis testing	188
5.4.3	Objective six	189
	Overview of the objective	189
	Measurements for analysis of the objective	189
5.5	Benchmark suite	189
5.5.1	Benchmark problems	189
	Problems used to test the effectiveness of DFMGP	190
	Problem classes used to test the reusability of the DFMs evolved by the GA and GP approaches	190

	Real-world problems used to test the reusability of the DFMs evolved by the GA and GP approaches	190
5.5.2	Function and terminal sets	191
5.5.3	Fitness cases	192
5.6	Technical specifications	193
5.7	Summary	193
6	The DFMGP Algorithm	194
6.1	Introduction	194
6.2	The DFMGP algorithm	194
6.3	Representation scheme	194
6.4	Initial population creation	195
6.5	Fitness evaluation	195
6.6	Selection method	195
6.7	Genetic operators	195
6.8	Parameter tuning	196
6.9	Summary	196
7	A GA Approach for Deriving DFMs for DFMGP	197
7.1	Introduction	197
7.2	Genetic algorithm for dynamic fitness measure GP	197
7.3	Representation	197
7.4	Initial population creation	199
7.5	Fitness evaluation	199
7.6	Selection method	200
7.7	Genetic operators	200
7.8	Parameter tuning	201
7.9	Summary	201
8	A GP Approach for Deriving DFMs for DFMGP	202
8.1	Introduction	202
8.2	Genetic programming for dynamic fitness measure GP	202
8.3	Representation	203
8.3.1	Terminal Set	203
8.3.2	Function Set	204
8.4	Initial population creation	205
8.5	Fitness evaluation	206
8.6	Selection method	206
8.7	Genetic operators	206
8.8	Parameter tuning	207
8.9	Summary	207
9	Results and Discussion	208
9.1	Introduction	208
9.2	Results of the GA approach and $DFMGP_{GA}$	208
9.2.1	Testing the effectiveness of $DFMGP_{GA}$	208
9.2.2	Testing the reusability of the derived DFMs within problem classes	210
9.2.3	Testing the reusability of the derived DFMs on real world problems	217
9.2.4	Analysis of the derived DFMs	219

9.3	Results of the GP approach and DFMGP_{GP}	220
9.3.1	Testing the effectiveness of DFMGP_{GP}	221
9.3.2	Testing the reusability of the derived DFMs within problem classes	222
9.3.3	Testing the reusability of the derived DFMs on real world problems	229
9.3.4	Analysis of the derived DFMs	230
9.4	Comparison of $\text{GA}/\text{DFMGP}_{\text{GA}}$ and $\text{GP}/\text{DFMGP}_{\text{GP}}$	234
9.4.1	Comparing the effectiveness of DFMGP_{GP} with DFMGP_{GA}	234
9.4.2	Comparing the reusability of the derived DFMs within problem classes	236
9.4.3	Comparing the reusability of the derived DFMs on real world problems	243
9.4.4	Comparing the derived DFMs	244
9.5	Summary	247
10	Conclusions and Future Work	248
10.1	Introduction	248
10.2	Objectives and Conclusions	248
10.3	Contributions	250
10.4	Future Work	252
10.4.1	Coevolving the parameters used by the fitness measures in DFMGP	252
10.4.2	Detecting problem properties	252
10.4.3	Testing other evolutionary algorithm approaches for evolving DFMs	252
10.4.4	Testing DFMs on other evolutionary algorithms	253
10.5	Summary	253
	Bibliography	254

List of Figures

2.1	GA fixed-length binary representation: Each bit (or group of bits) at a fixed locus represents the value of a variable being evolved.	8
2.2	GA fixed-length direct value representation: Each character or number at a fixed locus represents the value of a variable being evolved.	8
2.3	GA one-point crossover	10
2.4	GA two-point crossover	10
2.5	GA uniform crossover	11
2.6	GA one-point mutation of a bit string	11
2.7	GA one-point mutation of a character string	11
2.8	GA one-point mutation of a numerical string	11
2.9	Example GP parse tree	14
2.10	GP crossover	18
2.11	GP uniform crossover	19
2.12	GP subtree mutation	20
2.13	GP point mutation	20
4.1	Symbolic regression benchmarks: Mean final BS-OF scores - training set	73
4.2	Symbolic regression benchmarks: Mean final BS-OF scores - test set	74

4.3	Symbolic regression benchmarks: Mean generational BS-OF scores	75
4.4	Supervised classification benchmarks: Mean final BS-OF scores - training set	81
4.5	Supervised classification benchmarks: Mean final BS-OF scores - test set	81
4.6	Supervised classification benchmarks: Mean generational BS-OF scores	82
4.7	Boolean function synthesis benchmarks: Mean final BS-OF scores - training set	88
4.8	Boolean function synthesis benchmarks: Mean final BS-OF scores - test set	88
4.9	Boolean function synthesis benchmarks: Mean generational BS-OF scores	89
4.10	Path-finding benchmarks: Mean final BS-OF scores - training set	95
4.11	Path-finding benchmarks: Mean final BS-OF scores - test set	95
4.12	Path-finding benchmarks: Mean generational BS-OF scores	96
4.13	Symbolic regression benchmarks: Mean final BS GI	101
4.14	Symbolic regression benchmarks: Mean generational BS GI	102
4.15	Supervised classification benchmarks: Mean final BS GI	107
4.16	Supervised classification benchmarks: Mean generational BS GI	108
4.17	Boolean function synthesis benchmarks: Mean final BS GI	113
4.18	Boolean function synthesis benchmarks: Mean generational BS GI	114
4.19	Path-finding benchmarks: Mean final BS GI	119
4.20	Path-finding benchmarks: Mean generational BS GI	120
4.21	Symbolic regression benchmarks: Mean final population semantic diversity	124
4.22	Symbolic regression benchmarks: Mean final population entropy	124
4.23	Symbolic regression benchmarks: Generational population semantic diversity	125
4.24	Symbolic regression benchmarks: Generational population entropy	126
4.25	Supervised classification benchmarks: Mean final population semantic diversity	132
4.26	Supervised classification benchmarks: Mean final population entropy	132
4.27	Supervised classification benchmarks: Generational population semantic diversity	133
4.28	Supervised classification benchmarks: Generational population entropy	134
4.29	Boolean function synthesis benchmarks: Mean final population semantic diversity	140
4.30	Boolean function synthesis benchmarks: Mean final population entropy	140
4.31	Boolean function synthesis benchmarks: Generational population semantic diversity	141
4.32	Boolean function synthesis benchmarks: Generational population entropy	142
4.33	Path-finding benchmarks: Mean final population semantic diversity	148
4.34	Path-finding benchmarks: Mean final population entropy	148
4.35	Path-finding benchmarks: Generational population semantic diversity	149
4.36	Path-finding benchmarks: Generational population entropy	150
4.37	Symbolic regression benchmarks: Mean final BS size	154
4.38	Symbolic regression benchmarks: Mean generational population size	155
4.39	Supervised classification benchmarks: Mean final BS size	160
4.40	Supervised classification benchmarks: Mean generational population size	161
4.41	Boolean function synthesis benchmarks: Mean final BS size	166
4.42	Boolean function synthesis benchmarks: Mean generational population size	167
4.43	Path-finding benchmarks: Mean size of best solution	172
4.44	Path-finding benchmarks: Mean generational population size	173
6.1	Schematic of DFMGP. The symbols $F_i, F_{ii}, F_{iii} \dots F_N$ represent different fitness measures.	194
7.1	Overall GA approach	198
8.1	Overall approach	203
8.2	Example subtree rooted at arithmetic operator of type I	205

8.3	Example subtree rooted at arithmetic operator of type F	205
8.4	Example subtrees rooted at $IFGGT$ and $IFGLT$	206

List of Tables

3.1	Schematic of a program trace in BP	32
4.1	Fitness measures	58
4.1	Fitness measures (contd.)	59
4.1	Fitness measures (contd.)	60
4.2	SMAC tuned parameters for the fitness measures	61
4.3	Symbolic regression benchmarks, adapted from [232]	66
4.4	Supervised classification benchmarks, adapted from [232]	68
4.5	Boolean function synthesis benchmarks	69
4.6	Path-finding benchmarks	70
4.7	Symbolic regression benchmarks: Mean final BS-OF scores - training set	74
4.8	Symbolic regression benchmarks: Mean final BS-OF scores - test set	74
4.9	Sextic: Final BS-OF scores - training set	76
4.10	Sextic: Final BS-OF scores - test set	76
4.11	Sextic: Statistical tests on final BS-OF scores - training set	76
4.12	Sextic: Statistical tests on final BS-OF scores - test set	76
4.13	Nguyen: Final BS-OF scores - training set	77
4.14	Nguyen: Final BS-OF scores - test set	77
4.15	Nguyen: Statistical tests on final BS-OF scores - training set	77
4.16	Nguyen: Statistical tests on final BS-OF scores - test set	77
4.17	Pagie: Final BS-OF score - training set	77
4.18	Pagie: Final BS-OF score - test set	77
4.19	Pagie: Statistical tests on final BS-OF scores - training set	78
4.20	Pagie: Statistical tests on final BS-OF scores - test set	78
4.21	Keijzer: Final BS-OF score - training set	78
4.22	Keijzer: Final BS-OF score - test set	78
4.23	Keijzer: Statistical tests on final BS-OF scores - training set	78
4.24	Keijzer: Statistical tests on final BS-OF scores - test set	78
4.25	Vlad: Final BS-OF scores - training set	79
4.26	Vlad: Final BS-OF scores - test set	79
4.27	Vlad: Statistical tests on final BS-OF scores - training set	79
4.28	Vlad: Statistical tests on final BS-OF scores - test set	79
4.29	Dow: Final BS-OF scores - training set	79
4.30	Dow: Final BS-OF scores - test set	79
4.31	Dow: Statistical test on final BS-OF scores - training set	80
4.32	Dow: Statistical tests on final BS-OF scores - test set	80
4.33	Supervised classification benchmarks: Mean final BS-OF scores - training set	81
4.34	Supervised classification benchmarks: Mean final BS-OF scores - test set	81

4.35	Iris: Final BS-OF scores - training set	83
4.36	Iris: Final BS-OF scores - test set	83
4.37	Iris: Statistical tests on final BS-OF scores - training set	83
4.38	Iris: Statistical tests on final BS-OF scores - test set	83
4.39	Credit: Final BS-OF scores - training set	84
4.40	Credit: Final BS-OF scores - test set	84
4.41	Credit: Statistical tests on final BS-OF scores - training set	84
4.42	Credit: Statistical tests on final BS-OF scores - test set	84
4.43	Wine: Final BS-OF score - training set	84
4.44	Wine: Final BS-OF score - test set	84
4.45	Wine: Statistical tests on final BS-OF scores - training set	85
4.46	Wine: Statistical tests on final BS-OF scores - test set	85
4.47	Segment: Final BS-OF score - training set	85
4.48	Segment: Final BS-OF score - test set	85
4.49	Segment: Statistical tests on final BS-OF scores - training set	85
4.50	Segment: Statistical tests on final BS-OF scores - test set	85
4.51	Vowel: Final BS-OF scores - training set	86
4.52	Vowel: Final BS-OF scores - test set	86
4.53	Vowel: Statistical tests on final BS-OF scores - training set	86
4.54	Vowel: Statistical tests on final BS-OF scores - test set	86
4.55	Opt: Final BS-OF scores - training set	86
4.56	Opt: Final BS-OF scores - test set	86
4.57	Opt: Statistical tests on final BS-OF scores - training set	87
4.58	Opt: Statistical tests on final BS-OF scores - test set	87
4.59	Boolean function synthesis benchmarks: Mean final BS-OF scores - training set	88
4.60	Boolean function synthesis benchmarks: Mean final BS-OF scores - test set	88
4.61	Even-5 parity: Final BS-OF scores - training set	90
4.62	Even-5 parity: Final BS-OF scores - test set	90
4.63	Even-5 parity: Statistical tests on final BS-OF scores - training set	90
4.64	Even-5 parity: Statistical tests on final BS-OF scores - test set	90
4.65	Even-7 parity: Final BS-OF scores - training set	91
4.66	Even-7 parity: Final BS-OF scores - test set	91
4.67	Even-7 parity: Statistical tests on final BS-OF scores - training set	91
4.68	Even-7 parity: Statistical tests on final BS-OF scores - test set	91
4.69	Even-9 parity: Final BS-OF scores - training set	91
4.70	Even-9 parity: Final BS-OF scores - test set	91
4.71	Even-9 parity: Statistical tests on final BS-OF scores - training set	92
4.72	Even-9 parity: Statistical tests on final BS-OF scores - test set	92
4.73	11-multiplexer: Final BS-OF scores - training set	92
4.74	11-multiplexer: Final BS-OF scores - test set	92
4.75	11-multiplexer: Statistical tests on final BS-OF scores - training set	92
4.76	11-multiplexer: Statistical tests on final BS-OF scores - test set	92
4.77	3-bit multiplier: Final BS-OF scores - training set	93
4.78	3-bit multiplier: Final BS-OF scores - test set	93
4.79	3-bit multiplier: Statistical tests on final BS-OF scores - training set	93
4.80	3-bit multiplier: Statistical tests on final BS-OF scores - test set	93
4.81	4-bit multiplier: Final BS-OF scores - training set	93
4.82	4-bit multiplier: Final BS-OF scores - test set	93

4.83	4-bit multiplier: Statistical tests on final BS-OF scores - training set	94
4.84	4-bit multiplier: Statistical tests on final BS-OF scores - test set	94
4.85	Path-finding benchmarks: Mean final BS-OF scores - training set	94
4.86	Path-finding benchmarks: Mean final BS-OF scores - test set	94
4.87	Artificial Ant: Final BS-OF scores - training set	97
4.88	Artificial Ant: Final BS-OF scores - test set	97
4.89	Artificial Ant: Statistical tests on final BS-OF scores - training set	97
4.90	Artificial Ant: Statistical tests on final BS-OF scores - test set	97
4.91	Tartarus: Final BS-OF scores - training set	98
4.92	Tartarus: Final BS-OF scores - test set	98
4.93	Tartarus: Statistical tests on final BS-OF scores - training set	98
4.94	Tartarus: Statistical tests on final BS-OF scores - test set	98
4.95	Deceptive Tartarus: Final BS-OF scores - training set	98
4.96	Deceptive Tartarus: Final BS-OF scores - test set	98
4.97	Deceptive Tartarus: Statistical tests on final BS-OF scores - training set	99
4.98	Deceptive Tartarus: Statistical tests on final BS-OF scores - test set	99
4.99	Symbolic regression benchmarks: Mean final BS GI	101
4.100	Sextic: Final BS GI	103
4.101	Sextic: Statistical tests on final BS GI	103
4.102	Nguyen: Final BS GI	104
4.103	Nguyen: Statistical tests on final BS GI	104
4.104	Pagie: Final BS GI	104
4.105	Pagie: Statistical tests on final BS GI	105
4.106	Keijzer: Final BS GI	105
4.107	Keijzer: Statistical tests on final BS GI	105
4.108	Vlad: Final BS GI	105
4.109	Vlad: Statistical tests on final BS GI	106
4.110	Dow: Final BS GI	106
4.111	Dow: Statistical tests on final BS GI	106
4.112	Supervised classification benchmarks: Mean final BS GI	107
4.113	Iris: Final BS GI	109
4.114	Iris: Statistical tests on final BS GI	109
4.115	Credit: Final BS GI	109
4.116	Credit: Statistical tests on final BS GI	110
4.117	Wine: Final BS GI	110
4.118	Wine: Statistical tests on final BS GI	110
4.119	Segment: Final BS GI	111
4.120	Segment: Statistical tests on final BS GI	111
4.121	Vowel: Final BS GI	111
4.122	Vowel: Statistical tests on final BS GI	111
4.123	Opt: Final BS GI	112
4.124	Opt: Statistical tests on final BS GI	112
4.125	Boolean function synthesis benchmarks: Mean final BS GI	113
4.126	Even-5 parity: Final BS GI	115
4.127	Even-5 parity: Statistical tests on final BS GI	115
4.128	Even-7 parity: Final BS GI	115
4.129	Even-7 parity: Statistical tests on final BS GI	116
4.130	Even-9 parity: Final BS GI	116

4.131	Even-9 parity: Statistical tests on final BS GI	116
4.132	11-multiplexer: Final BS GI	117
4.133	11-multiplexer: Statistical tests on final BS GI	117
4.134	3-bit multiplier: Final BS GI	117
4.135	3-bit multiplier: Statistical tests on final BS GI	117
4.136	4-bit multiplier: Final BS GI	118
4.137	4-bit multiplier: Statistical tests on final BS GI	118
4.138	Path-finding benchmarks: Mean final BS GI	119
4.139	Artificial Ant: Mean final BS GI	121
4.140	Artificial Ant: Statistical tests on mean final BS GI	121
4.141	Tartarus: Mean final BS GI	121
4.142	Tartarus: Statistical tests on mean final BS GI	122
4.143	Deceptive Tartarus: Mean BS GI	122
4.144	Deceptive Tartarus: Statistical tests on mean BS GI	122
4.145	Symbolic regression benchmarks: Mean final population semantic diversity	124
4.146	Symbolic regression benchmarks: Mean final population entropy	124
4.147	Sextic: Final population semantic diversity	127
4.148	Sextic: Final population entropy	127
4.149	Sextic: Statistical tests on final population semantic diversity	127
4.150	Sextic: Statistical tests on final population entropy	127
4.151	Nguyen: Final population semantic diversity	128
4.152	Nguyen: Final population entropy	128
4.153	Nguyen: Statistical tests on final population semantic diversity	128
4.154	Nguyen: Statistical tests on final population entropy	128
4.155	Pagie: Final population semantic diversity	128
4.156	Pagie: Final population entropy	128
4.157	Pagie: Statistical tests on final population semantic diversity	129
4.158	Pagie: Statistical tests on final population entropy	129
4.159	Keijzer: Final population semantic diversity	129
4.160	Keijzer: Final population entropy	129
4.161	Keijzer: Statistical tests on final population semantic diversity	129
4.162	Keijzer: Statistical tests on final population entropy	129
4.163	Vlad: Final population semantic diversity	130
4.164	Vlad: Final population entropy	130
4.165	Vlad: Statistical tests on final population semantic diversity	130
4.166	Vlad: Statistical tests on final population entropy	130
4.167	Dow: Final population semantic diversity	130
4.168	Dow: Final population entropy	130
4.169	Dow: Statistical tests on final population semantic diversity	131
4.170	Dow: Statistical tests on final population entropy	131
4.171	Supervised classification benchmarks: Mean final population semantic diversity	131
4.172	Supervised classification benchmarks: Mean final population entropy	131
4.173	Iris: Final population semantic diversity	135
4.174	Iris: Final population entropy	135
4.175	Iris: Statistical tests on final population semantic diversity	135
4.176	Iris: Statistical tests on final population entropy	135
4.177	Credit: Final population semantic diversity	136
4.178	Credit: Final population entropy	136

4.179	Credit: Statistical tests on final population semantic diversity	136
4.180	Credit: Statistical tests on final population entropy	136
4.181	Wine: Final population semantic diversity	136
4.182	Wine: Final population entropy	136
4.183	Wine: Statistical tests on final population semantic diversity	136
4.184	Wine: Statistical tests on final population entropy	136
4.185	Segment: Final population semantic diversity	137
4.186	Segment: Final population entropy	137
4.187	Segment: Statistical tests on final population semantic diversity	137
4.188	Segment: Statistical tests on final population entropy	137
4.189	Vowel: Final population semantic diversity	138
4.190	Vowel: Final population entropy	138
4.191	Vowel: Statistical tests on final population semantic diversity	138
4.192	Vowel: Statistical tests on final population entropy	138
4.193	Opt: Final population semantic diversity	138
4.194	Opt: Final population entropy	138
4.195	Opt: Statistical tests on final population semantic diversity	139
4.196	Opt: Statistical tests on final population entropy	139
4.197	Boolean function synthesis benchmarks: Mean final population semantic diversity	139
4.198	Boolean function synthesis benchmarks: Mean final population entropy	139
4.199	Even-5 parity: Final population semantic diversity	143
4.200	Even-5 parity: Final population entropy	143
4.201	Even-5 parity: Statistical tests on final population semantic diversity	143
4.202	Even-5 parity: Statistical tests on final population entropy	143
4.203	Even-7 parity: Final population semantic diversity	144
4.204	Even-7 parity: Final population entropy	144
4.205	Even-7 parity: Statistical tests on final population semantic diversity	144
4.206	Even-7 parity: Statistical tests on final population entropy	144
4.207	Even-9 parity: Final population semantic diversity	144
4.208	Even-9 parity: Final population entropy	144
4.209	Even-9 parity: Statistical tests on final population semantic diversity	145
4.210	Even-9 parity: Statistical tests on final population entropy	145
4.211	11-multiplexer: Final population semantic diversity	145
4.212	11-multiplexer: Final population entropy	145
4.213	11-multiplexer: Statistical tests on final population semantic diversity	145
4.214	11-multiplexer: Statistical tests on final population entropy	145
4.215	3-bit multiplier: Final population semantic diversity	146
4.216	3-bit multiplier: Final population entropy	146
4.217	3-bit multiplier: Statistical tests on final population semantic diversity	146
4.218	3-bit multiplier: Statistical tests on final population entropy	146
4.219	4-bit multiplier: Final population semantic diversity	146
4.220	4-bit multiplier: Final population entropy	146
4.221	4-bit multiplier: Statistical tests on final population semantic diversity	147
4.222	4-bit multiplier: Statistical tests on final population entropy	147
4.223	Path-finding benchmarks: Mean final population semantic diversity	147
4.224	Path-finding benchmarks: Mean final population entropy	147
4.225	Artificial ant: Final population semantic diversity	151
4.226	Artificial ant: Final population entropy	151

4.227	Artificial ant: Statistical tests on final population semantic diversity	151
4.228	Artificial ant: Statistical tests on final population entropy	151
4.229	Tartarus: Final population semantic diversity	151
4.230	Tartarus: Final population entropy	151
4.231	Tartarus: Statistical tests on final population semantic diversity	152
4.232	Tartarus: Statistical tests on final population entropy	152
4.233	Deceptive tartarus: Final population semantic diversity	152
4.234	Deceptive tartarus: Final population entropy	152
4.235	Deceptive tartarus: Statistical tests on final population semantic diversity	152
4.236	Deceptive tartarus: Statistical tests on final population entropy	152
4.237	Symbolic regression benchmarks: Mean final BS size	154
4.238	Sextic: Final BS size	156
4.239	Sextic: Statistical tests on final BS size	156
4.240	Nguyen: Final BS size	156
4.241	Nguyen: Statistical tests on final BS size	157
4.242	Pagie: Final BS size	157
4.243	Pagie: Statistical tests on final BS size	157
4.244	Keijzer: Final BS size	157
4.245	Keijzer: Statistical tests on final BS size	158
4.246	Vlad: Final BS size	158
4.247	Vlad: Statistical tests on final BS size	158
4.248	Dow: Final BS size	159
4.249	Dow: Statistical tests on final BS size	159
4.250	Supervised classification benchmarks: Mean final BS size	160
4.251	Iris: Final BS size	162
4.252	Iris: Statistical tests on final BS size	162
4.253	Credit: Final BS size	162
4.254	Credit: Statistical tests on final BS size	163
4.255	Wine: Final BS size	163
4.256	Wine: Statistical tests on final BS size	163
4.257	Segment: Final BS size	163
4.258	Segment: Statistical tests on final BS size	164
4.259	Vowel: Final BS size	164
4.260	Vowel: Statistical tests on final BS size	164
4.261	Opt: Final BS size	165
4.262	Opt: Statistical tests on final BS size	165
4.263	Boolean function synthesis benchmarks: Mean final BS size	166
4.264	Even-5 parity: Final BS size	168
4.265	Even-5 parity: Statistical tests on final BS size	168
4.266	Even-7 parity: Final BS size	168
4.267	Even-7 parity: Statistical tests on final BS size	169
4.268	Even-9 parity: Final BS size	169
4.269	Even-9 parity: Statistical tests on final BS size	169
4.270	11-multiplexer: Final BS size	169
4.271	11-multiplexer: Statistical tests on final BS size	170
4.272	3-bit multiplier: Final BS size	170
4.273	3-bit multiplier: Statistical tests on final BS size	170
4.274	4-bit multiplier: Final BS size	171

4.275	4-bit multiplier: Statistical tests on final BS size	171
4.276	Path-finding benchmarks: Mean size of best solution	172
4.277	Artificial Ant: Final BS size	174
4.278	Artificial Ant: Statistical tests on final BS size	174
4.279	Tartarus: Final BS size	174
4.280	Tartarus: Statistical tests on final BS size	175
4.281	Deceptive tartarus: Final BS size	175
4.282	Deceptive tartarus: Statistical tests on final BS size	175
4.283	Symbolic regression benchmarks: Mean execution time (seconds)	176
4.284	Supervised classification benchmarks: Mean execution time (seconds)	177
4.285	Boolean function synthesis benchmarks: Mean execution time (seconds)	177
4.286	Path-finding benchmarks: Mean execution time (seconds)	177
5.1	Benchmark problems tackled	190
5.2	Benchmark training/test problems tackled	191
5.3	Real-world problems tackled	192
8.1	Terminal set	204
8.2	Function set	206
9.1	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Quality Scores	209
9.2	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Statistical Tests	209
9.3	$GA + DFMGP_{GA}$ vs. Standard GP: Execution Time (milliseconds)	210
9.4	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Sextic Class - Test Set Quality Scores	211
9.5	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Sextic Class - Statistical Tests	211
9.6	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Keijzer-6 Class - Test Set Quality Scores	212
9.7	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Keijzer-6 Class - Statistical Tests	212
9.8	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Even-N Parity Class - Test Set Quality Scores	213
9.9	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Even-N Parity Class - Statistical Tests	213
9.10	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: N-bit Multiplier Class - Test Set Quality scores	214
9.11	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: N-bit Multiplier Class - Statistical Tests	214
9.12	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Tartarus Class - Test Set Quality Scores	215
9.13	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Tartarus Class - Statistical Tests	215
9.14	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Deceptive Tartarus Class - Test Set Quality Scores	216
9.15	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Deceptive Tartarus Class - Statistical Tests	217
9.16	$DFMGP$ vs Standard GP: Real-World Problem Quality Scores	217
9.17	$DFMGP_{P1}$ vs. $DFMGP_{P2}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests	218
9.18	$DFMGP_{P2}$ vs. $DFMGP_{P1}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests	218
9.19	$DFMGP_{UN}$ vs. $DFMGP_{P1}/DFMGP_{P2}$ /Standard GP: Real-World Problems - Statistical Tests	218
9.20	Evolved Fitness Measure Sequences - Problem Classes	219
9.21	Evolved Fitness Measure Sequences - Combined Training Sets	220
9.22	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Quality Scores	221
9.23	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Statistical Tests	222
9.24	$GP + DFMGP_{GP}$ vs. Standard GP: Execution Time (milliseconds)	222
9.25	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Sextic Class - Test Set Quality Scores	223
9.26	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Sextic Class - Statistical Tests	223
9.27	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Keijzer-6 Class - Test Set Quality Scores	224

9.28	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Keijzer-6 Class - Statistical Tests	225
9.29	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Even-N Parity Class - Test Set Quality Scores	225
9.30	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Even-N Parity Class - Statistical Tests	225
9.31	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: N-bit Multiplier Class - Test Set Quality scores	226
9.32	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: N-bit Multiplier Class - Statistical Tests	226
9.33	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Tartarus Class - Test Set Quality Scores	227
9.34	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Tartarus Class - Statistical Tests	227
9.35	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Deceptive Tartarus Class - Test Set Quality Scores	228
9.36	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{RD2}</i> /Standard GP: Deceptive Tartarus Class - Statistical Tests	228
9.37	<i>DFMGP_{GP}</i> vs. Standard GP: Real World Quality Scores	229
9.38	<i>DFMGP_{P1}</i> vs. <i>DFMGP_{P2}</i> / <i>DFMGP_{UN}</i> /Standard GP: Real-World Problems - Statistical Tests	230
9.39	<i>DFMGP_{P2}</i> vs. <i>DFMGP_{P1}</i> / <i>DFMGP_{UN}</i> /Standard GP: Real-World Problems - Statistical Tests	230
9.40	<i>DFMGP_{UN}</i> vs. <i>DFMGP_{P1}</i> / <i>DFMGP_{P2}</i> /Standard GP: Real-World Problems - Statistical Tests	230
9.41	Evolved DFMs - Problem Classes	231
9.41	Evolved DFMs - Problem Classes (contd.)	232
9.42	Evolved DFMs - Combined training sets	234
9.43	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Quality Scores	235
9.44	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Statistical Tests	235
9.45	GP + <i>DFMGP_{GP}</i> vs. GA + <i>DFMGP_{GA}</i> : Execution Time (milliseconds)	236
9.46	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Sextic Class - Test Set Quality Scores	237
9.47	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Sextic Class - Statistical Tests	237
9.48	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Keijzer-6 Class - Test Set Quality Scores	238
9.49	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Keijzer Class - Statistical Tests	238
9.50	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Even-N Parity Class - Test Set Quality Scores	239
9.51	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Even-N Parity Class - Statistical Tests	239
9.52	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : N-bit Multiplier Class - Test Set Quality scores	240
9.53	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : N-bit Multiplier Class - Statistical Tests	240
9.54	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Tartarus Class - Test Set Quality Scores	241
9.55	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Tartarus Class - Statistical Tests	241
9.56	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Deceptive Tartarus Class - Test Set Quality Scores	242
9.57	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Deceptive Tartarus Class - Statistical Tests	242
9.58	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> : Real World Quality Scores	243
9.59	<i>DFMGP_{GP}</i> vs. <i>DFMGP_{GA}</i> /Standard GP: Real-World Problems - Statistical Tests	243
9.60	Evolved DFMs - Problem Classes	244
9.60	Evolved DFMs - Problem Classes (contd.)	245
9.60	Evolved DFMs - Problem Classes (contd.)	246
9.61	Evolved DFMs - Combined Training Sets	246

For dad, mum, Andrew and Azariah, my cheerleaders.

Chapter 1

Introduction

1.1 Purpose of the study

Genetic programming (GP) is an evolutionary algorithm that explores a program space [1]. Evolutionary algorithms (EAs) employ mechanisms inspired by biological evolution (that is, selection, recombination, mutation and reproduction) to solve complex problems [1–3]. A wide variety of problems from different fields can be recast as requiring a computer program that converts inputs into desired outputs [1]. GP breeds a population of computer programs, and employs the metaphor of evolution to search the space of possible programs for an optimal program that solves (or approximately solves) the problem at hand [1].

In GP, a fitness measure is used to distinguish “fit” candidate solutions [1]. The fit candidate solutions are preferentially selected to parent subsequent generations in the iterative search for an optimal solution. As a consequence, GP search becomes increasingly concentrated in regions of the search space that the fitness measure has deemed to be promising. By utilizing a fitness measure to apply selective pressure, GP mandates that mostly only fit candidate solutions are considered for future evolution [1]. Therefore, the fitness measure steers the direction of search.

The convention in GP is to apply a single fitness measure individually throughout the course of the algorithm. At the inception of GP [1], objective fitness (OF) was used as a fitness measure; a detailed description of OF follows in chapter 3. As the field developed, various fitness measures emerged in an attempt to overcome the shortcomings of GP, such as premature convergence and the growth of redundant code [4–13]. Nevertheless, it follows from Wolpert and Macready’s [14] No Free Lunch (NFL) theorems¹ that there is no universal “best” fitness measure. The NFL theorems state that all general-purpose search algorithms perform equally well over the entire set of optimization problems. General-purpose search algorithms exploit little, if any, knowledge concerning the problem being solved [14]. EAs are an example of general-purpose search algorithms [14]. In particular, GP and most of its variants employing alternative fitness measures employ the same executional steps regardless of the problem being solved [4–13]. The NFL theorems state that in this scenario, if algorithm $a1$ outperforms algorithm $a2$ on a given set of problems, the converse is true on another set of problems [14]. In the context of this study, if fitness measure $f1$ outperforms fitness measure $f2$ on a given set of problems, the converse is true on another set of problems. Importantly, the literature also anticipates that applying different fitness measures at different points of the GP problem solving process should be more effective than applying a single fitness measure individually throughout the algorithm [6, 15]. Given the above arguments, what is required is some insight into the appropriateness of the different fitness measures; that is, insight should be provided with respect to when to use which fitness measure.

This research proposes the use of different fitness measures at different points in the evolutionary process. We refer to this alternation of fitness measures as dynamic fitness measures (DFMs), and a GP algorithm using DFMs as dynamic fitness measure genetic programming (DFMGP). In related work, McKay [6, 15] prescribes a ramped approach to applying two fitness measures, objective fitness (OF) and fitness sharing (FS), in GP. In

¹The NFL theorems are derived under the assumption of a “finite world”, whereby the search space of candidate solutions, and the mapping of the candidate solutions onto performance scores are finite [14]. These assumptions generally hold for search optimization algorithms run on digital computers [14].

the ramped approach, FS is applied in the initial 25% of the GP generations. In turn, OF is applied in the last 25% of the GP generations. In the intermediate GP generations, fitness is calculated as a linear ramp between FS and OF [6, 15]. The ramped approach is observed to achieve a performance advantage over both OF and FS applied individually throughout GP. Based on McKay's [6, 15] findings, it is anticipated that DFMGP can improve on the performance of the conventional GP approach. However, the arbitrary partitioning of the GP generations in the ramped approach (first 25% - FS; subsequent 50% - ramped; subsequent 25% - OF) may not achieve consistent results on varied problems; this is because different problems pose different challenges to GP [16], and may require different shortcomings to be addressed on different generations. DFMGP will differ from the ramped approach by using fitness measures drawn from a richer database of measures, and applying a different fitness measure (or combination of fitness measures) on each generation, whereby a higher-level search algorithm is used to approximate the best fitness measure to use on each generation for the given problem (or set of problems).

Selecting the best fitness measure (or fitness measure combination) to apply on each GP generation is in itself a combinatorial optimization problem, similar to selecting parameter values and operators for single point and multi-point searches. Genetic algorithms (GAs) [3], a class of EA, and GP have proven to be effective at solving such problems. For example, in the study conducted by Dioşan and Oltean [17], GAs are used to select which genetic operator to apply at each point in the evolutionary process. Similarly, GAs are used for parameter tuning of evolutionary algorithms in [18]. In addition, [19] and [20], are good examples of using GP to construct optimal combinations of existing evolutionary algorithm components; both studies use GP to evolve new, optimal crossover operators from the atomic elements that make up existing crossover operators [19, 20]. This research employs GA and GP to search the space of DFMs for DFMGP. The GA approach derives a sequence of fitness measures to be applied in DFMGP, while GP generates new fitness measures by combining existing fitness measures. The distinguishing feature of the GA and GP approaches is that they operate at the higher (or meta-) level to search the space of DFMs for DFMGP. At the lower level, DFMGP attempts to solve the underlying problem.

Proceeding derivation of the DFMs, the effectiveness of DFMGP is determined. The performance of DFMGP applying the DFMs is compared with that of standard GP. The study also investigates the reusability of the evolved DFMs. DFMs are evolved for different problem classes² using a training set and tested on unseen instances of the class. The purpose is to evolve problem solvers that can generalize within a problem class. DFMGP is also applied to complex, real-world problems. Here, DFMs evolved by training on the problem classes are tested on real-world problems from the same problem domain. The idea is to train on simpler and less complex problems that will not take as much time to train on, and subsequently employ the evolved DFMs on the more complex problems.

1.2 Research perspective

The research investigates the hypothesis that DFMGP is more effective than the conventional GP approach. The performance of DFMGP is compared with standard GP on a broad spectrum of benchmark and real-world problems. Importantly, insight into the best practices for deriving DFMs is gained by using two different approaches to approximate optimal DFMs for GP: GA and GP.

The research also investigates the reusability of the derived DFMs within problem classes, and on more complex problems from the same problem domain. The implications of reusability are that the DFMs are not problem-specific, rather, GP practitioners can save time by deriving more general problem solvers.

²The literature [21] defines a problem class as a probability distribution over the instances of a given problem; this is the interpretation of the term used in the study. The term problem domain is also used: a problem domain is a grouping of similar problems e.g. symbolic regression problems, Boolean function synthesis problems, etc. [1].

Lastly, an analysis of the DFMs that produce the biggest performance improvements will offer the opportunity to understand the GP problem solving process better, from the point of view of identifying the fitness measures that are the most useful in the different phases of search for different problems.

1.3 Scope

The choice of fitness measure potentially affects every GP system. However, this thesis focuses on using DFMs to improve on abstract-syntax-tree GP, the canonical GP system proposed by Koza in [1]. Furthermore, while other optimizers can be used, this study employs GA and GP to approximate optimal DFMs for DFMGP.

1.4 Objectives

The objectives of this research are:

1. Apply GAs for evolving DFMs for DFMGP.
2. Apply GP for evolving DFMs for DFMGP.
3. Compare the performance of DFMGP with the conventional GP approach.
4. Compare the performance of GAs and GP in evolving DFMs.
5. Assess the reusability of the evolved DFMs.
6. Analyse the best performing DFMs to identify the fitness measures that are most useful in the different phases of search for different problems.

1.5 Contributions

This thesis makes the following contributions:

1. A detailed survey of the fitness measures prescribed for GP is conducted. To the author's knowledge, such an extensive survey of the different fitness measures has not previously been conducted. The survey discusses the fitness measures in detail, with respect to their motivations, advantages and disadvantages. Based on the theory underlying the fitness measures, inferences are made with respect to the different fitness measures that suit different problems.
2. A comparison of state-of-the-art fitness measures is conducted. The aim is to evaluate the effect that the different fitness measures have for different problems. The study looks at how the different fitness measures address each of the limitations they aim to overcome for different problems. There is currently a lack of comparison between the different fitness measures proposed in the literature: most studies simply compare against an OF measure, otherwise state-of-the-art fitness measures are not compared. The comparison in this study verifies the NFL theorems: no one fitness measure achieves the best result on all tackled problems, rather the performance of the fitness measures depends on the properties of the specific problem being tackled.
3. The study shows that DFMGP is more effective than the conventional GP approach. An analysis of the GA/GP-evolved DFMs reveals that fitness measures applied in the preliminary DFMGP generations support exploration while those applied in later DFMGP generations support exploitation. Exploration and exploitation are the two cornerstones of problem solving by search [22]: exploration, a de facto global search, is used to promote coverage of the search space [22]; in turn, exploitation, a de facto local search, is used to refine promising solutions when good points in the search space have been discovered

[22]. GP search is a constant balance between exploration and exploitation, with the former being more suited to the preliminary generations, and the latter, later generations. DFMGP's performance advantage over standard GP is therefore justified by the premise that the fitness measure used on each generation supports the more suitable search in the on-going phase of GP.

4. The study compares the use of GP versus the use of a GA to evolve DFMs for DFMGP, and finds that the former approach yields more effective DFMs. While GA derives a sequence of fitness measures to be applied in DFMGP, GP produces an arithmetic function combining the fitness measures. Therefore, rather than simply switching between explorative and exploitative fitness measures, the latter approach offers the opportunity to combine the two types of fitness measure, driving exploration and exploitation to occur simultaneously when required in DFMGP. This proves useful, because rather than moving between strict explorative and exploitative phases, GP search ideally maintains a constant trade-off between the two modes of search [22].
5. The study shows that the DFMs derived by GA and GP are reusable, i.e. DFMs can be evolved for a problem class and yield good results on unseen problem instances of the class. The DFMs evolved for a problem class can also be reused on unseen complex, real-world problems from the same problem domain, where DFMGP is shown to achieve better results than standard GP.

1.6 Thesis layout

This thesis is organized as follows:

Chapter 2: An Introduction to Genetic Algorithms and Genetic Programming

Chapter 2 introduces genetic algorithms (GAs) and genetic programming (GP), and discusses the two EAs in detail.

Chapter 3: A Survey of Fitness Measures in GP

Chapter 3 conducts a survey of the different fitness measures prescribed for GP. The fitness measures are described with respect to their motivations, advantages, disadvantages and problem suitability.

Chapter 4: A Comparison of Fitness Measures in GP

Chapter 4 presents a comparison of state-of-the-art fitness measures. The fitness measures are compared on benchmark problems with respect to different criteria, such as the best solution quality achieved, mitigating bloat and overfitting, etc.

Chapter 5: Methodology

Chapter 5 outlines the methodology used to achieve the objectives of the study. The objectives of the study are restated and details are provided with respect to how they will be achieved and measured. The details of the benchmark and real-world problems tackled are given. Also, the details for statistical testing of DFMGP are presented. Finally the technical specifications are provided.

Chapter 6: The DFMGP Algorithm

Chapter 6 presents the DFMGP algorithm. Details are given of the representation used, initial population creation, fitness evaluation, selection method, genetic operators and termination conditions.

Chapter 7: A GA Approach for Deriving DFMs for DFMGP

Chapter 7 presents the GA approach used for deriving DFMs for DFMGP. Details are given of the GA representation used, initial population creation, fitness evaluation, selection method, genetic operators and termination conditions.

Chapter 8: A GP Approach for Deriving DFMs for DFMGP

Chapter 8 presents the GP approach used to derive DFMs for DFMGP. This chapter follows the same format as Chapter 7 providing details of the GP representation used, initial population creation, fitness evaluation, selection method, genetic operators and termination conditions.

Chapter 9: Results and Discussion

Chapter 9 presents the results of the experiments described in Chapter 5. DFMGP and standard GP are compared on benchmark and real-world problems with respect to the best solution quality achieved. Statistical tests are conducted to verify the significance of the empirical observations. Furthermore, the implications of the results are discussed in detail.

Chapter 10: Conclusions and Future Work

Chapter 10 presents a summary of the findings from the research presented in this thesis, and indicates how the objectives presented in Chapter 1 have been met. Chapter 10 also provides directions for future work.

Chapter 2

An Introduction to Genetic Algorithms and Genetic Programming

2.1 Introduction

Genetic algorithms (GAs) and genetic programming (GP) differ principally with respect to the space that is searched. GAs search a solution space, which means that they search a space of potential values of a set of parameters or variables being optimized [3]. In turn, GP searches a program space, whereby the optimal program is implemented to find a solution to the problem [1]. Hence in GP, the program behavior that produces the desired solutions is evolved, rather than the solutions themselves being evolved [1]. Both GAs and GP are classified as evolutionary algorithms (EAs). EAs model Darwinian evolution [23], by using the principles of genetic variation and natural selection to approximate an optimal solution [1, 3]. In an EA, an initial population of candidate solutions is generated randomly. Subsequently, a fitness measure assigns each solution a numerical “fitness” value. The fitness value is used to bias the selection of promising (or high-fitness) candidate solutions, while less promising solutions are systematically eliminated [1, 3]. Genetic operators, such as mutation and crossover, are probabilistically applied to the high-fitness solutions to move the search to different regions of the search space, globally (using mutation) and locally (using crossover) [1, 3]. The genetic operators produce subsequent generations of candidate solutions. Subsequent generations are produced either synchronously - whereby the old generation is completely replaced - or asynchronously - whereby the generations overlap. The algorithm is considered successful if iterative cycles of fitness evaluation, selection and regeneration evolve optimal (or near-optimal) solutions [1, 3].

The previous chapter established that a key focus area of the research is applying DFMs in GP. Here, GAs and GP are employed at the higher (or meta-) level to approximate optimal DFMs for DFMGP. In this regard, sections 2.2 and 2.3 lay a background for the research by describing the GA and GP algorithms in detail. Finally, section 2.4 summarizes the information presented in the chapter.

2.2 Genetic algorithms

GAs, initiated by Holland [24], are one of the earliest implementations of evolutionary algorithm search. A GA attempts to discover an optimal solution to a problem by genetically breeding a population of candidate solutions over a sequence of generations of the algorithm [3, 24]. In GAs, the candidate solutions are represented as “chromosomes” [3, 24]. Typically, a chromosome is a Boolean, character or numerical string, whereby each locus within the string, referred to as a “gene”, denotes the value of a particular feature of the problem being tackled [3, 24]. The term “allele” is used to refer to the value assigned to a gene within a chromosome.

This section provides an overview of GA. First, the overall GA approach is summarized. Subsequently, a discussion is presented on the mechanics of GA: 1) representation scheme, 2) initial population creation, 3) fitness evaluation and selection, and 4) genetic operators. Lastly, a critical analysis of GAs is presented.

2.2.1 The genetic algorithm

The GA presented by Goldberg [3] is depicted in listing 2.1. This is considered to be one of the earliest formalizations of GAs [3]. The algorithm in listing 2.1 is referred to as canonical GA for the remainder of the manuscript.

LISTING 2.1: Pseudocode for the Canonical GA; adapted from [3].

```

1 Begin
2 Create an initial population and evaluate.
3 While termination criteria are not met:
4     While creating a new population is not complete:
5         Select two individuals from the current population,  $i_1$  and  $i_2$ .
6         If crossover probability,  $p_c$ :
7             Perform crossover on  $i_1$  and  $i_2 \rightarrow$  creating offspring  $i_3$  and  $i_4$ 
8         Else:
9             Copy  $i_1$  and  $i_2 \rightarrow i_3$  and  $i_4$ .
10        End If.
11        If mutation probability,  $p_m$ :
12            Perform mutation on  $i_3$  and  $i_4$ .
13        End If.
14        Evaluate  $i_3$  and  $i_4$ .
15        Insert  $i_3$  and  $i_4$  into the new population.
16    End While.
17    Replace current population with the new population.
18 End While.
19 End.

```

The canonical GA begins by creating a random initial population, and evaluating the fitness of each individual in the population (line 2). Next, while the user-specified termination criteria are not met, GAs iterate through cycles of fitness-based selection and regeneration. Each cycle is called a generation. In listing 2.1, a GA generation is depicted by a loop that repeats until creating a new population from the current population of candidate solutions is complete (lines 4-16). The probability that an individual is selected to parent the new population (line 5) is higher for more promising individuals. Also, parent selection is done with replacement, such that the same individual can be selected more than once to become a parent. Genetic operators are applied to the selected parents, i_1 and i_2 (lines 6-13). Crossover is applied with probability p_c : if a randomly generated number in the interval $[0, 1]$ is less than p_c , i_1 and i_2 are crossed over to yield i_3 and i_4 (line 7); otherwise if the random number is outside of the range of p_c , i_1 and i_2 are copied into i_3 and i_4 (line 9). Following crossover, mutation is applied with probability p_m : if a randomly generated number in the interval $[0, 1]$ is less than p_m , mutation is applied to both i_3 and i_4 (line 12); otherwise mutation is not applied. Next, the fitness of the generated offspring is evaluated (line 14), and the offspring are added to the accumulating new population (line 15).

The new population replaces the current population once the former has accumulated the same number of individuals as the latter (line 17). Complete replacement of the current population with the new population, as shown in listing 2.1, is known as the generational control model. The steady-state model is an alternative control model for GAs [3]: in the steady-state model, a single overlapping population is maintained, whereby the offspring repeatedly replace a proportion of the individuals in the population [3]. The steady-state model requires an additional parameter, the number of individuals to be replaced on each generation, to be defined. Goldberg [3] reverts to a generational control model to maintain the simplicity of canonical GAs.

GAs are inherently convergent [3]. The fitness-based selection of candidate solutions as the parents of future generations leads to a more focussed search in the promising regions of the search space. This is such that

the genetic material associated with high-fitness candidate solutions eventually dominates the GA population, while the genetic material associated with low-fitness solutions is supplanted [3]. The simple procedure shown in listing 2.1 is the basis for most applications of GAs. Nevertheless, there are a number of important details to fill in, as discussed in the ensuing sections.

2.2.2 Representation scheme

At inception [3], the GA chromosomes used were fixed-length binary strings; i.e. fixed-length strings comprised of 0s and 1s, as depicted in figure 2.1.

Chromosome 1: 00010111001111101100001111011100
 Chromosome 2: 1000011111110110000100101100101

FIGURE 2.1: GA fixed-length binary representation: Each bit (or group of bits) at a fixed locus represents the value of a variable being evolved.

The binary encoding offers the advantage of a small alphabet (i.e. a two digit alphabet of 0s and 1s). Small alphabets result in longer chromosomes required to represent the same range of values: an alphabet of cardinality k can represent k^l different values, whereby l is the length of the chromosome strings; it follows that the smaller k is, the larger l must be in order for the encoding to represent the same range of values [3]. Longer chromosomes are desirable because they potentially offer more opportunity for useful allele combinations to occur along the length of the chromosomes, enhancing search [3]. The binary encoding is intuitive for problems where binary variables are being evolved, such that each bit in a chromosome string can represent the value of a variable [3]. In the case of non-binary variables being evolved, a pre-processing step is required, whereby preceding GA optimization, the variables are converted to binary form; here, some of the variables may be represented as a sequences of bits within the chromosome strings, such that caution should be exercised to ensure that genetic operators working on the individual bits to not produce invalid values for the variables [25]. After an optimal solution is found, post-processing is also required to map the solutions found back onto the space of the original variables [25].

Alternatively, direct value encodings are also used, whereby the chromosomes can contain characters, numbers or any objects directly connected to the problem [25], as depicted in figure 2.2.

i) Character encoding
 Chromosome 1: AAABBADCDDDDAAGGGGGCCDDAAAAA
 Chromosome 2: BBBABADCDDDDAAGGGGGCCDDAAAAA

ii) Real number encoding
 Chromosome 1: 1.5 2.0 2.5 0.3 0.1 0.9 2.5 3.3 2.0
 Chromosome 2: 1.6 1.0 2.9 0.3 1.1 2.9 2.0 3.3 1.0

FIGURE 2.2: GA fixed-length direct value representation: Each character or number at a fixed locus represents the value of a variable being evolved.

In the current study, a higher (or meta-) level GA will be used to search the space of DFMs for DFMGP. Similar studies that have used GAs to optimize the configuration of lower-level EAs have employed binary [26–28], as well as direct value encodings [17, 29–31]. For example Grefenstette [26] used a binary encoding in a secondary GA that tuned the numerical parameters of a primary GA, including the population size, crossover rate and mutation rate of the latter. This involved pre-processing to translate the parameters evolved by the secondary GA to binary form, and post-processing to translate the evolved solutions back to the original parameters. Grefenstette’s secondary GA [26] was found to evolve parameters that produced a significant

improvement in the performance of the primary GA [26]. More recent work has employed direct value encodings. For example, in [29], a secondary GA employing a direct value encoding is used to optimize the numerical parameters of a primary GA. The secondary GA in [29] is found to derive optimal parameters for the primary GA on a complex real-world problem. In [17], Dioşan and Oltean use a GA approach to design an EA, whereby the GA optimizes the sequence of genetic operators utilized by the EA during a generation. The GA chromosome representation in [17] consists of an array of integers that indicate the chromosome creation and population altering operations performed by the EA being evolved; here, each possible EA operation is associated with an integer - e.g. the integers 0 and 1 represent crossover and mutation respectively. Dioşan and Oltean's GA approach [17] is shown to evolve EAs that perform better than standard EA approaches adopted in the literature. In general, direct value encodings have become more popular in recent applications of GAs because they offer an intuitive representation of the search space, and a simplified GA implementation, whereby pre-processing and post-processing steps are not required.

2.2.3 Initial population creation

In GAs, the initial population is generated randomly, whereby for each chromosome, the allele assigned to each gene is drawn from the set of available values with uniform probability [3]. Random initial population creation facilitates broad coverage of the search space of solutions, whereby the initial population samples a significant proportion of the universe of allele combinations [3]. The randomly generated chromosomes in the initial population are not likely to do very well; nevertheless, a large enough proportion of the initial population should, by random chance, do better than the rest of the population [3]. GAs rely on this initial fitness gradient, which provides a basis for "natural selection" [3].

2.2.4 Fitness evaluation and selection

In several problems, the objective of GA search is easily stated as the minimization of some cost function, or the maximization of some utility or profit function [3]. This allows for the fitness measure to be specified as the relevant objective function, whereby "fitter" solutions minimize (or maximize) the distance to the search objective [3].

The objective fitness measure described above is used to select the most promising population individuals as parents of subsequent generations. In canonical GA, Goldberg [3] applies fitness-proportionate selection. The metaphor used for fitness-proportionate selection is a biased roulette wheel, whereby each individual in the current population is allocated a slot on the wheel in proportion to the ratio of its fitness value to the total fitness of the current population [3]: here, selection is akin to spinning the wheel, whereby the higher the fitness of an individual, the more likely the individual is selected [3]. Alternatively, another popular selection method used in the literature is tournament selection [32]. In tournament selection, the GA practitioner specifies a tournament size, k . To select a parent using tournament selection, k individuals are drawn from the population at random; subsequently, only the individual with the highest fitness value is retained as the result of the selection [32].

Fitness proportionate selection employs a strict favoritism of the high-fitness candidate solutions, potentially driving search towards rapid convergence [32]. On the other hand, tournament selection is comparatively less strict: *for small values of k* , the probability that low-fitness individuals are also selected increases. The effect is that tournament selection slows down the convergence of GAs. This is desirable because GAs can make mistakes, whereby alleles with small contributions to the objective fitness become fixed at some of the loci, due to early spurious associations with other highly fit alleles [3, 32]. In this case, rapid convergence will also mean the proliferation of the low-fitness alleles at the expense of more useful alleles at the same loci. Conversely, the slower convergence effected by employing tournament selection *with small values of k* mitigates the loss of potentially useful genetic material [32]. Overall, tournament selection offers the opportunity to control the convergence rate of GA by manipulating the value of k [32].

2.2.5 Genetic operators

Crossover is the principal genetic operator used in canonical GA [3]. In turn, mutation is a secondary genetic operator also applied in canonical GA [3]. The GA genetic operators are discussed in detail in the text below.

Crossover

Given two parent chromosomes, crossover randomly selects the same point in both parents; subsequently, all data beyond the selected point in either parent is swapped between the parents to yield two offspring [3, 25]. Figure 2.3 depicts crossover.

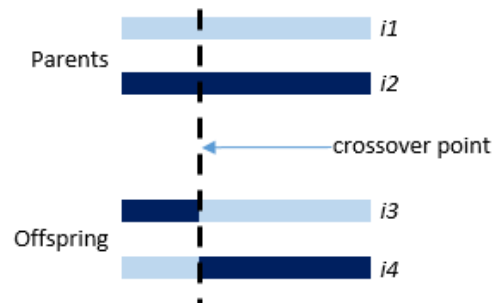


FIGURE 2.3: GA one-point crossover

Variants of the crossover operator described above include two-point and uniform crossover. Two-point crossover calls for two random points to be selected in both parents, whereby all the data between the selected points is swapped between the parents [25], as shown in figure 2.4.

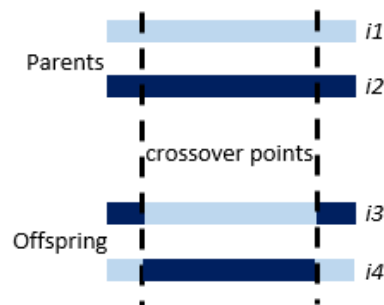


FIGURE 2.4: GA two-point crossover

In uniform crossover, each gene in the first offspring originates from the first parent with probability p_u , and from the second parent with probability $1 - p_u$. In turn, each gene in the second offspring originates from the second parent with probability p_u , and from the first parent with probability $1 - p_u$. If p_u is set to 0.5, the offspring have half of the genes from the first parent and the other half from the second parent [25]. Uniform crossover is shown in figure 2.5.

Two-point and uniform crossover cause more disruption of the parent chromosomes, whereby given genetically diverse parents, the offspring differ more from either parent [33]. This effect is argued to lead to a more detailed search of the promising regions of the search space, whereby the two operators are more thorough with respect to trying out different allele combinations [33]. A key advantage of uniform crossover is that the genes are treated separately and inherited independently of position, encouraging a more detailed search of the optimal alleles for each gene [33]. In an experiment conducted in [33], two-point and uniform crossover are shown to consistently outperform one-point crossover on a number of benchmark problems.

Overall, the bulk of GA's search power stems from fitness-based selection combined with crossover [3]. Fitness based selection ensures that candidate solutions that have performed well in previous generations are repeatedly tested, whereas crossover facilitates the exchange of genetic information between the solutions [3].

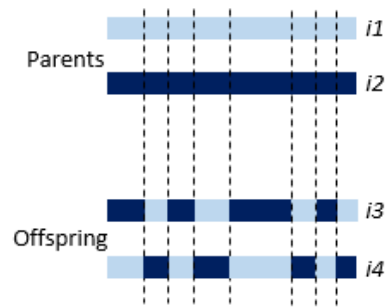


FIGURE 2.5: GA uniform crossover

Goldberg [3] draws a parallel to the manner in which innovation occurs in human interaction: innovation most often occurs by combining different ideas that have worked well in the past to give birth to new ideas. In the same way, fitness-based selection and crossover search the space of allele combinations that have worked well in previous generations with the aim of improving on the allele combinations [3].

Mutation

Mutation produces variation in a single parent candidate solution, by modifying the allele at a randomly selected point. For example, if the chromosomes used are binary strings, mutation flips the bit at the selected mutation point [3, 25], as shown in figure 2.6.



FIGURE 2.6: GA one-point mutation of a bit string

If the chromosomes used are character strings, mutation involves replacing the character at a randomly selected point with another character drawn with uniform probability from the alphabet of available characters [25]. Figure 2.7 shows an example of mutation in a character representation.

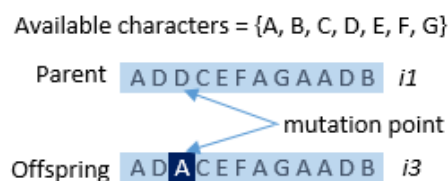


FIGURE 2.7: GA one-point mutation of a character string

In turn, if the chromosomes used are numerical strings, mutation involves manipulating the number at a randomly selected point e.g. by adding or subtracting a random number drawn from a Gaussian distribution with a mean of zero [34]. Figure 2.8 shows an example of mutation in a numerical representation.

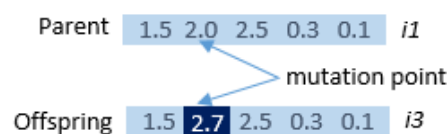


FIGURE 2.8: GA one-point mutation of a numerical string

Variants of the one-point mutation operators shown in the above figures include two-point and uniform mutation. In two point mutation, two random points in a chromosome are selected, and the genes at the

selected points mutated [35]. In turn, in uniform mutation, each gene in a chromosome string has a probability p_u of being mutated [35, 36]. Two-point and uniform mutation impose larger jumps in the search space, whereby the offspring produced are more diverse from their parents.

GA mutation plays the important role of restoring lost diversity [32]. For example, in cases where low-fitness alleles become fixed at a given locus, random mutation at the locus can lead to the rediscovery of more useful alleles [32]. Hence mutation potentially rescues the search from converging to sub-optimal regions of the search space [32]. Nevertheless, low mutation probabilities are typically applied in GAs, in order to prevent the evolution from degenerating into a random search [3, 25].

2.2.6 A critical analysis of genetic algorithms

The convergent nature of GAs was established in section 2.2.1. In the literature [3, 25], the term “global optimum” is used to refer to the most optimal points in a search space. GA convergence is useful if search has discovered solutions that are in the basin of a global optimum, such that population convergence leads to the optimum [3, 25]. Conversely, a “local optimum” is an optimal point within a neighbourhood of candidate solutions, which gives the false illusion of a global optimum within the neighbourhood [3, 25]. GA convergence can lead to search becoming stuck at a local optimum. This undesirable occurrence is termed premature convergence. Nevertheless, various measures can be undertaken to mitigate premature convergence, including employing tournament selection with a small tournament size [32], and increasing the mutation rate [32]. More disruptive genetic operators, that effectively perturb the GA population out of the local optima, can also be defined.

In another vein, GAs are stochastic in nature, whereby each run of a GA on a given problem produces a different result [3]. This is because a number of steps in the GA algorithm involve random number generation: creating a random initial population, evaluating whether or not to apply the genetic operators based on probability, as well as selecting random crossover and mutation points all involve the use of random numbers. GAs use random choice as a tool to guide search towards regions of the search space with *likely* improvement; this is because the exact path to improvement is not known, hence the purpose of search. Essentially, GAs perform a directed search with the intervention of chance [3]. A repercussion of the stochastic nature of GAs is that when reporting on the performance of a GA on a problem, it is common practice to conduct a number of runs of the algorithm, and subsequently report on the average performance over the total number of runs conducted. Hence additional GA runs are required, which can be costly for computationally expensive problems.

GAs also require parameter tuning, whereby the choice of genetic operator probabilities, selection method, population size, as well as the termination criteria specified all impact on the performance of the algorithm. Different parameter configurations suit different problems [30]; for example, tournament selection with a small tournament size may be more appropriate for problems prone to local optima, rather than trivial problems. Hence the GA practitioner is required to specify a suitable parameter configuration for each newly encountered problem. Parameter tuning is in itself a non-trivial problem [37]: optimal GA parameters can be estimated empirically, where the practitioner experiments with different configurations and chooses the parameters that produce the best result, or by the use of parameter tuning algorithms [38–40]. Hence parameter tuning is also associated additional computation, required to find a suitable configuration for the given problem.

Despite the above-mentioned challenges, GAs demonstrate a number of advantages compared to other search optimizers. GAs conduct the same executional steps of fitness-based selection and regeneration, regardless of the specific problem being tackled [3, 24]. Therefore, GAs are general problem solvers, which can be applied on different problems, as long as suitable chromosome representations and fitness measures are defined. GAs have been applied to problems from different fields, including problems found in biology, computer science, operations research, and the social sciences [3, 25]. GAs also offer an advantage over point-to-point search methods, such as simulated annealing [41, 42], tabu search [41], and hill-climbing [41]. Goldberg [3] argues that point-to-point search is dangerous because it almost guarantees the location of local optima

in multi-modal (many peaked) search spaces. By contrast, GAs adhere to the old adage that there is safety in numbers, and maintain a population of candidate solutions [3]. GA search works from a database of multiple search points, which give the algorithm the capability to climb multiple search space optima in parallel. As a result, the probability of becoming stuck in local optima is reduced over point-to-point methods [3].

2.3 Genetic programming

GP, initiated by Koza [1], searches a program space. Therefore, rather than evolving parameters or variables as in the case of GAs, GP evolves executable code. At inception [1], GP used an abstract syntax tree (AST) representation. An AST is a parse tree model of an entire program, whereby each node of the tree denotes a construct occurring in the program source code [1]. ASTs are “abstract” because they abbreviate the detail appearing in the concrete program: for example, a syntactic construct like an *if-condition-then* expression may be denoted by means of a single tree node with three branches [1]. ASTs are easily manipulated and transformed, while preserving the existence of a formally defined program through the transformations [1].

A key aspect of the GP paradigm is the use of a variable-length representation. The variable-length representation allows the program structure to evolve in problems where it is difficult to a priori specify the size and structure of the optimal program [1]. Drawing a parallel with GA, the syntax trees evolved by GP can also be referred to as chromosomes. Nevertheless, the concept of a gene being a fixed locus within a chromosome, as in canonical GA, does not apply. This is because the GP crossover operator exchanges genetic material from different loci (see section 2.3.5). Hence there are no dedicated loci within the GP chromosomes; rather, as stated above, the structure of the programs evolves. GP search aims to find a chromosome, which when decoded into a candidate program and executed, yields the desired program behavior [1].

This section provides an overview of GP. First, the overall GP approach is summarized. Subsequently, a discussion is presented on the mechanics of GP: 1) representation scheme, 2) initial population creation, 3) fitness evaluation and selection, and 4) genetic operators. Lastly, a critical analysis of GP is presented.

2.3.1 The genetic programming algorithm

The GP algorithm presented by Koza [1] is depicted in listing 2.2. The algorithm in listing 2.2 is referred to as canonical GP for the remainder of the manuscript.

The following difference is observed between canonical GP and canonical GA: in GP, p_c and p_m are genetic operator application rates, and not application probabilities as in GAs. This is such that in GP, $n\%$ of the population is created using a particular genetic operator. Nevertheless, GAs and GP employ the same cycles of fitness evaluation, selection and regeneration. Canonical GP generally applies the generational control model [1], however steady-state GPs have also been implemented in the literature [43, 44].

LISTING 2.2: Pseudocode for the Canonical GP; adapted from [1].

```

1 Begin
2 Create an initial population and evaluate.
3 While termination criteria are not met:
4     While creating a new population is not complete:
5         Create  $p_c\%$  of the new population using crossover:
6             Select two individuals from the current population,  $i_1$  and  $i_2$ .
7             Perform crossover on  $i_1$  and  $i_2$  → creating offspring  $i_3$  and  $i_4$ .
8             Evaluate  $i_3$  and  $i_4$ .
9             Insert  $i_3$  and  $i_4$  into the new population.
10        Create  $p_m\%$  of the new population using mutation:
11            Select one individual from the current population,  $i_1$ .
12            Perform mutation → creating offspring  $i_3$ .

```

```

13         Evaluate  $i3$ .
14         Insert  $i3$  into the new population.
15     Create  $(1 - (p_c + p_m))\%$  of the new population using reproduction:
16         Select one individual from the current population,  $i1$ .
17         Copy  $i1 \rightarrow i3$ .
18         Evaluate  $i3$ .
19         Insert  $i3$  into the new population.
20     End If.
21 End While.
22     Replace current population with the new population.
23 End While.
24 End.

```

Like GAs, GP is an inherently convergent algorithm, whereby the fitness measure steers search towards promising regions of the search space; here, the genetic material associated with high-fitness solutions eventually dominates the GP population, while the genetic material associated with low-fitness solutions is supplanted [1]. The procedure shown in listing 2.2 is the basis for most applications of GP. The ensuing sections discuss the mechanics of GP in detail.

2.3.2 Representation scheme

The AST GP parse trees are built using primitive types called functions and terminals [1]. Functions that require arguments make up the internal nodes of the parse trees. In turn, terminals can be constants, input variables or functions that do not require arguments, and make up the terminal nodes of the trees. The term “arity” is used to refer to the number of arguments input to a primitive. For example, the arithmetic function “+” has an arity of 2 because it takes two arguments. Terminals have an arity of 0 because they have no arguments.

Prior to applying GP, the GP practitioner specifies the function and terminal alphabets, referred to as the function and terminal set respectively [1]. The choice of function and terminal set is usually problem specific. For example, evolving a program to approximate an arithmetic function would require a function and terminal set comprised of arithmetic primitives; in turn, evolving a program to approximate a Boolean function would require a function and terminal set comprised of Boolean primitives [1]. Figure 2.9 shows an example of an arithmetic parse tree, whereby $\{+, -, \times\}$ represent arithmetic functions drawn from a function set, and the constant 1 and variables $\{a, c, d\}$ are drawn from a terminal set. The tree in figure 2.9 would be the same as the inorder traversal $(1 - (1 + a)) + (a \times (c \times (d + a)))$.

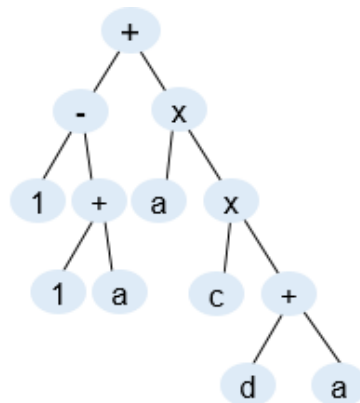


FIGURE 2.9: Example GP parse tree

Importantly, Koza [1] specifies that the function and terminal sets used for GP should satisfy the *sufficiency* and *closure* properties. The *sufficiency* property requires that the functions and terminals provided to GP are

capable of expressing a solution to the problem under consideration [1]; otherwise a solution cannot be expected to be found if the function and terminal sets used are “incomplete” in this respect [1].

The closure property has two components: 1) *evaluation safety*, and 2) *type consistency* [45]. *Evaluation safety* ensures that all functions are well defined for any arguments received [1, 45]. This is achieved by modifying the standard behavior of primitives. For example, it is common practice to use protected versions of arithmetic functions that can throw exceptions, such as the division, logarithm and square root functions. As an illustration, the arithmetic division function always returns a number, unless the denominator of the division is 0, in which case the result of division is undefined. To satisfy the *evaluation safety* constraint, the division operator has to be redefined, such that an acceptable value is returned as a result of division by 0. In this vein, when applying GP to solve arithmetic problems, Koza [1] defines a protected division operator (denoted by %), for which the result of division by 0 is 1.

Type consistency is necessitated by the fact that the crossover operator used in GP arbitrarily selects a subtree from one parent to place it in another [1, 45]. Therefore any function in the function set should accept as any argument any terminal from the terminal set, as well as any value returned by any other function from the function set [1, 45]. This means that all functions should return values of the same type, and that all their arguments are constrained to that type as well [45]. Under the *type consistency* constraint, canonical GP is not designed to handle a mixture of data types. In this vein, Koza [1] describes a way to relax the constraint and incorporate different data types in GP, using the concept of constrained syntax structures [1]. Constrained syntax structures directly specify the child nodes that each function can have, whereby the child nodes are restricted to a list of legal types for each function argument [1]. Strongly typed genetic programming (STGP) [46] builds on the concept of enforcing type constraints. In STGP, the accepted data types for each argument of each function are specified; furthermore, the data types returned by each function and terminal are also specified [46]. Hence in STGP, legal programs are maintained by ensuring that the correct functions and terminals are supplied as arguments to all functions. The random initial population creation routine is modified such that random but valid arguments are supplied to all functions, hence only legal trees are created. For crossover, the change means that only subtrees rooted at identical types can be exchanged between individuals [46]. For mutation, the types of the selected mutation point and the root of the randomly generated subtree substituted into the mutation point should match; furthermore, creating the random subtree is done following the same method as the initial population creation to create a legal subtree [46].

In the current study, a higher (or meta-) level GP will be used to search the space of DFMs for DFMGP. Examples of both STGP and canonical (i.e. untyped) GP are seen in meta-level search in the literature. In [47], GP is used to search the space of search algorithms employed to solve the travelling salesman (TSP) problem. The GP approach proposed in [47] is strongly-typed to accommodate the types of the various primitives useful to the lower level search algorithms. The results in [47] show that GP evolves optimal search algorithms for the TSP problem. In [48], GP is used to evolve the crossover operators used by a lower-level GA to solve function optimization problems; here, each GP chromosome encodes a crossover operator which contains arithmetic symbols (mathematical operators, constants and some variables). All functions defined in the GP function set in [48] need only accept and return arithmetic types, such that typing is not required. The GP approach in [48] is shown to perform well, evolving crossover operators that facilitate better GA performance on a number of problems compared to standard GA operators adopted in the literature.

2.3.3 Initial population creation

As in the case with GAs, the GP initial population is generated randomly. Koza [1] specifies three methods for initial population creation, namely full, grow and ramped half-and-half.

Full

The full method is depicted in listing 2.3. In listing 2.3, each initial population individual is generated by using a depth-first recursive approach to assign functions to all nodes that occur prior to reaching the maximum tree depth, m ; beyond that, only terminal nodes are assigned. Individuals created with this method will have the same tree depth. The individuals are however likely to differ in shape and structure, unless all functions in the function set have an equal arity. A key objective in creating the initial population is a diversity of shapes and structures, facilitating broad coverage of the search space [1]. In this case, the range of program shapes and structures produced by the full method may be quite limited, even with the use of mixed-arity function sets.

LISTING 2.3: Pseudocode for Full; adapted from [1].

```

1 Begin
2 Set the maximum tree depth,  $m$ .
3 For Each element of the population:
4     Set current depth = 1.
5     Set current node = the root node.
6     If current depth <  $m$ :
7         Select a function,  $f$ , from the function set with uniform probability.
8         Set the value of current node to  $f$ .
9         Assign  $n$  child nodes to the current node, whereby  $n$  is the arity of  $f$ .
10        Set current depth = current depth + 1.
11        For Each assigned child node  $c$ :
12            Set current node =  $c$ .
13            Recurse from line 6.
14        End For Each.
15    Else:
16        Select a terminal,  $t$ , from the terminal set with uniform probability.
17        Set the value of the current node to  $t$ .
18    End If.
19 End For Each.
20 End.

```

Grow

The grow method is depicted in listing 2.4. The grow method is similar to the full method. However the key difference is that both function and terminal nodes can be selected at any depth. The grow method allows for the creation of an initial population with more varied tree shapes and structures. Nevertheless, the number of full trees occurring in the initial population may be limited.

LISTING 2.4: Pseudocode for Grow; adapted from [1].

```

1 Begin
2 Set the maximum tree depth,  $m$ .
3 For Each element of the population:
4     Set current depth = 1.
5     Set current node = the root node.
6     If current depth <  $m$ :
7         Select the node type from the set  $\{F, T\}$ , with uniform probability.
8     Else:
9         Set the node type to  $T$ .
10    End If.
11    If node type is  $F$ :
12        Select a function,  $f$ , from the function set with uniform probability.

```

```

13     Set the value of current node to  $f$ .
14     Assign  $n$  child nodes to the current node, whereby  $n$  is the arity of  $f$ .
15     Set current depth = current depth + 1.
16     For Each assigned child node  $c$ :
17         Set current node =  $c$ .
18         Recurse from line 6.
19     End For Each.
20     Else If node type is  $T$ :
21         Select a terminal,  $t$ , from the terminal set with uniform probability.
22         Set the value of the current node to  $t$ .
23     End If.
24 End For Each.
25 End.

```

Ramped half-and-half

The ramped-half-and-half method is depicted in listing 2.5. Ramped-half-and-half is proposed because the full and grow methods are incapable of providing a wide variety of tree shapes and structures on their own. The proposed method combines the full and grow methods: using a range of depth-limits (hence the term “ramped”), half the initial population is constructed using the full method, and the other half constructed using the grow method. Ramped half-and-half is the sole method actually implemented by Koza in [1].

LISTING 2.5: Pseudocode for Ramped half-and-half; adapted from [1].

```

1 Begin
2 Set the maximum tree depth,  $m$ .
3 Partition the population evenly into  $m - 1$  parts.
4 Set current partition = the first partition.
5 Set the partition tree depth  $d = 2$ .
6 While there are still empty partitions:
7     Generate half the individuals for the current partition using the full method with maximum tree depth  $d$ .
8     Generate the other half of the individuals for the current partition using the grow method with maximum tree depth  $d$ .
9     Set current partition = the next partition.
10    Set  $d = d + 1$ .
11 End While.
12 End.

```

2.3.4 Fitness evaluation and selection

The fitness evaluation in canonical GP is identical to canonical GA, whereby the convention is to specify the fitness measure as the minimization of some cost function, or the maximization of some utility or profit function directly related to the search objective. Nevertheless, recall that since the inception of GP, different alternatives to objective fitness measures have emerged in an attempt to overcome the different shortcomings of GP [4–13]. A key milestone in this research is the study of the different alternative fitness measures prescribed for GP in the literature. Chapter 3 conducts a detailed survey of the fitness measures.

Tournament selection is the most popular selection method used in GP literature [49]. Fitness-proportionate selection was also used at the inception of GP [1]. The implementations of tournament and fitness-proportionate selection are the same as in GAs. The literature [49] motivates for the use of tournament selection in GP for the same reasons as in GAs, whereby tournament selection offers the opportunity to control the convergence rate of GP by manipulating the value of the tournament size, k , and small values of k are shown to slow down the convergence of the GP population, mitigating premature convergence [49].

2.3.5 Genetic operators

As in the case with GAs, crossover is the principal genetic operator used in canonical GP, while mutation is also used as a secondary operator [1]. In addition to crossover and mutation, reproduction is an operator that simply copies selected individuals into the next generation. The three operators are discussed in detail in the text below.

Crossover

Given two parents, crossover selects a crossover point (i.e. a node) in each parent tree with random uniform probability. Subsequently, the subtrees rooted at the selected nodes in each parent are swapped between the parents to yield two offspring [1]. Figure 2.10 depicts crossover.

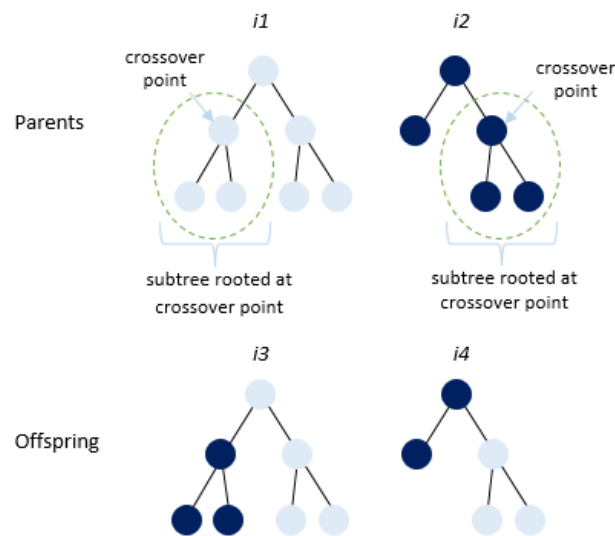


FIGURE 2.10: GP crossover

In practice, the standard GP crossover operator described above exchanges only small subtrees. Typical GP primitive sets lead to trees with an average branching factor of at least two, such that there are more nodes at depths occurring further down the trees [45]. Hence selecting crossover points with random uniform probability will lead to crossover exchanging small subtrees that occur further down along the parent trees; Langdon et. al. [45] argue that crossover can in fact be reduced to simply swapping terminal nodes between the parent trees. To mitigate this effect, the widely adopted approach in the literature is to constrain the selected crossover points to function nodes 90% of the time, and terminal nodes 10% of the time [1, 45].

A number of crossover variants have been defined in the literature [50–53]. Uniform crossover [51] is an interesting variant similar to the uniform crossover operator used in GAs. In GP, uniform crossover between two parent trees requires the identification of a common region shared by the trees. Here, the common region is identified by simultaneously traversing the pair of trees starting at the root nodes to find the contiguous section in which the trees share the same arity in the nodes visited; the common region begins at and includes the root nodes. Uniform crossover proceeds by jointly traversing the parent trees to identify the nodes in this common region. Subsequently, similarly to the GA case, each node in the common region is swapped with the node at the corresponding locus in the other parent tree, with probability p_u . The nodes in the boundary of the common region (i.e. nodes that are leaves of the tree fragment representing the common region) are also identified; terminal nodes in this boundary region are also swapped with probability p_u , while for function nodes in the boundary region, the subtrees rooted at the function nodes are swapped with probability p_u . Figure 2.11 depicts uniform crossover.

In uniform crossover, the nodes and subtrees in the parent trees are treated separately and inherited independently of position, encouraging a more detailed search of the program space. Uniform crossover also

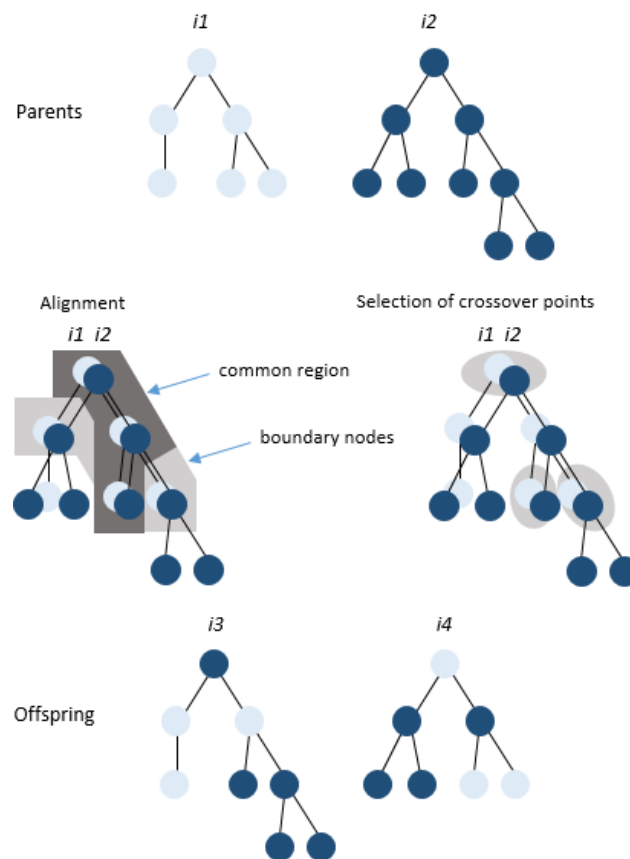


FIGURE 2.11: GP uniform crossover

applies the notion of homology, whereby within common regions, the exchange of homologous primitives can happen much like the same way it does in GAs. One-point crossover [51] is a similar operator that applies the notion of homology. One-point crossover identifies a common region between two parent trees in the same way as uniform crossover. Next a single crossover point within the common region is selected and the subtrees occurring at the point are swapped between the parent trees. Uniform and one-point crossover are however limited in their applicability: for problems with mixed-arity function sets, the chance of identifying a common region shared between two selected parent trees is diminished, precluding crossover. The indication is that the notion of homologous crossover is not easily transferred to GP.

Context-aware crossover is a more recent crossover variant proposed in the literature [53, 54]. In context aware crossover, a subtree is randomly selected from one parent. Next, an exhaustive search is conducted of the best position to place the subtree within the second parent. The exhaustive search produces a pool of offspring, whereby an offspring is generated for each candidate placement of the subtree. Subsequently, the fittest offspring in the pool is selected as the result of crossover. The aim in context-aware crossover is to ensure that the exchanged subtrees are utilized in the best possible contexts in the produced children, rather than the random placement of subtrees done by standard GP. Nevertheless, the exhaustive search on the pool of possible offspring is costly, moreso for problems with expensive fitness computations. Majeed and Ryan [53, 54] argue that this cost can be offset by applying standard crossover at the beginning of a GP run, and restricting the use of context-aware crossover to refining the GP population in later generations. Nevertheless, parameter tuning issues arise as to which is the best generation to introduce context-aware crossover, which may differ for different problems.

Overall, as in the case of GAs, the bulk of GP's search power stems from fitness-based selection combined with crossover [1, 45]. Fitness-based selection ensures that high-performance individuals are repeatedly tested, whereas crossover facilitates the exchange of genetic information between the solutions [1, 45].

Mutation

Two mutation variants commonly used in the literature are subtree mutation [1] and point mutation [55]. Given a parent, subtree mutation selects a random mutation point in the parent tree. Subsequently, the subtree rooted at the mutation point is replaced by a randomly generated subtree. Figure 2.12 depicts subtree mutation.

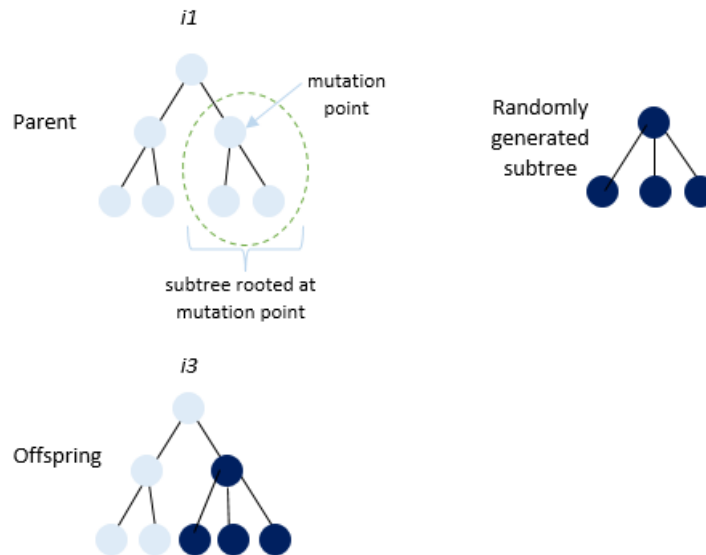


FIGURE 2.12: GP subtree mutation

Point mutation also selects a random mutation point. Subsequently, if the selected point is a function node, the function at the selected node is substituted with another function of the same arity; here, only the node is substituted, and not the entire subtree rooted at the node. In turn, if the selected point is a terminal node, the terminal at the selected point is substituted with another terminal. Figure 2.13 depicts point mutation. Note that point mutation does not produce variation with respect to the size and shape of the GP parse trees. This can be useful with respect to minimizing the increase in size of the parse trees during the course of GP [55].

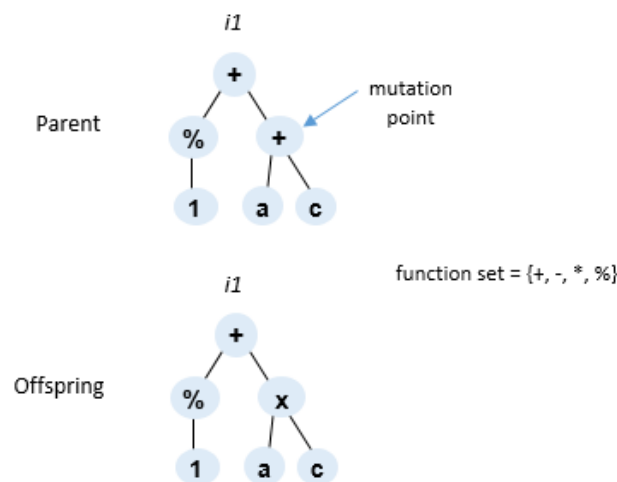


FIGURE 2.13: GP point mutation

As in the case with GAs, GP mutation plays the important role of increasing the population diversity and improving on the probability of escaping local optima [1].

2.3.6 Reproduction

Reproduction is typically applied to produce only a fraction of the next generation. The operator simply copies selected individuals into the next generation. The purpose of the reproduction operator is to make more copies of fitter individuals in subsequent generations [1].

2.3.7 A critical analysis of genetic programming

The goal of computers being able to write their own programs, and hence solve problems autonomously, is central to artificial intelligence and machine learning. Therefore GP plays a pivotal role in these fields. Since inception, GP has been employed to solve many practical problems, both in industry and academia [45].

Nevertheless, like GAs, GP is susceptible to premature convergence on problems prone to local optima [1, 4, 56–58]. McPhee and Hopper [58] report that canonical GP generally exhibits a rapid loss of diversity. Particularly, canonical GP populations are observed to start converging to a common root structure as early as the 16th generation [58]. Furthermore, only a handful of the initial population solutions are shown to contribute genetic material to the final populations [58]. The indication is that canonical GP spends a lot of time permuting a small number of individuals, rather than taking advantage of the diversity present in the initial population [58]. This strong convergence is termed premature convergence when search finds and exploits sub-optimal solutions. Hitchhiking and genetic drift are the likely culprits of premature convergence [58]. Not all loci in the parse tree representation share the same significance; in general, nodes closer to the root of a tree have more descendant nodes within the tree, and as such have more significance with respect to the expression encoded by the tree and its fitness. As search progresses, the genetic material from individuals that contain relatively fit genetic material in significant loci will tend to dominate future GP generations. Hitchhiking occurs when low-fitness genetic material contained in the less significant loci dominates these loci as a result of early spurious associations with the fit genetic material occupying significant loci [58]. In turn, genetic drift refers to the change in the frequency of the primitives in a population due to random sampling [58]. Once the frequency of a primitive is low (e.g. due to the effects of hitchhiking), genetic drift plays the role of further whittling down the occurrences of the primitive, which can lead to the loss of useful primitives [58]. The measures undertaken to mitigate premature convergence in the literature include employing tournament selection with a small tournament size [1], and increasing the mutation rate [58]. Alternatively, different fitness measures have also been proposed to mitigate premature convergence; this discussed in detail in chapter 3 of the thesis.

In another vein, like GAs, GP is also stochastic in nature. Because search incorporates random number generation, each run of GP on a given problem yields a different result. Hence additional runs are required to estimate the average performance of GP on a given problem, which can be costly for computationally expensive problems. GP also requires parameter tuning, whereby like the GA case, the choice of genetic operator probabilities, selection method, population size, as well as the termination criteria specified all impact on the performance of the algorithm. Parameters are tuned empirically, where the practitioner experiments with different configurations and chooses the parameters that produce the best result, or by the use of parameter tuning algorithms. The cost of parameter tuning should also be taken into account when applying GP to solve a given problem.

Despite the above-mentioned challenges, GP offers key advantages over other search optimizers. Like GAs, GP conducts the same executional steps of fitness-based selection and regeneration, regardless of the specific problem being tackled [1]. Hence, GP is a general problem solver, which can be applied on different problems, as long as suitable chromosome representations and fitness measures are defined. GP also offers the advantage of a population-based, rather than point-to-point search, reducing the chance of locating local optima in multi-modal search spaces.

Overall, canonical GA mimics nature more closely than GP, with respect to evolving fixed-length chromosomes and applying homologous crossover between matching genes [3]. Nevertheless, GP offers the opportunity to directly evolve programs of high complexity, without having to define the structure of the program in advance [1]; here, the GP variable-length representation allows the program structure to evolve in problems where it is difficult to a priori specify the structure and size of the optimal solution.

2.4 Summary

This chapter has described GAs and GP in detail. GAs and GP were discussed in terms of the sequence of steps executed by the algorithms, and details provided with respect to the mechanics of the algorithms, including the representation scheme, initial population creation, fitness evaluation and selection, and genetic operators employed in the algorithms. The chapter also presented a critical analysis of GAs and GP. The discussion established that GAs and GP are ubiquitous and well-documented general-purpose optimizers with the capability of solving problems from different application domains.

Chapter 3

A Survey of Fitness Measures in GP

3.1 Introduction

This chapter presents a survey of the fitness measures prescribed for GP. Following an extensive examination of the literature, the fitness measures are categorized based on their *modi operandi*, as follows:

1. *Objective fitness*: These fitness measures assign worth to candidate solution programs based on concurrence with a set of values representing the search objective. This involves evaluating each solution against the complete search objective, on each generation of GP [1].
2. *Divide-and-conquer fitness*: These fitness measures modify objective fitness by applying a divide-and-conquer strategy. The GP practitioner decomposes the complete search objective into useful sub-objectives [59–65]. Alternatively, the fitness measure autonomously decomposes the problem [5, 66]. Subsequently, the “simpler” sub-objectives are optimized with the aim of solving the complete problem [59–65].
3. *Fitness sharing*: These fitness measures modify objective fitness by incorporating a niching strategy, facilitating the investigation of multiple search space optima in parallel [6, 15, 67–70].
4. *Dynamic fitness*: These fitness measures modify objective fitness by actively varying the criteria for fitness evaluation during the course of GP [8–11, 71]. This action prevents the population from settling down and converging on local optima [8–11, 71].
5. *Subjective fitness*: These fitness measures assign worth to candidate solution programs based on competition or collaboration with other solutions [13]. A solution’s subjective fitness is not absolute (or objective), rather it is a function of the specific strategies that compete or collaborate with the solution in the course of fitness determination [12, 13].
6. *Novelty search*: These fitness measures completely ignore the concept of a search objective [4, 72–76]. Instead, candidate solution programs are distinguished based on the extent to which they differ from previous solutions [4, 72–76]. Fundamentally, novelty search ignores the pressure to achieve high objective fitness, rather the selective pressure is for the candidate solution programs to “do something new” [4, 72–76].

The fitness measures that comprise the above-listed categories are surveyed according to the following format:

1. *Motivation*: The specific purpose underlying the fitness measure.
2. *Implementation*: The implementation of the fitness measure.
3. *Variants*: Existing variants of the fitness measure. This item is omitted where the literature does not define variants.
4. *Advantages*: The advantages of the fitness measure.

5. *Disadvantages*: The disadvantages of the fitness measure.
6. *Discussion*: Observations in the literature pertaining to the fitness measure. The observations are discussed in context of the theory underlying the fitness measure. In some cases, recommendations are made towards improving the fitness measure. Finally, deductions are made as to the types of problems to which the fitness measure is suited.

For the sake of uniformity, unless otherwise stated, the fitness measures in this thesis are formulated as maximization functions. However, the discussion is also applicable to minimization. Both maximization and minimization are classes of optimization [1]; for example, switching from maximization to minimization, or vice versa can be achieved by simple negation of the numerical output resulting from the fitness evaluations [1].

In formulating the fitness measures, the terminology “taken from” is used for equations copied directly from the referenced text; however, the notation in these equations may differ from the referenced text, in conformity with the uniform notation applied in the manuscript. The terminology “adapted from” is used for equations that borrow from the referenced text; here, the equations in the referenced text are specific to the problems evaluated in the text; the equations are restated to express a general formulation for the given fitness measure. Lastly, where neither terminology is used, the equation is a formulation derived by the author, based on the description of the given fitness measure in the literature.

3.2 Objective fitness

Objective fitness (OF) measures are the original fitness measures prescribed at the inception of GP [1]. OF measures rate the candidate solution programs in a population based on their distance to the search objective. In objective fitness GP (OF-GP), the candidate solution programs that are closer to achieving the objective are considered to be better than solution programs that are further away.

Motivation

OF is motivated by the definition of fitness in nature [1]. In nature, fitness is associated with the quality (or survivability) of an individual, which determines the probability that the individual reproduces [1]. In GP, a candidate solution program achieves the best possible quality by reaching the search objective [1, 77, 78]. Accordingly, it is intuitive to perceive quality based on proximity to the objective [1]. Drawing a parallel with nature, OF evaluation steers search towards high (objective) quality solution programs and their offspring [1, 77, 78].

Implementation

In quantitative problems, the widely adopted approach in the literature is to measure the OF of a candidate solution program as the sum of the absolute (or squared) difference between the result produced by the solution program and the expected result, over a set of input values [1, 77–84]. In the case of minimization, the lower the sum of the differences, the fitter the solution program. This result can also be inverted for maximization. The combinations of input values and expected results utilized for this purpose are known as fitness cases [1, 77, 78]. A fitness case takes the form (x, y) , whereby x represents one or more program inputs, and y represents the expected result for x [1]. Fundamentally, fitness cases are a sampling of the independent variables or the distinctive initial conditions of a system [1]. Numerous applications of GP employ a set of fitness cases to teach GP how to solve a problem, namely a training set, and a different set of fitness cases to evaluate how well GP has learnt how to solve the problem, namely a test set [1, 45].

Another common implementation of OF applied in quantitative and qualitative problems simply counts the number of times a candidate solution program outputs a value that is within an acceptable tolerance of the

corresponding expected result for the given fitness case; every such output value registers a “hit”. Here, the OF counts the number of hits, such that a higher number of hits means a better solution program [1]. The hit ratio measure described here only rewards perfect (or near perfect) achievement of the evolutionary objectives. Therefore, this fitness measure is not suited to complex applications, such as in the evolutionary robotics domain [63, 85–87], where given a challenging task, it is difficult to find solution programs that achieve the objectives perfectly [63, 85–87]. In such scenarios, applying the hit ratio measure would lead to a lack of fitness gradient information where there is no basis to distinguish between solution programs [85]. This occurrence is termed the bootstrap problem [63, 85–88], where the lack of fitness gradient reduces GP to a random search. In this vein, the sum of the absolute (or squared) difference to the target values may be a more appropriate OF measure for these problems. Alternatively, a divide-and-conquer strategy that facilitates a piece-meal achievement of the objectives can be applied, as discussed in section 3.3.

Equation 3.1 represents OF. Equation 3.1 is adapted from [1].

$$F_1(i, m) = \sum_{n=1}^{|m|} \delta(S(i, x_n), y_n) \quad (3.1)$$

whereby:

- i. $F_1(i, m)$ is i 's OF score.
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem. $|m|$ is the size of m .
- iv. $S(i, x_n)$ is the output value returned by i for the n^{th} fitness case in m .
- v. y_n is the target value for the n^{th} fitness case in m .
- vi. $\delta(S(i, x_n), y_n)$ is a function computing the level of concurrence of $S(i, x_n)$ with y_n .

Possible calculations for $\delta(S(i, x_n), y_n)$:

Absolute difference:

$$\delta(S(i, x_n), y_n) = -|S(i, x_n) - y_n| \quad (\text{here, the negative sign induces maximization}).$$

Hit count:

$$\delta(S(i, x_n), y_n) = \begin{cases} 1, & \text{if } S(i, x_n) = y_n. \\ 0, & \text{if } S(i, x_n) \neq y_n. \end{cases}$$

In the current study, a candidate solution program is said to solve a fitness case, (x_n, y_n) , if the output produced by the program for the fitness case equals or reaches within an acceptable tolerance of the target value, y_n . Note that in some problems, a single fitness case is defined, rather than a set of fitness cases. For instance, in the Royal Tree [89, 90] and Max [91, 92] problems, a solution's OF is measured as the difference between the single numerical value produced as a result of evaluating the solution, and a single numerical expected value.

Advantages

Two advantages have been identified for OF:

- i. *Intuitiveness*: OF presents an intuitive approach to evaluating the quality of solutions [1, 77, 78].
- ii. *Ubiquity*: Owing to their prescription at the origination of the algorithm [1], OF measures are the most prevalent fitness measures applied in the literature. Therefore, GP practitioners benefit from standardized concepts, such as fitness cases and the measurement of distances in objective space, as well as a plethora of examples of the application of OF [1, 44, 77, 79–84, 89–98].

Disadvantages

Six disadvantages have been identified for OF:

- i) *Symmetric view of the fitness cases*: OF functions do not discriminate between “difficult” and “easy” fitness cases. For example, the hit ratio OF measure considers all candidate solution programs that solve the same number of fitness cases to be equally fit, no matter the specific fitness cases solved. Undoubtedly, on problems where some fitness cases are inherently more difficult than others, it would be more advantageous for the candidate solution programs that solve the difficult fitness cases to receive a higher reward.
- ii) *Premature convergence*: Premature convergence is the convergence of the search algorithm to local optima, resulting from the loss of useful population diversity [1, 4, 56–58]. OF influences premature convergence in two ways:
 - a) *Deception*: Deception occurs when solutions that appear promising only have a weak or no relationship to the search objective [1, 4, 56, 99]. By the same token, sometimes improving OF or making progress towards the objective may require taking intermediate steps that may at first seem deleterious with respect to the objective [1, 4, 56, 99]. Lehman and Stanley [4] give the example of a maze navigation problem in which there is an obstacle between the path-finding agent and the goal position. In this scenario, the agent may be required to move away from the goal position in order to circumvent the obstacle. However, OF does not allow the necessary transition [4]. Rather, an OF measure will always show preference for solution programs that minimize the distance to the goal position. Hence OF measures can fail to reward the stepping stones required to ultimately solve the problem, and effectively prune the necessary stepping stones out of the GP population [4, 67, 91, 99].
 - b) *Ignores useful subprograms*: OF does not explicitly merit the useful information contained in the internal structure of solutions [5, 100]. The OF score considers only the final output produced by a solution program, such that useful intermediate output, produced at the subprogram level, can be overlooked [5, 100]. Failure to merit a program’s intermediate results instigates the loss of useful subprograms [5, 66, 100]. Hence premature convergence can occur due to the loss of useful subprogram diversity [5, 66, 100].
- iii) *Bloat*: Bloat is the uncontrolled and limitless increase in the size of the individuals in the GP population [101–108]. Though it is not the only contributing factor [109–112], OF influences bloat [102, 103, 113]. In later generations of GP, search reaches a quasi-stationary state [114]. At this stage of the evolution, the possibility of fitness improvements is minimal, and most genetic operations will lead to fitness losses as a result of disrupting the useful subprograms accumulated by the population individuals during the evolution [114]. In the near-absence of fitness improvements, GP is reduced to a random search for new representations of the best solution found so far [102, 103, 115]. OF-GP perceives different representations of the same OF value to be of equal worth [102]. Unfavorably, the number of larger programs with a given OF value is greater than the number of smaller programs [50, 115]: given a program with a specific OF value, infinitely larger variations of the program can be created by adding neutral code¹ [109]. Therefore, in the quasi-stationary state, a random search of the space of programs with the best OF value will lead to a predominance of long representations [102, 103, 113].

Langdon [102, 103, 113, 115] applies Price’s Covariance and Selection theorem [119] to program length. According to the theorem, if an aspect of the genetic material is positively correlated with fitness, it will be increased in next generation’s population; conversely, if the correlation is negative, it will be reduced [115]. Langdon [102, 103, 113] empirically verifies positive correlation between OF and program length. In practice, the correlation is not due to long solution programs having better than average fitness, rather it is due to relatively short solution programs having a worse than average fitness [102,

¹The growth of neutral code is the principal cause of bloat [109, 116–118]. The term “neutral” describes code that has no effect on the fitness of the program that contains it [103].

- 115]. In the quasi-stationary state, a short solution program produced by removing a large subtree from a parent program is likely to have resulted from useful subprogram disruption [104, 109, 110, 117, 118]. On the other hand, programs retain their OF and increase their survivability by accumulating neutral code [104, 109, 110, 117, 118].
- iv) *Computationally expensive*: Fitness evaluation is the most computationally expensive aspect of GP [11, 120, 121]. OF evaluates the fitness of each candidate solution program on the complete set of fitness cases (or the complete training set of fitness cases where training and test sets are used). Hence the time for evolution, T , is proportional to the product of the population size, N , the number of generations of GP, g , and the size of fitness case set, $|m|$ (i.e. $T \propto (N \times g \times |m|)$) [71, 120–122]. Large fitness case sets therefore require a considerable amount of GP execution time; otherwise researchers would need to implement the evolution on a distributed computing architecture [123]. Ultimately, an expensive fitness measure limits the number of generations over which GP can be applied, curtailing the learning and adaptation required to increase the possibility of reaching a global optimum [120, 124].
 - v) *Overfitting*: In data mining applications, GP is trained on a training set [9, 125, 126]. The result of the evolution is a programmatic model of the underlying data, whereby the relationships in the data are expressed as a functional relationship mapping inputs to outputs [9, 125, 126]. The practical usefulness of the trained model lies in its capability to generalize to unseen data; that is, the model should be able to map inputs it was not trained on to the correct outputs. Overfitting occurs when a model performs well on the training set, but poorly on a test set of fitness cases [127–129]. By evaluating each candidate solution program on the complete training set, OF steers search towards solutions that fit the training data precisely, rather than solutions that achieve good generalization capability [129].
 - vi) *Limited applicability*: In several machine learning domains, the concept of OF fails simply because of the lack of a suitable objective metric of performance [130]. Chiefly, this problem is encountered in evolving competitive strategies [12, 13, 131–137]. In these problems, the only obvious way in which the worth of a candidate solution program can be determined is through competition with other solution programs [13]: this type of fitness evaluation is subjective and not objective [13]. The subjective fitness of a candidate solution program is not absolute, rather it is a function of the specific strategies that compete with the solution program in the course of fitness determination [12, 13]. In this context, true objective accuracy can only be achieved by evaluating each member of the population against every imaginable strategy, to gain significant knowledge about the search space [12]. This approach is computationally infeasible as it would typically require a colossal amount of computation [12].

Discussion

Numerous applications of GP have used an OF measure. The literature shows that OF-GP discovers optimal or near-optimal solutions on a broad spectrum of benchmark [1, 77] and real-world problems [1, 138–141]. Nevertheless, the performance of OF-GP ultimately depends on the structure of the fitness landscape. The fitness landscape is a mapping of candidate solution programs onto fitness values by a fitness measure [142–144]. The fitness landscape can be visualized as a three-dimensional plot; the candidate solution programs are placed on a two-dimensional subspace according to a neighborhood structure; the third dimension of the plot, which may contain peaks and valleys, is the fitness of the solution programs [142–144]. This landscape informs on the correlation of fitness between parent candidate solution programs and their offspring: a weakly correlated landscape is rugged and discontinuous, whereas a strongly correlated landscape is smooth [144]. Conventionally, an OF landscape (whereby an OF measure maps the candidate solution programs onto fitness values) represents the difficulty of the problem being solved [143–147]: finding the global or even a local optimum can be difficult in a rugged OF landscape, whereas it is quite easy to do so in a smooth OF landscape [143–147]. Also, OF-landscapes prone to local optima entrap OF-GP: for example, premature convergence is

seen in the maze [4], artificial ant [4] and 11-multiplexer [6, 67] problems. Fitness plateaus, which are flat areas of the OF landscape in which the neighbouring solutions have the same fitness, are another source of difficulty for OF-GP. Fitness plateaus reduce GP to a random search due to a lack of gradient information [99]: Langdon and Poli [99] deduce that OF-GP encounters difficulty in the artificial ant problem due to an OF landscape that contains several plateaus, peaks and deep valleys.

The above discussion motivates that OF-GP is suited to problems not prone to local optima and fitness plateaus; examples include the simple Boolean 6-multiplexer [1], even-3,4 parity [1] and quartic polynomial regression [1] problems. The ideal OF landscape is smooth, providing GP with consistent gradient information to steadily guide search towards a global optimum.

3.3 Divide-and-conquer fitness

This category is comprised of layered learning (LL) [7, 59–65] and behavioral programming (BP) [5, 66, 100, 148–150].

3.3.1 Layered learning

Layered learning (LL) GP involves a bottom-up, iterative application of evolutionary search to reach a difficult search objective [7, 59–65]. In the initial generations of GP, fitness is evaluated based on a simplified version of the search objective. In subsequent generations, fitness is evaluated based on progressively difficult, intermediate versions of the objective. Finally in the last stages of GP, fitness evaluation is based on the complete search objective [7, 59–65].

Motivation

LL is motivated by the poor scalability of GP as problems increase in complexity [1, 60]. A number of complex problems involve “chicken and egg” dilemmas [60], whereby it is the interaction of several goal behaviors that contributes towards the accomplishment of the given task, rather than the individual behaviors. Winkeler and Manjunath [60] provide the example of evolving a strategy for memory use. In the target strategy [60], access to memory facilitates the existence of state, allowing time-varying responses to time-varied inputs. In this case, a program that simply writes to memory, without being able to read from memory, does not achieve any performance gain. Similarly, a program that simply reads from memory, without being able to write to memory, does not achieve performance gain. In such scenarios, the progress of search becomes unpredictable, and the development of complex behavior is left to chance [60]. To prevent this occurrence, LL tackles a complex problem systematically, by gradually augmenting the problem definition, until the complete problem is specified [60].

In [7] LL is used to mitigate overfitting in the symbolic regression domain. In symbolic regression problems, each candidate solution program is a regression model - a functional relationship between system inputs (i.e. explanatory variables) and outputs (i.e. response variables) [7, 151]. The term “functional complexity” refers to the smoothness of this response surface; i.e. functional complexity is a measure of the complexity of the relationship between the explanatory and response variables [7, 151]. In real-world problems, the training sets used to derive the regression models are typically corrupted by noise, due to lack of accuracy in measuring the data samples [7, 152, 153]. Functional complexity correlates to overfitting, where overfitted models, which fail to generalize to unseen data, match the random noise patterns found in noisy training data, such that their complexity is increased [7]. To prevent GP from overfitting a given problem, Amir et. al. [7] train GP on progressively more functionally complex problems, culminating in the original problem [7]. Amir et. al. [7] also control the functional complexity of the candidate solution programs in each layer of the evolution, increasing the allowed complexity with the progress of GP.

Implementation

In LL, a complex problem is divided into incremental pieces [59–65]. LL begins with a random initial population and a simplified problem definition. Subsequently, each increment augments the problem definition, where the problem is incrementally solved by building on the solution programs from previous increments [7, 60]. Equation 3.2 represents LL.

$$F_2(i, \theta_j) = F_\tau(i, m_{\theta_j}) \quad (3.2)$$

whereby:

- i. $F_2(i, \theta_j)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. θ_j is the current problem definition. θ_j is part of a series of progressively augmented problem definitions $\theta_1 \subset \theta_2 \subset \dots \subset \theta_j \subset \dots \subset \theta_C$, whereby the symbol \subset means ‘is simpler than’, and θ_C is the complete problem definition.
- iv. F_τ is a function that can progressively evaluate $\theta_1 \subset \theta_2 \subset \dots \subset \theta_j \subset \dots \subset \theta_C$; most implementations of LL use an OF measure to evaluate F_τ .
- v. m_{θ_j} is the set of fitness cases used in the current problem definition, θ_j .
- vi. $F_\tau(i, m_{\theta_j})$ is the fitness of i , given F_τ and m_{θ_j} .
For example, if F_τ is an OF measure, $F_\tau(i, m_{\theta_j})$ is calculated as in equation 3.1, with the exception that θ_j dictates the scope of fitness evaluation.

Variants

The division of a complex problem into increments is problem-specific. However the literature describes three approaches that can be applied on a number of problems:

1. *Variance-based LL* [7]: This variant aims to mitigate overfitting in the symbolic regression domain. The work done in [7] is based on the following premise: given a regression model f , f 's functional complexity can be measured by the variance of its output values over the training data [7, 154]. Here, less complex models exhibit lower variance over the training data compared to overfitted, noise-hugging models [7, 154]. In [7], a sequence of training sets, T_1, T_2, \dots, T_n , is generated from the original training set, T , to represent progressively more functionally complex regression problems for the incremental layers of learning. For each T_j in T_1, T_2, \dots, T_n , the input values of the fitness-cases are the same as the original training set, but the output values are different. A function transformation is used to derive each T_j from T : the transformation reduces the functional complexity of the original problem; in [7], the exponential function is used, such that in each T_j , the variance of the output values is reduced, while preserving the shape of the data [7]. The functional complexity of the candidate solution programs is also controlled in each layer of the evolution. The variance of the output values is determined for each candidate solution program, and the programs constrained not to exceed a variance threshold determined by the evolution layer [7]. The variance threshold is increased with the progress of GP to allow more complexity in later layers of the evolution [7].
2. *Partitioning the fitness case set* [65, 155–157]: In this variant, the fitness case set used for fitness evaluation is partitioned into subsets. Fitness is evaluated on a small subset of the fitness cases in the initial generations [155, 156]. Subsequently, each increment considers an additional subset of the fitness cases in the fitness calculation [155, 156].
3. *Partitioning the fitness case vector* [61, 155, 156]: This variant is used on problems with multiple outputs (for example, in the two-by-two bit multiplier problem [156] each solution program outputs four bits). The variant focuses on the different outputs in turn. Here, fitness is evaluated on fewer outputs in the

initial generations [155, 156]. Subsequently, each increment considers an additional output in the fitness calculation [155, 156].

LL can also be classified based on the criteria used for the termination of the increments:

1. *Generation incremental LL* [59, 60]: Each increment is run for a predefined (fixed) number of GP generations.
2. *Performance-based incremental LL* [60, 63]: Each increment is run until some threshold criteria are reached. For example in [60], the increments are terminated when a threshold OF score is reached. In [7], the increments are terminated when overfitting starts to occur: overfitting the initial increments impairs the population with respect to the complete problem [7]. The *performance-based* termination of the increments is based on feedback from the GP algorithm, and may therefore be more useful than running the increments for a fixed number of generations. In [7], *performance* is the main termination condition for the increments; *generation-based* termination is applied as a secondary condition, to prevent long execution times.

Advantages

Four advantages have been identified for LL:

- i) *Exploits modularity to improve search*: A modular problem is a problem in which the global optima are comprised of useful lower-order functions (or modules) that are clearly delineated from the optima [1, 158]. LL's divide-and conquer approach improves on GP's search capability: a complex problem that comprises of identifiable subtasks can be tackled by solving the subtasks [59, 60].
- ii) *Mitigates the bootstrap problem*: LL incorporates rewards for intermediate steps towards the objective. This affords GP an adaptive fitness gradient, such that better candidate solution programs can be distinguished at different stages of the evolutionary process, mitigating the bootstrap problem [63].
- iii) *Mitigates overfitting*: *Variance-based LL* [7] mitigates overfitting by progressively augmenting the functional complexity of the problem definition.
- iv) *Reduces computational expense*: In *partitioning the fitness case set*, the fitness of the candidate solution programs is predominantly evaluated on subsets of the fitness case set [65, 155–157]. Thus, fewer evaluations are required to find solutions for simplified versions of the problem. As a result, it is computationally less expensive to arrive at a given level of performance on the complete problem [60]. Furthermore, the overall computational expense of fitness evaluation is reduced, increasing the speed of GP optimization [60].

Disadvantages

One disadvantage has been identified for LL:

- i) *Configuration difficulty*: LL increases the amount of user intervention required to run GP. Apart from defining a fitness measure, the GP practitioner is mandated to specify a list of simplified subtasks [60, 159]; this involves both defining and ordering the subtasks [159]. Such specifications require extensive problem knowledge, which is generally not available for complex real-world problems [159]. The GP practitioner is also mandated to specify when to shift between increments [86]. LL is particularly sensitive to the duration of the increments. Running too long increments can result in overfitting the initial increments [7, 60]. Conversely, running too short increments curtails necessary learning on the individual increments [60]. Overall, LL's configuration difficulty prevents the method from scaling up well to more complex tasks. Notably, most implementations of LL in the literature involve only two or three increments [60–64].

Discussion

The literature shows that LL improves on the performance of OF-GP on tasks with difficult end-goals [7, 60, 62–64]. For example, Amir et. al. [7] outline a strategy for implementing LL in the symbolic regression domain: this strategy mitigates overfitting and improves on the best solution found by OF-GP [7]. Nevertheless, implementing an LL approach can be challenging: this is because partitioning a problem into increments requires a considerable level of problem-specific knowledge [60, 62–64]. If the GP practitioner chooses to implement LL by *partitioning the fitness case set* or *partitioning the fitness case vector*, arbitrary partitions may not suffice. In this case, the practitioner may have to do several re-runs of LL, to gather problem-specific information on the best way to partition the data. This may not be computationally feasible.

Modular problems permit more intuitive application of LL, whereby the initial increments can be run on the modules comprising the problems [158]. For example, the performance of OF-GP in the Boolean even-6 parity problem is improved by incorporating a preliminary layer of learning on the simpler even-2 parity problem [61]; this works because in the Boolean even- n parity domain, higher-order functions are inherently comprised of the lower-order functions [1, 77]. Intuitive assumptions of modularity have also helped in real-world problems [60, 62–64]. In [60], a target tracking task that attempts to keep a moving object centered in the field of view of a mounted camera is broken down into the simpler subtasks of: 1) tracking the depth of the object, 2) tracking the velocity of the object, and finally 3) tracking both the depth and velocity of the object. Barlow et. al. [62] also apply an intuitive decomposition. In [62] the task is to develop a navigation controller for an unmanned aerial vehicle (UAV). The authors in [62] fragment the given task according to the lower-order objectives delineated in the goal strategy: 1) navigating the UAV to a target position, and 2) circling the UAV around the target position once located. Another example of intuitive problem decomposition is seen in the keep-away soccer domain, where Hsu et. al. [63, 64] borrow from the hierarchical learning strategies of human soccer teams to decompose the problem into two increments: teaching the soccer agents to pass a ball 1) without a defender present, and 2) with a defender present. The intuitive problem decompositions in the mentioned examples are shown to improve on the best solutions found by OF-GP [60, 62–64].

The above discussion motivates that LL is suited to modular problems. Ultimately, it would be useful if some heuristics are derived that can be used to detect modularity, and subsequently recommend LL as a result. For example, Krawiec and Wieloch [158] propose a heuristic to detect modularity in Boolean function synthesis GP. The authors [158] define the term “part quality function”; a part quality function is a subgoal (or module) optimized by the subtrees of candidate solution programs, analogously to the way the complete solution programs optimize an OF measure [158]. The authors [158] also coin the term “monotonicity”: given a problem instance, monotonicity is a measure of the correlation between a given part quality function and the OF over a large sample of candidate solution programs; here, useful subgoals exhibit high monotonicity. In [158], the extent of modularity is measured as the maximum monotonicity exhibited by the subgoals of a problem instance. Without actually running the GP algorithm, the subgoals are sampled from the set of all possible outputs over the fitness cases defined for the problem instance (e.g. given a 4-input Boolean function with $2^4 = 16$ fitness cases, the subgoals are sampled from the $2^{16} = 65536$ possible outputs over the fitness cases), and the monotonicity of each subgoal determined. Krawiec and Wieloch [158] show that the proposed heuristic is a useful measure of the modularity of Boolean problems. The proposed heuristic is however not applicable to problems with continuous (i.e. non-discrete) outputs defined for the fitness cases, where determining the subgoals proves difficult. Overall, more work is required with respect to detecting modularity on different problems. Alternatively, the behavioral programming paradigm, discussed in the ensuing section, proposes fitness measures that autonomously detect and exploit the modularity of a given problem.

3.3.2 Behavioral programming

In a number of recent publications [5, 66, 100, 148–150], Krawiec et. al. propose behavioral programming (BP). In BP, the fitness of a candidate solution program is computed based on the OF of the complete program and the usefulness of its constituent subprograms [5, 66].

Motivation

BP is motivated by the observation that OF ignores the useful information contained in the internal structure of solution programs [5, 66, 100]. In BP, the loss of useful subprograms is mitigated by assessing fitness on both the program and subprogram level. In essence, BP identifies and exploits modularity to improve on the search efficiency of GP [5, 66, 100].

Implementation

BP identifies useful subprograms by recording the intermediate results of OF evaluation. To determine the fitness of a candidate solution program, an OF measure iterates over fitness cases of the form (x_n, y_n) (see section 3.2). For each fitness case, the input, x_n , entered into the program tree [1]. Subsequently the program is executed through traversal [1]: intermediate results incurred at the subprogram level are propagated upward in the tree to yield the program output [1].

BP acquires a program trace for each candidate solution program during the process described above [5, 66]. At the beginning of OF evaluation, BP creates an empty list for each fitness case. Subsequently, as each fitness case is evaluated, the intermediate result produced at each visited subprogram is appended to the list corresponding to the fitness case. Table 3.1 is taken from [66] and represents the schematic of a program trace. Each row in the table lists the intermediate outcomes of the OF evaluation of a single fitness case. The columns in the table correspond to the subprograms that make up the candidate solution program. For example, the entry $S_1(x_1)$ represents the intermediate outcome of the OF evaluation of input x_1 at subprogram S_1 .

TABLE 3.1: Schematic of a program trace in BP

Input	Program trace					Expected output
x_1	$S_1(x_1)$	$S_2(x_1)$	$S_3(x_1)$	y_1
x_2	$S_1(x_2)$	$S_2(x_2)$	$S_3(x_2)$	y_2
...
x_n	$S_1(x_n)$	$S_2(x_n)$	$S_3(x_n)$	y_n
...
x_m	$S_1(x_m)$	$S_2(x_m)$	$S_3(x_m)$	y_m

In BP, fitness is calculated as a function of OF and one or more measures resulting from an analysis of the program trace data. Equation 3.3 represents BP.

$$F_3(i, m) = \alpha(F_1(i, m), T_1(i, m), T_2(i, m), \dots, T_t(i, m)) \quad (3.3)$$

whereby:

- i. $F_3(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem.
- iv. $F_1(i, m)$ is the OF of i , determined according to equation 3.1.
- v. The $T_1(i, m), T_2(i, m), \dots, T_t(i, m)$ are measures resulting from an analysis of i 's trace data; the measures used are specific to the BP variant being implemented, as discussed in the ensuing section.

- vi. $\alpha(\dots)$ is a multi-objective fitness measure that facilitates the simultaneous optimization of $F_1(i, m)$ and the $T_1(i, m)$, $T_2(i, m), \dots, T_t(i, m)$.

Variants

Krawiec et. al. [5, 66, 148, 149] describe two approaches to BP. The approaches are:

1. *Pattern Guided Evolutionary Algorithm (PANGEA)* [5, 66, 149]: This approach transforms a candidate solution program's trace data into a data set input to a machine learning algorithm. The machine learning algorithm is used to derive a model of the program's intermediate execution [5, 66]. The trace data is translated as follows; the last column of table 3.1 plays the role of the label (or dependent variable), and each preceding column is an feature (or independent variable); in addition, each row in the table represents an example [5, 66]. In this manner, the program trace data defines a supervised learning task, where the aim is to find a model that maps the independent variables onto the dependent variable (i.e the expected output) for each example. If the expected output is categorical, the dependent variable contains class labels and the machine learning task is data classification; on the other hand, if the expected output is continuous, the dependent variable is a response variable and the machine learning task is regression. Any inductive learning method can be used to model the trace data [66]: for example in [66], the C4.5 decision tree induction algorithm [160] is used as a machine learning classifier on categorical output.

Given a candidate solution program, i , the aim of using a machine learning algorithm to obtain a model of i 's trace data, $M(i)$, is to determine the extent to which each of i 's subprograms is useful with respect to predicting the expected output. $M(i)$ indirectly reveals a description of i 's intermediate execution. This information is conveyed in two measures; 1) $c(M(i))$ - the complexity of $M(i)$, and 2) $e(M(i))$ - the error produced by $M(i)$ [5, 66]; for example, in the case where decision trees are used to model the program trace data, $c(M(i))$ and $e(M(i))$ are measured as the number of nodes, and the classification error of the resulting decision tree respectively [66]. Importantly, if the subprograms represent meaningful information, the inductive learning method should be able to construct a model of minimal complexity from the trace data [5, 66]; also the learning method should be able to construct a model that yields minimal error over the trace data [5, 66]. In PANGEA, the candidate solution programs with useful subprograms are identified by incorporating $c(M(i))$ and $e(M(i))$ into the fitness measure. A multi-objective fitness measure is employed to simultaneously optimize $F_1(i)$, $c(M(i))$ and $e(M(i))$: $F_1(i)$ is maximized, whereas $c(M(i))$ and $e(M(i))$ are minimized.

PANGEA also incorporates a customized mutation operator [5]. The identified useful subprograms from all candidate solution programs are copied into a global archive maintained throughout the course of GP; given a candidate solution program, the customized mutation operator selects a subtree from the program with uniform probability, and replaces the subtree with a useful subtree from the archive; the most useful subtrees in the archive are prioritized for this replacement [5].

2. *Behavioral Consistency GP* [148]: This approach is based on the concept of equivalence classes: an equivalence class is a group of fitness cases that map onto the same output value; two fitness cases, (x_p, y_p) and (x_q, y_q) , belong to the same equivalence class if $y_p = y_q$. If two inputs, x_p and x_q , are expected to produce the same output, then a subprogram, S_k , that reaches the same state for x_p and x_q is an important component of the program containing S_k , because the program should reach the same state for x_p and x_q at some stage of its execution. Conversely, if the two inputs, x_p and x_q , are expected to produce different outputs, then a subprogram, S_k , that reaches the same state for x_p and x_q has lost the ability to distinguish between the inputs, which should be considered undesired and penalized. Therefore a subprogram, S_k , is deemed useful if the equivalence classes induced at S_k are consistent with the equivalence classes defined in the expected program output [148].

Krawiec and Solar-Lezama [148] define the consistency of a subprogram as the extent to which the subprogram output is the same as the expected output. This is summarized in two rules: 1) Given two fitness cases (x_p, y_p) and (x_q, y_q) , if $y_p = y_q$ in the expected program output, a subprogram, S_k , at which $S_k(x_p) \neq S_k(x_q)$ is penalized, and 2) Given two fitness cases (x_p, y_p) and (x_q, y_q) , if $y_p \neq y_q$ in the expected program output, a subprogram, S_k , at which $S_k(x_p) = S_k(x_q)$ is penalized. Ideally, a candidate solution program would contain a subprogram, S_k , that produces the expected output for all the fitness cases; i.e. $S_k(x_p) = S_k(x_q)$ if and only if $y_p = y_q$ [148].

Krawiec and Solar-Lezama [148] employ the conditional entropy measure² to assess the consistency of a program trace with respect to the expected program output. The authors [148] associate a variable, S_k , with the intermediate outcome of the OF evaluation at the k^{th} subprogram [148]; S_k varies over the fitness cases. In a similar way, a variable, Y , is associated with the expected output over the fitness cases. Two measurements are determined: 1) $H(Y|S_k)$; the amount of information that Y adds to S_k , and 2) $H(S_k|Y)$; the amount of information that S_k adds to Y . Both $H(Y|S_k)$ and $H(S_k|Y)$ attain values of 0 if and only if S_k produces the expected output for all the fitness cases; otherwise, the lower the measures, the higher the consistency of S_k with Y . The behavioral consistency of a candidate solution program, $I(i)$, is defined according to the subprogram at which the sum of $H(Y|S_k)$ and $H(S_k|Y)$ attains its minimum, as shown in equation 3.4, taken from [148].

$$I(i) = \min_k (H(Y|S_k(i)) + H(S_k(i)|Y)) \quad (3.4)$$

whereby:

- i. $I(i)$ is the behavioral consistency of the candidate solution program i .
- ii. k is an index iterating through all the subprograms of i .

Krawiec and Solar-Lezama [148] apply a multi-objective fitness measure to simultaneously optimize $F_1(i)$ and $I(i)$: $F_1(i)$ is maximized, while $I(i)$ is minimized.

Advantages

Three advantages have been identified for BP:

- i) *Exploits modularity to improve search*: BP conveys information about the prospective quality of a candidate solution program. In *PANGEA*, a candidate solution program with low error and low complexity of the model obtained from its trace data comprises of useful intermediate results that can be mapped onto the expected output [5, 149]. The GP algorithm need only put such a program through a small number of transformations (e.g. mutations) to reach a global optimum [5, 149]. GP search is improved by combining OF with the model error and complexity measures to identify such promising solution programs. A similar occurrence is seen in *behavioral consistency GP*: a candidate solution program containing subprograms that have a high level of consistency with the expected output requires few modifications to reach the search objective [148]. *Behavioral consistency GP* facilitates the identification of such solution programs [148]. Ultimately, assessing fitness on both the program and subprogram level mitigates the loss of useful subprograms, which has positive implications for the search efficiency of GP [5, 148, 149].
- ii) *Autonomous optimization criteria in PANGEA*: *PANGEA* invents the additional optimization criteria that it employs [5]. Importantly, the GP practitioner does not have to decide what makes a subprogram useful. Instead, by relying on a machine learning approach to describe the intermediate execution of candidate

²In information theory, the conditional entropy quantifies the amount of additional information needed to describe the outcome of a variable, X , given that the value of another variable, Y , is known [161]. If the probability that $X = x$ is denoted by $p(x)$, then we denote by $p(x|y)$ the conditional probability that $X = x$ given that we already know that $Y = y$. Then the conditional entropy, $H(X|Y)$, is just the Shannon entropy [161] with $p(x|y)$ replacing $p(x)$, averaged over all possible Y ; i.e. $H(X|Y) = - \sum_{x,y} p(x|y) \log p(x|y)p(y)$.

solution programs, the solution programs with useful subprograms are identified without the need for additional human intervention [5].

- iii) *Easy to configure*: BP is easy to configure because it does not incur additional parameters to the ones required for OF-GP [5, 148].

Disadvantages

Two disadvantages have been identified for BP:

- i) *Computational and memory overhead*: The cost of maintaining the program trace data during fitness evaluations increases with an increase in the number of fitness cases specified for the problem as well as an increase in the program sizes. Whereas counter measures to reduce code bloat might contain the program sizes, a modern big data problem with a large number of fitness cases may render the BP approach impractical.
- ii) *Limited applicability of behavioral consistency GP*: *Behavioral consistency GP* [148] is restricted to problems where equivalence classes are induced over the fitness cases. Hence *behavioral consistency GP* will not be effective on problems where each input is mapped onto a different expected output value. The approach is also unlikely to yield benefits over OF-GP in continuous problems, because equivalence is not easily determined over continuous data [148]. This inference is supported by the data in [148], where *behavioral consistency GP* fails to yield a performance advantage on symbolic regression problems.

Discussion

PANGEA is shown to improve on the performance of OF-GP on a number of Boolean, categorical and symbolic regression benchmark problems [5, 66]. *PANGEA* also solves a number of problems that prove unsolvable for OF-GP, including the Boolean even-8 parity and Keijzer-4 regression problems [5]. Krawiec and O'Reilly [149] justify the above observations by showing that *PANGEA*'s consideration of the internal structure of programs provides a better estimate than OF-GP with respect to the distance to the global optima. In this vein, *PANGEA* improves on the search efficiency of GP. Nevertheless, *PANGEA* has its limitations. For example, the approach is shown to perform slightly worse than OF-GP on the Keijzer-1, Keijzer-5 and Keijzer-12 regression problems in [5]; this may be an instance of the No Free Lunch (NFL) theorem, whereby *PANGEA* does not achieve the best result on all problems in the universe of GP problems.

Behavioral consistency GP performs well on problems where useful modules can be detected [148]. For example, the approach outperforms OF-GP on qualitative problems where equivalence classes are induced over the fitness cases [148]. However, as anticipated, the performance of this BP variant is degraded on continuous problems; in [148] *behavioral consistency GP* is shown not to achieve performance gain over OF-GP on a number of symbolic regression problems.

Overall, the above arguments motivate that BP is suited to modular problems, where useful subprograms can be found and exploited. *PANGEA* is more effective than *behavioral consistency GP* on continuous problems. In *PANGEA*, the use of an embedded machine learning algorithm to induce models of the intermediate execution of candidate solution programs removes the need for human-designed heuristics that inform on the usefulness of subprograms. This is better than the human-designed behavioral consistency measure employed in *behavioral consistency GP*, which has limited use on continuous problems.

3.4 Fitness sharing

Fitness sharing (FS) is based on the concept of niching [6, 15, 67–70, 162]. Niching methods promote the existence of distinct stable subpopulations (or niches) within a main population [163]. According to [163],

a niching method must be able to maintain these niches for an infinite time period. The candidate solution programs occupying a niche possess a level of homogeneity, whereas disparate niches are heterogeneous with respect to each other. FS modifies OF to generate selective pressure within, as opposed to across, regions of the search space [6, 15, 67–70, 162].

Motivation

OF-GP systems tend to converge to a single search space optimum, which may or may not be a global optimum (see section 3.2). This situation is undesirable in multi-modal (many peaked) search spaces. FS affords GP the opportunity to investigate distinct OF optima in parallel [6, 67–70]. The approach is motivated by regulated competition in nature [164]. Nature does not converge to a single species; rather competition typically occurs within species as opposed to across species [164]. Drawing a parallel, FS measures inhibit head to head competition between widely disparate regions of the search space [164].

Implementation

FS is set up by modifying OF in such a way that candidate solution programs are penalized based on similar programs occurring within the population [6, 67–70]. The manner in which the similarity of the solution programs is determined depends on the particular variant of FS being implemented, and is detailed in the ensuing section. Penalizing the candidate solution programs on the basis of similarity divides an FS population into distinct limited-size niches. Here, the number of solution programs occupying a niche is limited by the higher penalty imposed on the fitness of programs in densely populated niches [6, 67–70]. On the other hand, the solution programs in newly discovered niches incur a lower penalty (due to sparse population of the niches), compared to solution programs with similar fitness in densely populated niches, favoring the exploitation of new niches [6, 67–70]. Equation 3.5 represents FS. Equation 3.5 is taken from [165].

$$F_4(i, m) = \frac{F_1(i, m)}{\xi(i)} \quad (3.5)$$

whereby:

- i. $F_4(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem.
- iv. $F_1(i, m)$ is the OF of i , determined according to equation 3.1.
- v. $\xi(i)$ is the niche count which measures the number of solution programs that share fitness with i . $\xi(i)$ is calculated by summing a sharing function over all members of the population; here, the sharing function used is specific to the FS variant being implemented.

Variants

The literature proposes two main types of FS: *semantic FS* [6, 67] and *structural FS* [68–70]. *Semantic FS* demarcates niches in semantic space: whereas OF provides a coarse-grained view of the performance of a candidate solution program (see section 3.2), semantic analysis is fine-grained; the semantics of a solution program is the ordered set of outputs produced by the program over the $|m|$ fitness cases input to GP, expressed as a vector, $S(i, x_n) \in \mathbb{R}^{|m|}$ [166]; semantic space refers to the space of all possible such vectors [167]. *Structural FS* demarcates niches in structural space. Structural (genotype) space is the space of encoded representations of the candidate solution programs [168]. In the context of AST GP, the structural space is the space of all possible parse trees, given the function set and terminal set defined for the problem tackled by GP [169]. *Semantic FS* and *structural FS* are discussed below:

1. *Semantic FS* [6, 15, 67]: This variant is applied on problems where a set of fitness cases is defined. In *semantic FS*, the shared fitness of a candidate solution program is calculated by sharing the objective payoff for each fitness case among all the solutions programs in the current population that solve the fitness case, and summing over all fitness cases [6, 15, 67]. Therefore *semantic FS* penalizes solution programs that solve the same fitness cases, such that selective pressure is exerted towards solving unique fitness cases [6, 15, 67]. Equation 3.6 represents *semantic FS*. Equation 3.6 is taken from [170].

$$F_{4a}(i, m) = \sum_{(x_n, y_n) \in T(i)} \frac{1}{|P(x_n, y_n)|} \quad (3.6)$$

whereby:

- i. $F_{4a}(i, m)$ is the fitness assigned to i .
 - ii. i is the candidate solution program.
 - iii. m is the set of fitness cases defined for the given problem.
 - iv. $T(i)$ is the subset of the fitness cases in m solved by i .
 - v. (x_n, y_n) is the n^{th} fitness case in $T(i)$.
 - vi. $P(x_n, y_n)$ is the subset of the solutions in the population that solve (x_n, y_n) .
 $|P(x_n, y_n)|$ is the size of $P(x_n, y_n)$.
2. *Structural FS* [68–70]: In this variant, a niche is defined as a group of parse trees that are within a threshold distance from each other in structural space [68]. This threshold distance is dubbed the niche radius. The shared fitness of a candidate solution program is determined by penalizing its OF value based on the number and similarity of solution programs in the current population occurring within the same structural niche [68]. Therefore *structural FS* penalizes solution programs with similar structures, such that selective pressure is exerted towards discovering different structures. Equations 3.7 and 3.8 represent *structural FS*. Equations 3.7 and 3.8 are taken from [165] and [68, 165] respectively.

$$F_{4b}(i, m) = \frac{F_1(i, m)}{\sum_j S(dist(i, j))} \quad (3.7)$$

whereby:

- i. $F_{4b}(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. $F_1(i, m)$ is the OF score of i .
- iv. $S(dist(i, j))$ is a penalty incurred by i , based on the extent of structural similarity between i and the j^{th} solution program in the population.
 $\sum_j S(dist(i, j))$ is the sum of the $S(dist(i, j))$ over all $j \neq i$ in the current population.

Equation 3.8 details the calculation of $S(dist(i, j))$.

$$S(dist(i, j)) = \begin{cases} 1 - \frac{dist(i, j)}{\sigma}, & \text{if } dist(i, j) \leq \sigma \\ 0, & \text{if } dist(i, j) > \sigma \end{cases} \quad (3.8)$$

whereby:

- i. $dist(i, j)$ is the pairwise structural distance between i and j .
 $dist(i, j)$ is computed by simultaneously traversing the program trees of i and j to determine the extent to which the nodes in the trees differ: e.g. the trees may contain different nodes at matching positions, or one tree may have fewer nodes than the other. Essentially, $dist(i, j)$ is the weighted sum of the penalties incurred as a result of the differences between i and j at matching positions.
- ii. σ is the user-specified structural niche radius.

Advantages

Three advantages have been identified for FS:

- i) *Mitigates premature convergence*: The maintenance of disparate niches in FS translates to the maintenance of population diversity. Going by the theory that diversity mitigates premature convergence [1, 56, 58], FS mitigates this occurrence [6]. *Semantic FS* promotes variety in the fitness cases solved by the candidate solution programs in the population, curtailing rapid convergence to solution programs that solve only the “easy” fitness cases [6]. In turn, *structural FS* inhibits structural convergence of the entire population [68, 69].
- ii) *Appropriate fitness measure for multi-modal search*: A GP practitioner may want to discover one or more of the global and/or local optima in a given search space. FS facilitates this by investigating multiple search space optima in parallel, such that the practitioner can retrieve the best solution programs pertaining to different optima.
- iii) *Mitigates bloat*: *Semantic FS* is anticipated to mitigate bloat: by penalizing candidate solution programs that solve the same fitness cases, *semantic FS* assigns low fitness scores to offspring that solve the same fitness cases as their parents; a number of these offspring result from genetic operations in neutral code regions, such that preventing the spread of such solution programs curtails the spread of neutral code and corresponding increases in code size. The data in [170] supports the premise that *semantic FS* mitigates bloat, although the authors stop short of commenting that: in [170] *semantic FS* is shown to evolve small and fit solution programs on an image segmentation task when compared to other OF measures. Ekárt and Németh [68] argue that *structural FS* mitigates bloat. *Structural FS* favors smaller candidate solution programs because they are less likely to have a number of substructures in common with other programs in the population [68]. In [68], *structural FS* is shown to evolve significantly smaller solution programs on a symbolic regression task when compared to OF-GP.

Disadvantages

Three disadvantages have been identified for FS:

- i) *Limited applicability of semantic FS*: *Semantic FS* demarcates the niches in a population based on a set of fitness cases. However, recall some problems define a single fitness case, rather than a set of fitness cases. Hence *semantic FS* does not conceptually apply to a number of problems.
- ii) *Configuration difficulty of structural FS*: In *structural FS*, setting the user-specified structural niche radius requires a priori knowledge of how far the search space optima are, such that the structural niches can be demarcated based on the optima. However, information about the search space and the distance between the optima is generally not available for real-world optimization problems.
- iii) *High computational cost*: In *structural FS*, evaluating the fitness of a candidate solution program, i , requires calculation of the pairwise structural distances between i and every other solution program in the population. Therefore, on each generation, the cost of fitness evaluation is quadratic. Furthermore, the cost of a pairwise structural distance computation increases with the progress of GP, due to increases in program size [68, 69].

The computational expense of both FS variants also increases with a steady state GP: here, the membership of the population keeps changing as new offspring join the parent population [171]. As a result, the shared fitness of each candidate solution program in the population, which is a function of the membership of the population, needs to be constantly updated. For this reason, most of the FS implementations use a generational control model [6, 15, 67–70, 172].

Discussion

In work done by McKay [6, 15], *semantic FS* is shown to improve on the performance of OF-GP on the benchmark Boolean 6- and 11-multiplexer problems. McKay [6, 15] justifies his results by citing the ability of FS measures to maintain population diversity, mitigating premature convergence. However, FS constantly enforces the maintenance of niches, such that convergence and exploitation of the solution programs occurs within, and not across niches. As a result, widespread convergence of the GP population in later generations is inhibited; yet widespread convergence is often necessary to involve more search points in exploiting the promising solution programs discovered by search [6, 15]. To relax the niching constraints imposed by FS in later generations, McKay [6, 15] prescribes a ramped approach. In the ramped approach, *semantic FS* is applied in the initial 25% of the GP generations. OF is applied in the last 25% of the GP generations. In the intermediate GP generations, fitness is calculated as a linear ramp between *semantic FS* and OF [6, 15]. The ramped approach is shown to evolve higher-quality solution programs on different problems when compared to both *semantic FS* and OF-GP applied individually throughout the course of GP [6, 15]. The implication is that the ramped approach strikes a balance between mitigating premature convergence in the preliminary GP generations, and subsequently allowing widespread convergence and exploitation in later GP generations. Importantly, the results in [6, 15] highlight the fact that different fitness measures may be appropriate for different phases of GP search; hence the case for DFMGP, proposed in the current study. A potential drawback of the ramped approach is that the exploration and exploitation phases may span different GP generations for different problems, such that the arbitrary partitioning of generations (first 25% - *semantic FS*; subsequent 50% - ramped; subsequent 25% - OF) does not achieve consistent results on different problems. Rather than the arbitrary user-defined partitions in the ramped approach, it may be more advantageous to switch between the fitness measures based on feedback from the GP algorithm. DFMGP will differ from the ramped approach by using a higher-level search algorithm to approximate the best fitness measure to use on each generation, based on the feedback from GP.

Ekárt and Németh [69] also propose an adaptive approach for *structural FS*. The adaptive approach is based on the following observation: in *structural FS*, the population diversity is an increasing function of the niche radius [69]: because FS limits the number of candidate solution programs occurring within a niche, augmenting the niche radius increases the selective pressure for evolved solution programs to be further away from each other [69]. Adaptive *structural FS* continually adjusts the niche radius, based on the feedback from search: if a loss of population diversity is encountered, the niche radius is increased in order to restore the diversity; on the other hand, if an increase in fitness is encountered, the niche radius is decreased in order to increase convergence; if both an increase in fitness and a loss of diversity occur, the former overrules the latter. Adaptive *structural FS* is shown to alter the population diversity to relevant levels at the different stages of GP; that is, a large niche radius is seen to enforce high diversity in the preliminary GP generations, and subsequently the niche radius is reduced with the progress of GP, permitting low diversity and exploitation in later generations [69]. The authors in [69] stop short of comparing the quality of the solution programs evolved by adaptive *structural FS* with those evolved by *structural FS* and OF-GP.

Overall, FS is suited to multi-modal search, where the fitness measure can be used to mitigate premature convergence. Relaxing the niching constraints imposed by FS in later GP generations can help to improve on exploitation once good points in the search space have been discovered.

3.5 Dynamic fitness

This category is comprised of dynamic subset (DS) measures [10, 11, 71, 120, 173] and stepwise adaptation of weights (SAW) [8, 9].

3.5.1 Dynamic subset measures

Dynamic subset (DS) measures are intended for use on problems where a set of fitness cases is defined. DS measures modify OF by presenting GP with different subsets of the fitness case set for fitness evaluation at different points in the algorithm [10, 11, 71, 120]. The fitness case subsets are dynamically adapted (or selected) from the complete set of fitness cases with the progress of GP.

Motivation

DS measures present the problem of solving all fitness cases in the fitness case set as a series of distinct sub-problems, throughout the course of GP [10, 11]. Thus the target of evolution keeps shifting, such that the GP population is prevented from converging to a single region of the search space [10, 11, 71]. Ultimately, DS measures confer benefits with respect to mitigating premature convergence, bloat, overfitting and computational expense, as detailed in this section.

Implementation

To set up a DS measure, the GP practitioner specifies an algorithm that dictates the adaptation (or selection) of subsets of the fitness case set; here, the algorithm used is specific to the DS variant being implemented, as detailed under the DS variants. Equation 3.9 represents DS measures.

$$F_5(i, \theta_j) = F_\tau(i, m_{\theta_j}) \quad (3.9)$$

whereby:

- i. $F_5(i, \theta_j)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. θ_j is the current problem definition. θ_j changes with the adaptation (or selection) of the fitness case subset used for fitness evaluation, m_{θ_j} . m_{θ_j} is continually adapted (or selected) from the fitness case set with the progress of GP.
- iv. F_τ is a function that can evaluate the fitness of the candidate solutions programs over the θ_j ; most DS measures use an OF measure to evaluate F_τ .
- v. $F_\tau(i, m_{\theta_j})$ is the fitness of i , given F_τ and m_{θ_j} .
For example, if F_τ is an OF measure, $F_\tau(i, m_{\theta_j})$ is calculated as in equation 3.1, with the exception that θ_j limits the scope of fitness evaluation.

Variants

The literature describes a number of DS measures. The DS measures are listed below:

1. *Random Subset Selection (RSS)* [71]: In this variant, a different subset of fitness cases is drawn from the fitness case set on each new generation of GP. The fitness cases that make up a subset are drawn from the fitness case set with uniform probability. A similar variant is *Random Sampling Technique (RST)* described in [129, 174], where a different fixed-size random subset of fitness cases is selected each time a user-specified period of GP generations elapses.
2. *Dynamic subset selection (DSS)* [71]: As in the case with RSS, this variant selects a different subset of fitness cases from the fitness case set on each generation of GP. Here, the fitness cases that make up a subset are selected by conducting two passes through the fitness case set. The first pass applies a weight to each fitness case in proportion to the weighted sum of 1) “difficulty”: the rate at which the candidate solution programs failed to solve the fitness case on the generation it was last selected, and 2) “age”: the number of generations since the fitness case was last selected. The second pass selects fitness cases for the subset; selection is done without replacement, and is based on a probability biased towards highly-weighted

fitness cases [71]. Therefore, *DSS* prioritizes the evaluation of “difficult” fitness cases, which are not easily solved by the candidate solution programs. Focussing on “difficult” fitness cases reduces priority on the evaluation of fitness cases that are easily solved, which in turn prevents search from converging to solution programs that solve only the latter. *DSS* also prioritizes the evaluation of fitness cases that have not been selected for the fitness case subsets for a significant number of generations; this action prevents the fitness cases that are easily solved from being completely ignored in future fitness evaluations, which can lead to the solution programs losing the ability to solve these fitness cases. The overall aim in *DSS* is to reduce priority on the evaluation of fitness cases that are easily solved while ensuring that the candidate solution programs being evolved do not lose the ability to solve these fitness cases [71].

3. *Topology-based selection (TBS)* [11]: As in the case with *DSS* and *RSS*, this variant selects a different subset of fitness cases from the fitness case set on each generation of GP. *TBS* uses the topological relationship between fitness cases to determine the content of each subset [11]. According to Lasarczyk et. al. [11], two fitness cases, (x_p, y_p) and (x_q, y_q) , share a close topological relationship if the candidate solution programs that solve (x_p, y_p) solve (x_q, y_q) , and vice versa where the candidate solution programs that solve (x_q, y_q) solve (x_p, y_p) . Based on the performance of the programs in the population, *TBS* continually adjusts the topological relationships between the fitness cases in the fitness case set with the progress of GP [11]. On each generation of GP, the selected subset is comprised of fitness cases that are dissimilar with respect to the induced topology. This is such that search is prevented from converging to solution programs that solve only similar or related fitness cases [11].
4. *Limited-error fitness (LEF)* [10]: This variant is intended for use on supervised classification problems, where a candidate solution program classifies a fitness case correctly if the program outputs the target class label for the fitness case, otherwise the fitness case is misclassified [1, 10]. In *LEF*, fitness is evaluated based on an ordering over the fitness case set. At the start of a GP run, the fitness case set is shuffled into a random order; in subsequent generations, the fitness case set is ordered to prioritize the evaluation of “difficult” fitness cases; this is described in the ensuing paragraph. The fitness evaluation of a candidate solution program involves traversal of the ordered set of fitness cases, where 1) the fitness cases classified correctly by the program and 2) the fitness cases misclassified by the program, are counted during the traversal. Here, the solution program’s fitness is related to how many of the ordered set of fitness cases it classifies correctly preceding a threshold number of misclassifications. When the threshold is reached, the fitness cases not yet evaluated in the course the traversal are regarded as misclassified. Gathercole and Ross [10] determine the fitness score as the total number of misclassified cases; in the case of maximization, the fitness is measured as the number of cases classified correctly prior to reaching the threshold number of misclassifications.

LEF continually adjusts the threshold number of misclassifications that governs traversal of the ordered set of fitness cases. In so doing, *LEF* continually adjusts the subset of fitness cases used for fitness evaluation [10]. Changes in the misclassification threshold are effected when search shows signs of stagnation; that is, when there is no improvement in the best-of-generation solution program (i.e. the best-of-generation individual - BOGI) for a threshold number of generations [10]. Each time a change is made to the misclassification threshold, the ordered set of fitness cases is put through one pass of a bubble sort algorithm; here, the most frequently misclassified fitness cases in the previous generation are “bubbled” towards the beginning of the ordered set, such that the “difficult” fitness cases are prioritized for fitness evaluation [10]. The changes to the misclassification threshold are determined based on the number of fitness cases in the ordered set not evaluated by the BOGI, as follows. If the BOGI makes fewer misclassifications than the threshold, the given problem is made harder by lowering the threshold to reduce the number of fitness cases in the ordered set included in the fitness evaluations; this action increases the focus of GP on the “difficult” fitness cases, which prevents the population from converging to solution programs that solve only “easy” fitness cases. In turn, if the BOGI exceeds the threshold

before evaluating all the fitness cases, the threshold is increased so that more fitness cases in the ordered set are included in the fitness evaluations; this means that a wider variety of solution programs in the next generation will have good fitness scores, which increases the probability that new programs will be produced that move the GP algorithm away from the sub-optimal state.

Advantages

Three advantages have been identified for DS measures:

- i) *Mitigate premature convergence*: *LEF* alters the fitness evaluation criteria when search stagnation is detected in GP, such that the evolving population is not allowed to converge to sub-optimal solutions [10]. Also, *LEF* and *DSS* force the candidate solution programs to deal with “difficult” fitness cases; this action reduces priority on the evaluation of fitness cases that are easily solved, which in turn prevents search from converging to solution programs that solve only these fitness cases [10, 71].

TBS selects fitness cases that are dissimilar with respect to the induced topology to make up the fitness case subsets used for fitness evaluation; hence search is prevented from converging to solution programs that solve only similar or related fitness cases [11].

Overall, the DS strategy of varying the fitness evaluation criteria during the course of GP prevents the population from settling down and converging on local optima [10, 11, 71].

- ii) *Mitigate overfitting*: DS measures also confer benefits with respect to reducing overfitting on supervised learning tasks [127, 129, 174, 175]. When different subsets of a training set of fitness cases are presented to the evolving population during the course of GP, the solution programs that survive multiple generations are capable of performing well on the different subsets; conversely, brittle solution programs which cater for certain fitness cases only are not guaranteed survival [128, 129]. Surviving candidate solution programs are thus likely to capture the underlying relationships in the training set without overfitting it [128, 129, 174].
- iii) *Reduce computational expense*: *DSS*, *RSS* and *TBS* are formulated for the specific purpose of reducing the computational effort associated with fitness evaluation [11, 71, 120, 121]; here, evaluating only subsets of the complete fitness case set at a time enhances the evolution speed of GP [11, 71, 120, 121].

On the other hand, the processing required to select the fitness case subsets in *DSS*, *RSS* and *TBS* makes a small contribution to the GP runtime, when compared to the cost of fitness evaluation. The fitness evaluation of each candidate solution program involves traversing the program tree for each fitness case, which is exacerbated as program sizes increase with the progress of GP. On the other hand, in *RSS*, the process of selecting the fitness case subset for evaluation is simple random selection. In *DSS*, additional memory is required to keep track of the age and difficulty of each fitness case; processing is also required to calculate the weight of each fitness case on each generation. *TBS* requires additional memory to store the topological relationships between fitness cases. In *TBS*, the topological relationships between fitness cases are represented by a weighted graph, where each edge of the graph denotes the relationship between a pair of fitness cases [11]. Here, processing is required to adjust the weight of each edge on each generation, based on the fitness cases solved by the candidate solution programs. Overall, while there is an increase in the memory requirements, selecting the fitness case subsets in *DSS* and *TBS* involves simple arithmetic, whereby the age, difficulty or topological relationships pertaining to the fitness cases are incremented or decremented based on the feedback from GP; subsequently, the fitness cases are selected based on the weights assigned to these quantities.

Disadvantages

One disadvantage has been identified for DS measures:

- i) *Limited applicability*: DS measures are formulated to work on problems where a set of fitness cases is defined. Hence DS measures do not conceptually apply to problems where a single fitness case is defined, rather than a set of fitness cases.

Discussion

The literature shows that DS measures improve on the performance of OF-GP on applications prone to local optima; examples include the Boolean even-n parity problems [10] and the real-world thyroid data classification problem [71].

The main concern when applying DS measures is a worst case scenario where resampling the fitness case set during the course of GP makes fitness a moving target [176]. Here, changing the fitness case subset can lead to the useful characteristics acquired by candidate solution programs on previous subsets becoming irrelevant with respect to the new fitness evaluation criteria, such that the search is non-progressive. Ideally, the population should retain the useful characteristics acquired from previous fitness case subsets. In this vein, Ross [176] argues that fitness case subsets that adequately represent the complete fitness case set minimize the negative effects on fitness during transitions between the subsets. With the use of sampling, the candidate solution programs with good fitness scores can be distant from the target strategy delineated in the complete fitness case set; this will be acute for unrepresentative subsets. On the other hand, representative subsets ensure congruence between high-fitness candidate solution programs and the target strategy, whereby such programs can be retained during the transitions between the subsets. When using GP to evolve regular expressions, Ross [176] observes that in the case of randomly sampled fitness case subsets, larger subsets are more representative than the complete fitness case set; hence configuring a larger subset size leads to improved algorithm performance [176]. However, larger subsets are associated with more computational effort required to evaluate the fitness of candidate solution programs. Therefore, in [176], configuring the subset size is a trade-off between adequately representing the complete fitness case set and the expense of a fitness computation. This dilemma is addressed by some of the DS variants. *TBS* [11] can be applied to the given problem, whereby the topological relationships between fitness cases are used to discourage similarity between the fitness cases selected for the subsets, such that small subsets can sufficiently represent the complete fitness case set. *DSS* can also be applied, whereby the priority of cases not included in the fitness case subsets for a significant number of generations is increased, such that the evolved candidate solution programs do not lose the ability to solve these fitness cases.

Overall, DS measures are shown to mitigate premature convergence on problems prone to local optima. DS measures are also shown to mitigate overfitting on supervised learning tasks. Furthermore, DS measures can be used to reduce the computational effort of OF evaluation on problems with large fitness case sets.

3.5.2 Stepwise adaptation of weights

Stepwise adaptation of weights (SAW) measures are also intended for use on problems where a set of fitness cases is defined. SAW modifies OF by assigning adaptive weights to the fitness cases in the fitness case set [8, 9]. The adaptive weights are used to increase or decrease the influence of the individual fitness cases on the fitness evaluation.

Motivation

SAW is intended to mitigate premature convergence [8]. By employing adaptive weights, SAW effectively varies the fitness evaluation criteria during the course of GP. Importantly, SAW increases the weights applied to “difficult” fitness cases, which are not easily solved. This action increases the priority of the “difficult” fitness cases, while reducing the priority of fitness cases that are easily solved. As a result, search is prevented from converging to solution programs that solve only the latter class of fitness cases [8, 9].

Implementation

To set up SAW, the GP practitioner assigns equal weights to the fitness cases in the initial generation [8, 9]. Subsequently, SAW periodically adapts the fitness case weights at an interval of λ generations, where λ is a user-specified parameter. Here, SAW increases the weights of the fitness cases not solved by the current best-of-generation solution program [8, 9]. Equation 3.10 represents SAW. Equation 3.10 is adapted from [8, 9].

$$F_6(i, m) = \sum_{n=1}^{|m|} W_n \times \delta(S(i, x_n), y_n) \quad (3.10)$$

whereby:

- i. $F_6(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem. $|m|$ is the size of m .
- iv. $S(i, x_n)$ is the output value returned by i for the n^{th} fitness case in m .
- v. y_n is the target value for the n^{th} fitness case in m .
- vi. W_n is the adaptive weight applied to the n^{th} fitness case in m .
- vii. $\delta(S(i, x_n), y_n)$ is a function computing the level of concurrence of $S(i, x_n)$ with y_n .

As in equation 3.1, the possible calculations for $\delta(S(i, x_n), y_n)$ are:

Absolute difference:

$$\delta(S(i, x_n), y_n) = -|S(i, x_n) - y_n| \text{ (the negative sign induces maximization).}$$

Hit count:

$$\delta(S(i, x_n), y_n) = \begin{cases} 1, & \text{if } S(i, x_n) = y_n. \\ 0, & \text{if } S(i, x_n) \neq y_n. \end{cases}$$

Variants

The literature defines two versions of SAW; classic SAW (CSAW) [8, 9] and precision SAW (PSAW) [8]. Both CSAW and PSAW initialize the weights of all cases in the fitness case set to the numerical value of 1. Also, both variants adapt the weights of the fitness cases on each generation, based on feedback from the current best-of-generation solution program. The SAW variants are discussed below:

1. *Classic SAW (CSAW)* [8, 9]: CSAW increases the weights of the unsolved fitness cases by a constant factor (for example 1).
2. *Precision SAW (PSAW)* [8]: For each unsolved fitness case, x_n , PSAW increases the weight of x_n by a factor of $|S(i, x_n) - y_n|$, where $S(i, x_n)$ and y_n are the output produced and the target value for x_n respectively. Therefore, PSAW increases the weight of a fitness case in proportion to the extent of error (i.e. the margin between $S(i, x_n)$ and y_n) on the fitness case.

PSAW is applied on quantitative problems, where the extent of error on a fitness cases can be obtained. GP benchmark symbolic regression problems [1] are classic examples of quantitative problems.

Advantages

Two advantages have been identified for SAW:

- i) *Mitigates premature convergence*: Like DSS and LEF, SAW forces the candidate solution programs to deal with “difficult” fitness cases; this action reduces the priority of fitness cases that are easily solved, which in turn prevents search from converging to solution programs that solve only these fitness cases [8, 9].

- ii) *Mitigates overfitting*: As in the case with DSS, SAW has the capability to mitigate overfitting on supervised learning tasks. Given a training set defined for the problem tackled by GP, SAW continually adjusts priority of fitness cases in the training set with the progress of GP, such that solution programs that survive multiple generations are capable of performing well on different subsets of the training set. Surviving solution programs are thus likely to capture the underlying relationships in the training set without overfitting it.

Disadvantages

One disadvantage has been identified for SAW:

- i) *Limited applicability*: SAW is formulated to work on problems where a set of fitness cases is defined. Hence SAW does not conceptually apply to problems where a single fitness case is defined, rather than a set of fitness cases.

Discussion

The literature [8, 9] shows that SAW can improve on the performance of OF-GP. In [8], PSAW produces better results than OF-GP in the symbolic regression domain. Furthermore, PSAW consistently produces better results than CSAW on symbolic regression problems [8]. PSAW increases the priority of each fitness case in proportion to the extent of error on the fitness case. Hence in PSAW the fitness cases are prioritized in proportion to their “difficulty”. Therefore, the performance advantage of PSAW over CSAW in [8] is justified by the fact that PSAW imposes finer granularity with respect to how the fitness cases are prioritized, whereby the more difficult fitness cases are assigned heavier weight in the fitness evaluations; this action leads to more difficult fitness evaluation criteria, which mitigates convergence of the population even further.

CSAW generally improves on the performance of OF-GP when solving classification problems using supervised learning [9]. Nevertheless, CSAW is seen to perform poorly on the heart disease classification task, due to the way in which the equivalence classes are defined in the task [9]. From section 3.3.2, recall that an equivalence class is a group of fitness cases that map onto the same output value; two fitness cases, (x_p, y_p) and (x_q, y_q) , belong to the same equivalence class if $y_p = y_q$. Two equivalence classes are specified in the heart disease classification task: “heart disease absent” and “heart disease present” [9]. In such scenarios involving highly sensitive data, a false diagnosis of “heart disease absent” is worse than a false diagnosis of “heart disease present”; it is safer to say that everyone has a heart disease [9]. These types of classification problems incorporate a cost matrix that biases the measurement of misclassifications. In this example, the incorrect assignment of a fitness case to the “heart disease absent” class results in a higher penalty. In such scenarios, SAW is not programmed to adhere to the different penalties associated with the fitness cases from the different equivalence classes [9]: SAW increases the weights of all misclassified cases equally, regardless of the contribution of the specific misclassifications to the fitness value. The result is that SAW can mislead search by prioritizing the fitness cases from a low-penalty equivalence class [9]. Rather, it is preferable for the weight on a fitness case to be increased in proportion to the type of error made on the case, where the fitness cases from low-penalty equivalence classes are assigned less weight than the fitness cases from high-penalty equivalence classes. This a worthwhile consideration for extending the function of SAW in future.

Overall, SAW can mitigate premature convergence on problems prone to local optima. SAW can also be applied to mitigate overfitting on supervised learning tasks. However, more work is needed with respect to tailoring SAW to work on problems where biased cost matrices are used to evaluate the fitness of candidate solution programs.

3.6 Subjective fitness

This category is comprised of competitive fitness (CF) [13, 177–181] and cooperative fitness (CoopF) [182–185].

3.6.1 Competitive fitness

Competitive fitness (CF) is a long standing paradigm in machine learning [137, 186–189]. In competitive fitness GP (CF-GP), candidate solution programs are optimized solely on the basis of competition with coevolving solution programs [1, 12, 190, 191]. Here what is required is not necessarily a definition of the search objective, but rather some concept of “better” [12].

Motivation

CF is motivated by the limited applicability of OF-GP. Particularly, CF is formulated for evolving competitive strategies [12, 13, 132]: OF does not conceptually apply to this class of problems, due to the lack of a suitable objective metric for measuring performance [12, 13]. Panait and Luke [13] give the example of using GP to evolve a soccer playing program, whereby it is difficult to determine how the fitness of the candidate solution programs will be assessed a priori. In this scenario, the only obvious way in which the fitness of a candidate solution program can be determined is through competition with other solution programs [13]. If a suitable “expert” strategy is available, the candidate solution programs can be measured against the strategy, however in this case, the evolved solution programs would only be optimal with respect to this “expert”, rather than being generally optimal with respect to a variety of opponents.

CF is also implemented on problems where there is a known OF measure [13, 177–181]. Here, a CF measure replaces an OF measure, conferring advantages with respect to mitigating premature convergence, as detailed in this section.

Implementation

In CF-GP, a fitness assessment topology is employed to facilitate fitness evaluation: a fitness assessment topology details the fitness evaluation context for each candidate solution program, by specifying the nature and number of competitions involved in a fitness computation [13]. The fitness assessment topologies specified in the literature are detailed under the CF-GP variants. Equation 3.11 represents CF.

$$F_7(i, m) = \varphi(i, \mu_0, \dots, \mu_{n-1}, m) \quad (3.11)$$

whereby:

- i. $F_7(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem.
- iv. $\varphi(\dots)$ is a function that represents competition between i and n coevolving candidate solutions, $(\mu_0, \dots, \mu_{n-1})$, over the m fitness cases. The nature of $\varphi(\dots)$ and the value of n depend on the implemented fitness assessment topology.

In CF-GP, the basis of competition is problem-specific. In game playing domains, a solution program, A , is considered to be “better” than a solution program, B , if A outplays B , or wins the game [13, 132–134, 136, 191, 192]. Conversely, when implementing CF-GP on the intertwined spirals problem, Juillé and Pollack [177] perceive “better” to be based on the extent to which a candidate solution program can solve the fitness cases not solved by its opponent; here, the fittest candidate solutions are the ones that solve the largest number of unique fitness cases, relative to the coevolving solutions [177]. According to Juillé and Pollack [177], the latter approach implicitly incorporates a novelty search (see section 3.7); that is, the competing solutions programs are constantly encouraged to solve unique fitness cases or “do something new”; this approach to measuring the CF of candidate solution programs also has the advantage of being universally applicable on optimization problems where a set of fitness cases is defined.

Variants

Two main approaches to CF-GP are described in the literature:

1. *Single-population CF-GP* [12, 13, 132–134, 136, 177, 192–196]: In this approach, a single homogeneous population is evolved. On each generation, the fitness of a candidate solution program is a function of competition with solution programs from the same population. *Single-population CF-GP* is restricted to single-population domains.
2. *Coevolution (Multi-population CF-GP)* [1, 178–181, 190, 191]: This approach involves the coupled evolution of two or more heterogeneous populations [1, 190, 191]. The evolutionary cycles in the populations coincide, and the fitness of a candidate solution program is a function of competition with solution programs drawn from the coevolving population(s) [1, 190, 191]. The genetic operators are applied to the opposing populations independently, such that the populations are genetically distinct, and communicate strictly to determine fitness [1, 190, 191]. *Coevolution* can also be implemented by applying the *host-parasite* model proposed by Hillis [188] to single-population domains: here a population of candidate solution programs (the host population) coevolves with a population of different subsets of the fitness case set (the parasite population) [188]. In each competition between a host, A , and a parasite, B , A is rewarded based on the number of fitness cases (within B) solved; conversely, B is rewarded based on the rate at which A fails to solve all of its fitness cases. This type of rating on the parasite population favors an adaptation towards “difficult” fitness cases [188]. This action prevents search from converging to solution programs that solve only “easy” fitness cases [188].

Both the above-mentioned approaches employ a fitness assessment topology to determine the CF. The topologies described in the literature are listed below. In the listing below, a population size of N is assumed for *single-population CF-GP*. In the case of *coevolution*, *2-population coevolution* is assumed, where the coevolving populations have the same population size of N . The topologies are:

1. *Single elimination tournament (SET)* [12, 13]: This topology is applied in *single-population CF-GP*. On each generation, SET involves the entire current population in a binary knock-out tournament of $\log N$ (base 2) levels. The fitness of a candidate solution program is the maximum level attained by the program in the tournament. Thus the overall tournament winner is assigned a fitness value equal to the maximum tournament level ($= \log N$ (base 2)) [13]. The total number of competitions required per generation is $N/2 + N/4 + \dots + 2 + 1 = N - 1$ [13]. Therefore, the computational complexity of this topology is linear, as in the case with OF evaluation [13].
2. *Round robin* [1, 13]: This topology is applied in both *single-population CF-GP* and *coevolution*. In the single population case, the fitness of a candidate solution program is a function of competition with every other solution program in the same generation of the current population. In *2-population coevolution*, the fitness of a candidate solution program is a function of competition with every solution program in the same generation of the coevolving population. *Round robin* provides the most thorough measurement of the fitness of the candidate solution programs in a population with respect to each other [13]. Nevertheless, the topology is computationally expensive, requiring order N^2 competitions per generation [13]. The quadratic complexity renders the topology unsuitable for problems where competitions are time consuming and require significant computational resources [132, 133]. *Round robin* is seldom applied in the literature.
3. *Random pairing* [13, 132, 133, 190, 194]: This topology is applied in both *single-population CF-GP* and *coevolution*. In the single population case, the fitness of a candidate solution program is a function of competition with another solution program sampled with uniform probability from the same generation of the current population. In *2-population coevolution*, the fitness of a candidate solution program is a

function of competition with a single solution program sampled with uniform probability from the same generation of the coevolving population. *Random pairing* is computationally favorable, requiring order N competitions per generation [13]. The minimal computation renders the topology suitable to problems with computationally expensive competitions [132, 133]. The main disadvantage of *random pairing* is that the assigned fitness value is highly dependent on the luck of having a poor opponent, rather than being dependent on the actual ability of the candidate solution program [13]. The topology is therefore not suited to “noisy” problems, where there is lack of accuracy in measuring the scores resulting from competitions between solutions [13]; CF evaluation on such problems should be based on competition with a sizeable number of opponents, in order to average out the effects of noise [13].

4. *K-random opponents* [13, 179]: This topology is applied in both *single-population CF-GP* and *coevolution*. In the single population case, the fitness of a candidate solution program is a function of competition with K distinct solution programs sampled with uniform probability from the same generation of the current population. In *2-population coevolution*, the fitness of a candidate solution program is a function of competition with K distinct solution programs sampled with uniform probability from the same generation of the coevolving population [179]. The *K-random opponents* topology conducts order $N \times K$ competitions on each generation [13]. *K-random opponents* straddles the *random pairing* and *round robin* topologies; when the value of K is set to 1, the approach is identical to *random pairing*; conversely, when the value of K is set to $N - 1$, the approach is similar to *round robin* [13].

Advantages

Three advantages have been identified for CF:

- i) *Appropriate fitness measure for evolving competitive strategies*: CF measures are appropriate for evolving competitive strategies. This class of problems lacks a suitable metric for OF evaluation [130].
- ii) *Mitigates premature convergence*: CF systems are open-ended: since competitors evolve simultaneously, they continually present new challenges for each other [130]. This evolutionary scheme enforces an “arms race”, whereby innovation in a candidate solution program begets innovation in the competing solution programs [197]. This dynamism mitigates premature convergence [137, 197].
- iii) *Mitigates the bootstrap problem*: CF provides a “hittable” target in the *initial generations* of GP [130]. This is because low-fitness candidate solution programs are assigned fitness based on competition with other low-fitness solution programs. Watson and Pollack [130] illustrate this advantage by drawing a parallel with a real-life chess tournament: if any two novice chess players play against a Grand Master, both novice players will lose, such that their individual performances are indistinguishable. On the contrary, if the two chess players play against each other, the relative performance of the chess players will be evident. In similar fashion, competition between low-fitness candidate solution programs in the initial population allows the GP algorithm to distinguish the differences in performance; this is preferable to all the solution programs being measured against a difficult objective fitness measure.

Disadvantages

Three disadvantages have been identified for CF:

- i) *Relativism*: Watson and Pollack [130] argue that CF-GP is vulnerable to relativism. Given that the candidate solution programs are the sole source of information with respect to each other’s performance, there are no guarantees that evolution will proceed in a direction that maximizes the absolute fitness of the solution programs [130]. For example, in cases where the genetic operators are equally likely to increase or decrease the absolute fitness of the candidate solution programs, two high-fitness solution programs

may register the same score in competition with each other as two low-fitness solution programs [130]. In such scenarios, the CF-GP is non-progressive.

- ii) *Focusing on the wrong things*: Competing candidate solution programs focus on exploiting each other's weaknesses [130]. This may lead to solution programs that specialize in the weaknesses of their opponents, rather than general solutions to the task [130]. For example, in the predator-prey pursuit problem [190], if the fitness of either population is limited by the choice of function and terminal set, the coevolving population is trained to defeat the limited population, as opposed to being trained to defeat a general predator or prey strategy.
- iii) *Potential for loss of gradient information in coevolutionary systems*: A *coevolutionary* system loses its adaptive gradient in scenarios where one population becomes too good, compared to the coevolving population(s) [130]. This is a likely scenario, because the coevolving populations do not share genetic material. The loss of gradient once again results in an "unhittable" target [130]. The coevolving populations no longer exert the mutual pressure to outperform each other, and are thus likely to drift with no improvement [130]. For example, in the predator-prey pursuit problem, Haynes and Sen [190] argue that better function and terminal sets are required in the predators' syntax, in order to evolve a strategy capable of tackling the highly evasive algorithm developed by the prey population [190].

Discussion

CF-GP is shown to evolve effective competitive strategies on different problems; examples include evolving soccer softbot teams [132], backgammon players [192] and chess players [134, 198].

CF-GP is also shown to outperform OF-GP on problems where there is a known OF measure [177–181]. In [177], single-population CF-GP produces better results than OF-GP on the intertwined spirals problem: this problem contains a number of local optima [144, 199]. In [179] and [180], *host-parasite coevolution* achieves better classification accuracy than OF-GP when solving real-world classification problems using supervised learning. Also, in [178, 181], *host-parasite coevolution* achieves a higher success rate than OF-GP on a symbolic regression problem prone to local optima. CF-GP's performance advantage is justified by the fact that CF measures induce a competitive selective pressure to constantly innovate, mitigating premature convergence [178, 180, 181]. Also, Pagie and Hogeweg [181] argue that by focussing on "difficult" fitness cases, *host-parasite coevolution* imposes more efficient sampling of a given fitness case set, compared to OF-GP, which wastes evaluation effort on fitness cases that have already been solved. Focusing on "difficult" fitness cases also reduces the priority of fitness cases that are easily solved, which in turn prevents search from converging to solution programs that solve only the latter class of fitness cases.

All the same, CF-GP has its shortcomings. A number of the fitness assessment topologies used in CF-GP rely on a strong transitivity assumption: if A defeats B , and B defeats C , then A must be able to defeat C [13]. If this assumption is not necessarily true, any deductions made about the relative capabilities of candidate solution programs without explicitly comparing the programs is unreliable [13, 130]. Intransitive problems generally create problematic scenarios for CF-GP [130]. An example of intransitivity is seen in the game of chess, where a player's mastery of the game depends on different abilities (e.g. the ability to plan, the ability to anticipate the opponent's next move, etc.), and each player possesses different levels of skill with respect to the different abilities; here, different abilities can be relevant to different opponents, such that a scenario results in which a player A reliably beats B and B reliably beats C , yet A cannot beat C [130]. In this scenario, if C beats A , a loop occurs, as in the "rock, scissors, paper" game [130]. In evolving competitive strategies, the occurrence of such cyclic superiority relationships among candidate solution programs can lead GP to "chase its own tail", where the evolving population keeps revisiting the same regions of the search space, despite the apparent improvements in fitness [130].

Ultimately, CF-GP is inherently suited to evolving competitive strategies; however, challenges are encountered on intransitive problems, where cyclic superiority relationships exist between the candidate solution

programs. CF-GP can also be used to mitigate premature convergence on problems where there is a known OF measure.

3.6.2 Cooperative fitness

In cooperative fitness (CoopF) GP, candidate solution programs are optimized on the basis of collaboration with coevolving solution programs [182–185, 200].

Motivation

Cooperative Fitness (CoopF) GP uses a “divide-and-conquer” strategy to tackle complex multi-dimensional problems: a multi-dimensional problem is a problem that can be broken down into components, such that each of the components is a combinatorial optimization problem on its own [182–184]. Particularly, CoopF-GP is used on multi-dimensional problems where the problem dimensions interact [182–185, 200]. An example is seen in the multi-robot path planning domain [182], where the task is to compute the paths of various robots on a shared grid, such that each robot has an optimal or near optimal path, and the overall path of all the robots combined is also optimal. Here, generating an optimal path for each robot is a combinatorial optimization problem on its own [182]. Also, coordinating the robots to generate an optimal overall path, where collisions between the robots are minimized, is also a combinatorial optimization problem [182]. In such scenarios where the different dimensions of a problem influence each other, it does not make sense to tackle each dimension in isolation, rather the interaction between the problem dimensions forces them to be tackled simultaneously [182–185, 200].

In CoopF-GP, the search objective is divided into components that represent the dimensions of the problem being tackled [182–185]. Subsequently, two or more distinct heterogeneous populations are coevolved, such that each population optimizes a different component of the objective [182–184]. The coevolving populations interact to determine the fitness of their candidate solution programs. Essentially, CoopF-GP divides a complex problem into smaller subspaces which are collectively searched to solve the complete problem.

Implementation

To set up CoopF-GP, the GP practitioner fragments the problem into components. The practitioner determines the number of components, and the role that each component will play a priori [182–185]. The fragmentation is problem-specific and requires a level of problem domain knowledge [182–184]. Proceeding fragmentation, a random population is initialized for each component. The different populations are evolved in parallel, and their solution programs evaluated in the context of each other [182–184]. Also, the populations are genetically isolated and communicate strictly to determine fitness [182–184]. To obtain an overall solution, CoopF assembles representative members from each of the coevolving populations. At the population level, a solution program’s fitness depends on the fitness of the overall solution formed as a result of collaboration with representatives from the coevolving populations [184]. The fitness of the overall solution is typically based on an OF measure [182–185]. Equation 3.12 represents CoopF.

$$F_8(i, m) = \varphi(i, \mu_0 \dots \mu_{n-1}, m) \quad (3.12)$$

where

- i. $F_8(i, m)$ is the fitness assigned to i .
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem.
- iv. $\varphi(\dots)$ is a function that represents collaboration between i and n representative solution programs, $\mu_0 \dots \mu_{n-1}$, drawn from the coevolving population(s). The nature of $\varphi(\dots)$ and the value of n depend on the configuration specified by the user.

Variants

In CoopF-GP, it is common practice to determine the fitness of each candidate solution program in a given population by collaborating with the best solution programs from the coevolving populations [184]. Alternatively, a fitness assessment topology, as described in section 3.6.1, can be used to specifying the nature and number of collaborations involved in a fitness computation; here, the topologies are implemented the same way as in CF-GP, with the exception that the candidate solution programs collaborate, rather than compete, in the course of fitness determination.

Vanneschi et. al. [184] specify a further taxonomy for two-population CoopF-GP, which can be generalized for a larger number of populations. The authors [184] apply the concept of “turns”: a “turn” is the isolated adaptation of one of the coevolving populations [184]. On a population A ’s turn, the coevolving population is suspended, such the effect of the coevolving population on the fitness of the overall solutions is held constant, and A evolves uninterruptedly for a fixed number of generations [184]. The following strategies dictate the respective turns:

1. *Basic alternation* [184]: In *basic alternation*, each coevolving population executes a turn for a pre-fixed, user-configured number of generations. Population A ’s turn is preceded by population B ’s turn. Subsequently the strategy cycles back to population A ’s turn.
2. *Coevolve if no improvement (CINI)* [184]: In this strategy, each coevolving population executes a turn only when the other population does not yield significant improvement on the fitness of the overall solution for a threshold number of turns.
3. *Auto-alternation* [184]: In this strategy, each coevolving population executes a turn for a pre-fixed, user-configured number of generations. At the end of each turn, *auto-alternation* proceeds by evolving the “most promising” population. Let δ_F represent the improvement of the best-overall-solution fitness at the end of a turn. Then the population that produced the highest value of δ_F in its last turn will have the next turn.

Advantages

Two advantages have been identified for CoopF:

- i) *Improves search on multi-dimensional problems*: In CoopF-GP, employing distinct coevolving populations to tackle a given problem results in more independent optimization of each of the specified dimensions of the problem [182–185]. Particularly, CoopF improves on search by facilitating different step sizes in the different dimensions of a given problem, where the coevolving populations are allowed to execute turns spanning different generations [184]. This is useful in cases where the different dimensions of a problem represent non-homogeneous objectives; for example, a more detailed search may be required in one of the dimensions of a given problem relative to the other dimensions.
- ii) *Mitigates premature convergence*: CoopF confers the advantage of speciation through the maintenance of multiple genetically isolated populations evolving towards a common objective [58, 201]. Diversity is therefore perpetuated, whereby the coevolving populations constantly present new challenges to each other, mitigating premature convergence within the individual populations [201].

Disadvantages

Three disadvantages have been identified for CoopF:

- i) *Configuration difficulty*: Configuring CoopF-GP is non-trivial. GP practitioners are confronted with the credit assignment problem, whereby in addition to defining the different components of the search objective that will be optimized, they need to determine the weight of the contribution made by each component to the fitness of the overall solutions [182].

- ii) *Focusing on the wrong things*: In CoopF-GP, the coevolving populations focus on performing well in relation to the collaborating partner(s). This may lead to solution programs that are specialized with respect to the collaborating partner(s), rather than general solution programs that maximize the absolute fitness of the populations [185].
- iii) *Loss of gradient information*: CoopF systems are vulnerable to a loss of gradient information. This occurs when one population severely dominates the collaborating populations, such that the solution programs from the dominant population contribute disproportionately high scores to the fitness of the overall solutions [185]. As a result, none of the collaborating populations will have sufficient gradient information to facilitate learning [130, 185]. The populations are thus likely to drift with no improvement [130, 185].

Discussion

In general, CoopF demonstrates capability to improve on the performance of OF-GP on multi-dimensional problems [182–184]. According to Wiegand [185], CoopF offers two augmentations to conventional OF evaluation. The first is the division of a problem into smaller subspaces in which to search [185]. The second, facilitated by setting up different optimization systems for the different problem components, is the emphasis of search on the individual components. Overall, CoopF’s “partitioning and focusing” strategy [185] increases exploration of the individual problem components, improving on the search capability of GP. In [182], CoopF is shown to solve the complex problem of multi-robot path planning. Also, in [183], CoopF achieves a high-level of classification accuracy when solving a complex object detection task using supervised learning.

Vanneschi et. al. [184] implement CoopF in the symbolic regression domain. In [184], the task of symbolic regression using GP is augmented to include a subtask of optimizing the values of the numeric terminal constants used in the GP being executed. Hence two levels of optimization are implemented: 1) a GP is used to optimize the given regression function, and 2) a GA is used to optimize the values of the numerical terminal constants used in the GP; thus the evolutionary system is comprised of coevolving GP and GA populations [184]. The authors in [184] experiment with different strategies with respect to the execution of “turns” in the CoopF-GP. The data in [184] indicates that CoopF performs best when execution of the respective turns is informed by feedback from the evolutionary process. Particularly, *CINI* and *auto-alternation* consistently achieve on par or better success rates when compared to OF-GP on the tackled symbolic regression problems [184]. In *CINI*, lack of improvement in the fitness of the overall solutions for a threshold number of generations is used as feedback to break the current turn and allow the coevolving population to execute its turn. In *auto-alternation*, the extent of fitness improvement produced on each population’s turn is used as feedback to select the population that will execute the next turn; this is such that the population with higher capability to improve on the overall OF score can execute consecutive turns. On the other hand, *basic alternation* unconditionally executes alternate turns of the coevolving populations; here, the turn of a population, *A*, which has capability to improve on the overall OF score, can be interrupted, and the fitness landscape changed, such that the same improvements may no longer be achievable on *A*’s next turn [184]. Thus *basic alternation* can be detrimental to coevolutionary search, and is seen to not to produce reliable results in [184].

Overall, CoopF is suited to multi-dimensional problems. CoopF offers the opportunity to tackle the different dimensions of a given problem simultaneously in cases where the problem dimensions interact. CoopF is also used on multi-dimensional problems where the different problem dimensions represent non-homogeneous objectives; here, the use of distinct heterogeneous populations and alternate turns allows for different step-sizes in the different problem dimensions.

3.7 Novelty search

In novelty search GP (NS-GP), candidate solution programs are optimized solely on the basis of how distinctive they are in “behavior”, compared to current and prior solution programs: a behavior implicitly represents

the fitness of an individual, providing a problem-specific, fine-grained perspective of its performance [4, 74]. For example in the maze navigation problem [4], where the search objective is to navigate an agent to a target end-point, the behavior of a solution program is defined as the specific end-point reached when using the program to navigate the agent, rather than the coarse-grained view of measuring performance as the distance between the end-point reached and the target end-point. Fundamentally, NS promotes diversification from current and prior behaviors by generating a constant selective pressure for the evolving solution programs to “do something new” [4, 72–76].

Motivation

NS is motivated by the observation that OF-based systems are vulnerable to deception and premature convergence [4]. NS ignores the pressure to achieve high objective fitness, thus opening up search [4, 72, 73]. Chiefly, the method is inspired by natural evolution’s open-endedness and ability to keep producing novel organisms [4]. The ambitious goal of NS is to replicate nature’s unbounded capacity for innovation [4]. The intuition is that the perpetual search for novelty alone is a potent weapon against premature convergence [4].

Implementation

NS is set up by replacing an OF measure with a novelty metric [4, 72–76]. Use of the novelty metric requires a prior problem-specific definition of behavior [4, 72–76]. Here, the behavior of a candidate solution program is characterized by a behavior vector, $v_1, v_2, \dots, v_b, \dots, v_M$, where the v_b is the value of the b^{th} behavior feature for the given program. The term “behavior space” refers to the space of all possible such vectors. The novelty metric measures a candidate solution program’s distance to its k -nearest neighbors in the behavior space [4, 72–76], where neighboring solution programs are drawn from the current population and a novelty archive. Equation 3.13 represents the novelty metric. Equation 3.13 is taken from [4, 72–76].

$$F_9(i, m, k) = \frac{1}{k} \sum_{j=0}^k \text{dist}(i, j) \quad (3.13)$$

whereby:

- i. $F_9(i)$ is i ’s novelty score.
- ii. i is the candidate solution program.
- iii. m is the set of fitness cases defined for the given problem.
- iv. k is a user-configured parameter.
- v. $\text{dist}(i, j)$ is the distance in behavior between i and j , the j^{th} solution among the i ’s k -nearest neighbors. The measurement of $\text{dist}(i, j)$ is problem-specific. For example, in the maze problem [4], $\text{dist}(i, j)$ is measured as the Euclidean distance between the ending positions of i and j .

A key component of NS is the maintenance of a novelty-archive [4, 72–76]. On the initial generation of GP the archive is empty. Subsequently on each generation, each population member is eligible for incorporation into the archive; a candidate solution program is added to the archive if its behavior distance to the archive exceeds a user-configured novelty threshold parameter, ρ_{\min} [4, 72]. The goal is for the archive to represent the most distinctive behaviors encountered by GP so far; as a result, novelty is perpetuated when the fitness of each candidate solution program in the population is assigned in proportion to the distance to its k -nearest neighbors drawn from both the current population and the archive [4, 72]. The archive can be allowed to increase in size indefinitely [72]. Alternatively, a fixed size can be specified for the archive, where the solution programs added to the archive replace the least novel programs in the archive [72]. The latter strategy is conservative with respect to the computational and memory requirements of GP. Particularly, determining a solution program’s k -nearest neighbors calls for calculation of the pairwise behavior distance to each program in both the current population and the archive, which is computationally expensive for large archives.

Variants

In several optimization problems, NS's complete abandonment of the search objective is too radical an approach [202]. For vast behavior spaces, search guided by novelty alone can run for a long time before finding candidate solution programs that are relevant to the search objective [202]. Therefore, a number of NS variants attempt to minimize the time spent evaluating candidate solution programs with little or no relevance to the objective: this is achieved by incorporating the objective into the novelty search. One such variant is *novelty-fitness aggregation* [72]. In this variant, the fitness measure is a weighted sum of the OF and novelty scores. Other NS variants are applied on different optimization platforms, but also conceptually apply to GP [203–208]. These variants can also improve the NS [204–207]:

1. *Novelty-Based Multi-objectivization* [203, 204]: In this variant, a Pareto-based multi-objective fitness measure is employed, combining the OF and novelty scores. Pareto-based multi-objective measures are useful when optimizing conflicting objectives: a Pareto-based measure is used to discover Pareto-optimal solutions - the set of solutions in which none of the objectives can be improved in value without degrading the other objectives; essentially, all Pareto-optimal solutions are considered to be equally good and cannot be ordered completely [209]. In *novelty-based multi-objectivization*, the Pareto dominance relation facilitates the discovery of solutions with high novelty but low OF scores, and also the discovery of solutions with high OF but low novelty scores [203, 204].
2. *Minimal Criteria NS* [205]: This variant is applied in constrained optimization problems: in constrained optimization, one or more variables are optimized in the presence of some constraints on the variables [205, 207, 208]. Here, solutions that fail to satisfy the constraints are deemed as infeasible [205, 207, 208]. In *minimal Criteria NS*, the formula for fitness evaluation comprises of the novelty metric and a flag indicating whether a solution meets the minimal feasibility criteria for the domain; the problem-specific minimal feasibility criteria are configured a priori. Poor fitness scores are assigned to solutions that fail to meet the feasibility criteria. Otherwise novelty scores are assigned to the solutions as normal.
3. *Two population NS* [207, 208]: This variant is also applied in constrained optimization. In *two population NS*, two distinct populations are maintained, a feasible solution population and an infeasible solution population. The solutions in the feasible population are selected based on the novelty metric; on the other hand, the solutions in the infeasible population are selected based on a OF measure that minimizes constraint violations. Proceeding selection, the genetic operators are applied independently in the different populations to generate new solutions. The new solutions are tested for feasibility and placed in the appropriate populations. Importantly, *two population NS* takes into account the possibility that infeasible solutions can be the ancestors of feasible solutions.

Advantages

Three advantages have been identified for NS:

- i) *Does not rely on the structure of the OF landscape*: Ignoring the search objective means that NS is not influenced by the structure of the OF landscape. Therefore, NS can avoid local optima in multi-modal and deceptive problems. NS is also unlikely to stagnate on problems where the OF landscape contains plateaus.
- ii) *Mitigates premature convergence*: NS explicitly perpetuates the behavior diversity of a GP system. Going by the theory that diversity mitigates premature convergence [1, 56, 58], NS mitigates this occurrence [4].
- iii) *Mitigates bloat*: The data in [4, 210] indicates that NS mitigates bloat. Bloat is maladaptive to NS [4]. Particularly, neutral code protects against behavioral change, a phenomenon that is rewarded in NS. Thus in contrast to OF-GP, the selective pressure in NS perpetually discourages the growth of neutral code [4].

Disadvantages

Five disadvantages have been identified for NS:

i) *Configuration difficulty*: Configuring NS is difficult, because the concept of behavior has not been standardized in the literature. An example is seen on the artificial ant problem, where a candidate solution program controls an agent that collects a number of food pellets located on a trail. Here, Doucette and Heywood [72] perceive a candidate solution program to be novel if the agent controlled by the program consumes the pellets on the trail in a unique sequence. Conversely, for the same problem, Lehman and Stanley [4] perceive novelty in terms of the number of pellets consumed by the agent at several even-spaced time intervals. The results in [4] differ markedly from the results in [72]. In [4] NS outperforms OF-GP, whereas in [72] the reverse occurs with OF-GP outperforming NS.

In general, choosing a behavior function requires prior problem-specific knowledge of the particular behaviors that are relevant to the search objective [202, 211]. This expert knowledge may not be readily available and its use contradicts one of the main goals of automated search: minimizing the level of human involvement in the problem solving process [202, 211].

ii) *Poor scalability*: NS's reliability is influenced by the size of the behavior search space. The approach does not scale well to large or infinite behavior search spaces [202, 205, 212]. If the ratio of behavior configurations that are not relevant to the search objective by far exceeds the ratio of configurations relevant to the objective, the search for novelty alone may take a long time to discover the latter class of configurations [204]. Nevertheless, this drawback is addressed by a number of the NS variants that have been described in this section.

iii) *Poor exploitation capability*: Mouret [204] argues that NS is incapable of exploitation. Exploitation occurs when GP converges on promising candidate solution programs, such that the search is concentrated on the promising regions of the search space [1, 22]. NS inhibits population convergence, because the same level of selective pressure in favor of diversification is maintained, even when near-optimal candidate solution programs are found [204]. Thus NS lacks the ability to concentrate search on promising candidate solution programs.

iv) *Poor performance on trivial optimization problems*: NS steers search away from the initial population [73, 74, 76]. The approach is therefore not suited to problems where the initial population contains partial solutions. Rather, NS is suited to difficult problems, where there is more motivation for behavioral exploration [73, 74, 76]. NS is however justified by the fact that most real-world optimization problems are GP-hard [73, 74, 76].

v) *Computational and memory overhead*: NS incurs a computational overhead because computing the novelty score of each solution program calls for calculation of the pairwise behavior distance to each program in both the current population and the novelty archive, in order to determine the k -nearest neighbors. This computation is expensive for large archives. A memory overhead is also incurred as a result of the additional storage space required to maintain the novelty archive.

Discussion

NS is shown to outperform OF-GP on problems prone to deception and local optima [4, 74, 76]; these include problems from the path-finding [4], supervised classification [74] and symbolic regression domains [76]. For example, NS outperforms OF-GP on the artificial ant problem [4]; Langdon and Poli [99] have shown that on this problem, the OF landscape contains several local optima and fitness plateaus; hence ignoring the search objective proves worthwhile on the problem.

In configuring behavior, it is important to consider that NS-GP does not purposefully traverse the behavior space. Rather, as in canonical GP, new solution programs are generated by genetic operators applied in the

genotype space. The novelty of the behaviors of the resulting offspring is then determined [202]. The relationship between genotype and behavior is complex, and depends on the behavior configuration [202]. In NS-GP, once a genotype is found, other genotypes that map onto the same (or similar) behavior will not be considered novel, and are thus discounted by search [202]. This occurrence is beneficial if the genotypes that are mapped onto the same (or similar) behavior have the same potential with respect to discovering useful novelty that can advance the search [202]. Conversely, if the genotypes have different potential, important genotypes will be overlooked [202]. Therefore, behavior should be specified in such a way that similarity in behavior correlates to similarity in the potential of the genotypes [202]. However, measuring the potential of the genotypes is non-trivial [202]: Kistemaker and Whiteson [202] argue that as a result, it is difficult to find a behavior function that correlates the genotype and behavior spaces. In this vein, perhaps the best way to ensure congruence between fitness assessment and the genetic operations in NS is to apply behavior-based genetic operators, whereby the candidate solution programs are modified in the behavior space. Here, the behavior-based operators and the fitness measure should employ the same characterization of behavior. To the author's knowledge, no work has been done with respect to defining behavior-based genetic operators. Hence this is a potential area for future work.

Overall, a well-configured NS can improve the search capability of GP on problems with poorly formed OF landscapes: that is, deceptive and multi-modal problems, as well as problems with fitness plateaus. By ignoring the search objective, NS avoids the difficulty posed to OF-GP on such problems.

3.8 Summary

This chapter has identified the different fitness measures prescribed for GP. The fitness measures are surveyed in detail, and their motivations, variants, advantages and disadvantages discussed. Based on the theory underlying the fitness measures, inferences are made with respect to the different fitness measures that suit different problems. Nevertheless, the information regarding problem suitability is only useful to a certain extent, because in real-world scenarios, GP practitioners are seldom aware of the properties underlying a newly encountered problem. In this case, it is difficult to know a priori the fitness measures that will suit the given problem.

There are a number of overlapping areas in this survey: in some cases, different fitness measures are identified as being suited to the same types of problems. For this reason, it is useful to know the fitness measures that perform the best in a given scenario. Nevertheless, such deductions are hampered by the lack of comparisons between the different fitness measures in literature: most studies simply compare against an OF measure. This is an area that the GP community needs to look into. The subsequent chapter presents a comparison of the following fitness measures: 1) objective fitness (OF) 2) behavioral programming (BP), 3) fitness sharing (FS), 4) dynamic subset selection (DSS), 5) host-parasite coevolution (HP), and 6) novelty search (NS). The listed fitness measures are representative of the taxonomy of fitness measures presented in this chapter.

Chapter 4

A Comparison of Fitness Measures in GP

4.1 Introduction

Based on the literature surveyed in the previous chapter, the aim of this chapter is to test the hypothesis that different fitness measures are suitable for different problems. This chapter also provides the justification for the research in the thesis. An experiment is conducted to evaluate the effect that different fitness measures have for different problems. In particular, the study looks at how the different fitness measures address each of the limitations they aim to overcome for different problems. Six fitness measures are compared: 1) objective fitness (OF), 2) behavioral programming (BP), 3) fitness sharing (FS), 4) dynamic subset selection (DSS), 5) host-parasite coevolution (HP), and 6) novelty search (NS). The listed fitness measures improve on different aspects of GP: in essence, the fitness measures are fundamentally different in their motivations and modus operandi. The goal of selecting diverse fitness measures is to establish the different types of fitness that are suited to different benchmark problems. The listed fitness measures are also state-of-the-art measures. From the discussions in the previous chapter, OF is the original fitness measure applied at the inception of GP; most GP practitioners have the habit of relying on OF measures. BP is a new paradigm shown to improve on the performance of OF-GP. FS is the state-of-the-art in multi-modal function optimization. DSS mitigates premature convergence and overfitting; DSS also lowers the computational cost associated with fitness evaluation. HP is a long standing paradigm in GP literature, shown to mitigate premature convergence and search stagnation due to fitness plateaus. HP is also a competitive fitness measure: competitive fitness measures are argued to mitigate the bootstrap problem. NS is the state-of-the-art with respect to mitigating premature convergence on highly deceptive problems.

The previous chapter indicated a number of common observations pertaining to the above-listed fitness measures. For example, BP, FS, DSS, HP and NS are all capable of improving on the quality of the solution programs found by OF-GP. Furthermore, FS, DSS, HP and NS are all shown to maintain the diversity of GP, mitigating premature convergence. In addition, both DSS and HP play a role in minimizing the computational cost of GP. Also, both NS and FS are observed to mitigate bloat. This chapter makes an all-round comparison of the fitness measures, whereby the fitness measures are all compared based on selected criteria (solution quality, generalization, etc., the comparison criteria are detailed in section 4.2.2). This is done in order to provide a complete picture of the respective capabilities of the fitness measures: for example, a fitness measure may inadvertently influence some aspect of GP that it was not necessarily designed to improve on or mitigate.

This chapter is organized as follows. Sections 4.2 and 4.3 discuss the methodology and results of the chapter's experiment respectively. Finally, section 4.4 draws conclusions about the respective capabilities of the fitness measures based on the results of the experiment.

4.2 Experimental methodology

This section details the implementation of the fitness measures analyzed in the experiment. The criteria used for the comparison of the fitness measures are also discussed. Subsequently, a benchmark suite is specified for the experiment. Then, the experimental set up is described in detail. Lastly, the technical specifications for the experiment are outlined.

4.2.1 Fitness measures

In this study, an experimental treatment, $\{F_\alpha, p\}$, is the analysis of a given fitness measure, F_α , on a given benchmark problem, p . In each treatment, canonical abstract-syntax-tree GP is implemented, with the exception that F_α is the fitness measure applied. The previous chapter described the fitness measures analyzed in the experiment. Table 4.1 below provides the specifics of the fitness measures needed for the implementation. Table 4.1 also makes reference to the sections in the previous chapter describing each fitness measure.

TABLE 4.1: Fitness measures

Measure	Description	Formula
OF	See section 3.2	$F_1(i, m) = \sum_{n=1}^{ m } \delta(S(i, x_n), y_n) \quad (4.1)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_1(i, m)$ is i's OF score. i is the candidate solution program. m is the fitness case set defined for the problem. m is the size of m. (x_n, y_n) is the n^{th} fitness case in m, with an input value x_n and target value y_n. $S(i, x_n)$ is the output value returned by i for x_n. In quantitative problems: $\delta(S(i, x_n), y_n) = - S(i, x_n) - y_n$. Here, a fitness case, (x_n, y_n), is considered solved if $S(i, x_n) - y_n < 0.01$. <p>In qualitative problems:</p> $\delta(S(i, x_n), y_n) = \begin{cases} 1, & \text{if } S(i, x_n) = y_n. \\ 0, & \text{if } S(i, x_n) \neq y_n. \end{cases}$ <p>Here, a fitness case, (x_n, y_n), is considered solved if $S(i, x_n) = y_n$.</p>
BP	<p>See section 3.3.2.</p> <p>The Pattern-Guided Evolutionary Algorithm (PANGEA)^a variant of BP is implemented. Two versions of PANGEA are analyzed: BP1 is canonical GP, with the exception that equation 4.2 specifies the fitness measure used; BP2 is identical to BP1 with the added functionality of archiving useful subprograms, and employing an archive-supplied mutation operator, as recommended in [5].</p> <p>^aThe Waikato Environment for Knowledge Analysis (WEKA) Java API [213] is used to implement PANGEA. WEKA's implementation of the C4.5 algorithm [160] is used to model the program trace data for qualitative (categorical) problems; in turn WEKA's implementation of non-linear regression is used to model the program trace data for qualitative (numerical) problems.</p>	<p>The equation below is adapted from [66]: in the current study, the terms containing $l(i)$ and $e(i)$ are inverted to maximize the BP score (rather than minimize the score as in [66]).</p> $F_2(i, m) = F_1(i, m) \times \frac{1}{\log_2(l(i) + 1)} \times \frac{ m + 1}{e(i) + 1} \quad (4.2)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_2(i, m)$ is i's BP fitness score. i is the candidate solution program. m is the fitness case set defined for the problem. $F_1(i, m)$ is i's OF score, determined according to equation 4.1. $l(i)$ and $e(i)$ are the complexity and error (respectively) of the model trained on i's program trace.
FS	<p>See section 3.4.</p> <p>Semantic FS is implemented.</p>	$F_3(i, m) = \sum_{(x_n, y_n) \in T(i)} \frac{1}{ P(x_n, y_n) } \quad (4.3)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_3(i, m)$ is i's shared fitness score. i is the candidate solution program. m is the fitness case set defined for the problem. $T(i)$ is the subset of the fitness cases in m solved by i. (x_n, y_n) is the n^{th} fitness case in $T(i)$, with an input value x_n and target value y_n. $P(x_n, y_n)$ is the subset of the solution programs in the population that solve (x_n, y_n). $P(x_n, y_n)$ is the size of $P(x_n, y_n)$.

TABLE 4.1: Fitness measures (contd.)

Measure	Description	Formula
DSS	See section 3.5.1.	$F_4(i, \theta_j) = F_1(i, m_{\theta_j}) \quad (4.4)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_4(i, \theta_j)$ is i's DSS fitness score. i is the candidate solution program. θ_j is the current problem definition given fitness case subset, m_{θ_j}. m_{θ_j} is continually adapted from the complete fitness case set with the progress of GP, as described in section 3.5.1. F_1 is the OF measure defined in equation 4.1. $F_1(i, m_{\theta_j})$ is calculated as in equation 4.1, with the exception that θ_j limits the scope of fitness evaluation. <p>On each generation, the probability that a fitness case (x_n, y_n) is selected for fitness evaluation is determined according to its weight, $W(x_n, y_n)$, calculated in equation 4.5, which is taken from [71]:</p> $W(x_n, y_n) = D(x_n, y_n)^d + A(x_n, y_n)^a \quad (4.5)$ <p>whereby:</p> <ol style="list-style-type: none"> $W(x_n, y_n)$ is the weight of the fitness case. $D(x_n, y_n)$ is the difficulty of the fitness case, exponentiated to a user-specified power, d. $A(x_n, y_n)$ is the age of the fitness case, exponentiated to a user-specified power, a.
HP	See section 3.6.1.	$F_5(i, \theta_j) = F_1(i, m_{\theta_j}) \quad (4.6)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_5(i, \theta_j)$ is i's HP fitness score. i is the candidate solution program. θ_j is the current problem definition given the fitness case subset, m_{θ_j}. m_{θ_j} is coevolved with the GP candidate solution programs, as described in section 3.6.1. F_1 is the OF measure defined in equation 4.1. $F_1(i, m_{\theta_j})$ is calculated as in equation 4.1, with the exception that θ_j limits the scope of fitness evaluation.
NS	<p>See section 3.7.</p> <p>Two versions of NS are implemented: the versions differ with respect to the behavior descriptors used; otherwise, the novelty score is calculated as in equation 4.7.</p> <p>The first version of NS, NS1, applies a universal behavior descriptor based on the accuracy descriptor defined in [74]. The accuracy descriptor in [74] is formulated to work on supervised classification problems, where a candidate solution program solves a given fitness case by producing the target value for the fitness case. In the current study, the idea of solving a fitness case is extended to matching or reaching within an acceptable tolerance of the target value, such that the behavior descriptor can be applied to qualitative and quantitative problems likewise. Given a candidate solution program i, i's behavior is described as an m-length vector, $\{B(i, x_0), B(i, x_1), \dots, B(i, x_{ m -1})\}$, where m is the number of cases in the fitness case set. For each vector element, $B(i, x_n)$: $B(i, x_n) = 1$ if i solves the n^{th} fitness case; otherwise $B(i, x_n) = 0$. In this scenario, i's novelty is based on the extent to which it solves the fitness cases not solved by previous and existing solutions programs. Note that this behavior descriptor is in effect a semantic descriptor, whereby the novelty of the candidate solution programs is measured in semantic space; here, the higher-level concept of behavior has been simplified in order to specify a descriptor that is universal to problems that define fitness cases.</p>	$F_6(i, m, k) = \frac{1}{k} \sum_{j=0}^k \text{dist}(i, j) \quad (4.7)$ <p>whereby:</p> <ol style="list-style-type: none"> $F_6(i)$ is i's novelty score. i is the candidate solution program. m is the fitness case set defined for the given problem k is a user-specified parameter. i's novelty is based on the behavior distance to its k-nearest neighbors. $\text{dist}(i, j)$ is the behavior distance between i and the j^{th} solution among i's k-nearest neighbors. $\text{dist}(i, j)$ is calculated according to equation 4.8. $\text{dist}(i, j) = \frac{1}{M} \sum_{n=1}^M B(i, x_n) - B(j, x_n) \quad (4.8)$ <p>whereby:</p> <ol style="list-style-type: none"> $B(i, x_n)$ and $B(j, x_n)$ are the n^{th} elements of i and j's behavior vectors respectively. The absolute difference, $B(i, x_n) - B(j, x_n)$, is used to measure the distance between $B(i, x_n)$ and $B(j, x_n)$.

TABLE 4.1: Fitness measures (contd.)

Measure	Description	Formula
NS (contd.)	<p>The second version of NS, NS2, is used only in the path-finding domain (see section 4.2.3). Path-finding problems are known to be deceptive [4], and as such can benefit from a novelty search with problem-specific behavior descriptors. NS2 employs a problem-specific behavior descriptor adapted from [4]: a candidate solution program i's behavior is described based on the path-finding scores achieved by i at several evenly-spaced time intervals. NS2 is described in more detail in section 4.2.3.</p> <p>A novelty archive is maintained in both NS1 and NS2 [4]; on each GP generation, each population member has a $p\%$ chance of being added to the archive, where p is a user-specified parameter [4].</p>	

SMAC (the Sequential Model-based Algorithm Configuration) [39], a parameter tuning algorithm, is used to tune the parameters relevant to the experimental treatments. A number of parameter tuning algorithms exist [37, 214]. SMAC was selected for the study because SMAC is ubiquitous, and well documented [39]. Hutter et. al. [39] demonstrate SMAC's superiority to popular tuning algorithms like paramILS [38] and GGA [40]. Also, SMAC can be used to tune continuous, discrete, and categorical parameters simultaneously [39]. For each experimental treatment, SMAC is used to tune the parameters intrinsic to the given fitness measure, and to tune the selection method and the genetic operator probabilities used with the fitness measure. In tuning the selection method, SMAC chooses between fitness proportionate selection and tournament selection, and tunes the tournament size in the case of tournament selection. In tuning the genetic operator probabilities, SMAC adjusts the probability of standard crossover, standard mutation and reproduction, as well as the probability of archive supplied mutation when BP2 is the fitness measure used. The GP selection method and genetic operator probabilities significantly influence the performance of the fitness measures. For example, Lehman and Stanley [4] assert that when applying tournament selection in NS, a small tournament size should be used, which reduces the selective pressure; this is important because in NS, the novelty of the solution programs is based on comparison with the novelty archive, which contains representative solution programs from previous generations of the GP run; as a result, high selective pressure would mean strict favoritism of the solution programs that differ from previous generations, which has the undesirable consequence that what is considered novel keeps changing drastically from one generation to the next. In this regard, fitness-proportionate selection, which exerts a high selective pressure [1] is also incompatible with NS. However fitness-proportionate selection or tournament selection with a large tournament size may be more compatible with a different fitness measure. Also, Luke and Spector [215] assert that the configuration of the genetic operator probabilities significantly influences the performance of GP in general. The goal of employing SMAC is to obtain the optimal configuration for each experimental treatment; this is done in order to ensure a fair comparison between the fitness measures, whereby the parametric configurations of the treatments do not hamper the performance of GP.

Table 4.2 summarizes the SMAC tuned parameters for each treatment. A total of 500 SMAC iterations [39] are run to tune each treatment: in general, the probability that SMAC finds the true optimal parameter configuration, $\theta \in \Theta$ (whereby Θ represents the universe of configurations), approaches 1 as the number of iterations of the algorithm goes to infinity [39]; thus SMAC tuning is a trade-off between time constraints and reaching the optimal parameter configuration: from the author's observations, 500 iterations permit SMAC to discover good quality configurations for the experimental treatments within a reasonable amount of time.

TABLE 4.2: SMAC tuned parameters for the fitness measures

Fitness measure	Control parameters
All fitness measures (OF, BP1, BP2, FS, DSS, HP, NS1, NS2)	General parameters <ol style="list-style-type: none"> 1. Selection method (SMAC chooses between fitness proportionate selection and tournament selection, and tunes the tournament size in the case of tournament selection). 2. Crossover, mutation, and reproduction probabilities.
BP2	Intrinsic parameters (See section 3.3.2) <ol style="list-style-type: none"> 1. Archive-supplied mutation probability. 2. Maximum size of the useful subprogram archive; when this threshold is reached, the new subprograms to be added to the archive overwrite the least useful subprograms in the archive.
DSS	Intrinsic parameters (See section 3.5.1) <ol style="list-style-type: none"> 1. Size of the fitness case subset used for OF evaluation. 2. Age exponent. 3. Difficulty exponent <p>The age and difficulty exponents increase the probability that a fitness case is added to the subset used for OF evaluation, based on the last time the fitness case was selected (i.e. age) and the difficulty of the fitness case respectively.</p>
HP	Intrinsic parameters (See section 3.6.1) <ol style="list-style-type: none"> 1. Size of the fitness case subset used for OF evaluation.
NS1, NS2	Intrinsic parameters (See section 3.7) <ol style="list-style-type: none"> 1. k: the number of nearest neighbors used to calculate the novelty score. 2. Probability that a candidate solution program within a generation is added to the novelty archive. 3. Maximum size of the novelty archive; when this threshold is reached, the new programs to be added to the archive overwrite the least novel programs in the archive.

4.2.2 Criteria for comparison

This section lists the criteria used to compare the performance of the fitness measures. The selected criteria are: 1) solution quality, 2) generalization, 3) population diversity, 4) structural complexity, and 5) time taken to execute the experimental treatments. It is important to note that the choice of fitness measure is not the only factor that influences the listed criteria: factors like the function and terminal set used, population size and the number of GP generations are also important considerations [1, 58, 109, 129, 216]. However, this study focusses on the extent to which the choice of fitness measure influences the listed criteria. Apart from the fitness measure, as well as the SMAC-tuned selection method and genetic operator probabilities, the GP control parameters used are identical for all experimental treatments, as detailed in section 4.2.4. The criteria used for the comparison are discussed below.

Solution quality

Quality refers to the distance to the global optima. In the context of maximization, a high-quality candidate solution program maximizes the raw OF score yielded over the fitness case set. Ultimately, GP aims to maximize the raw OF of the solution programs evolved [1]; thus even when a different fitness measure is used to guide search, the raw OF of the candidate solution programs is still used as an indicator of the progress of GP.

The following is measured for each experimental treatment:

1. The OF score of the best solution program (BS) on each generation of GP: this is referred to as the generational BS-OF score. The OF score of the overall best solution program found by GP is also measured; this is referred to as the final BS-OF score.

The problems tackled in the study include supervised classification, Boolean function synthesis and path-finding problems, which exhibit different ranges in their raw OF scores: for example, raw OF scores in the Boolean even-5 parity problem range from 0 to 32; on the other hand, raw OF scores in the tartarus

problem range from 0 to 10. In order to compare the OF scores achieved on different problems, each raw OF score is divided by the maximum possible raw OF score for the given problem, as shown in equation 4.9: this ensures that all reported BS-OF scores span the interval $[0, 1]$, with higher values indicating higher quality:

$$OF(i, m) = \frac{F_1(i, m)}{F_{1max}} \quad (4.9)$$

whereby:

- i. Given a candidate solution program i , $OF(i, m)$ is i 's adjusted OF score, calculated as its raw OF score ($F_1(i, m)$) divided by the maximum possible raw OF score (F_{1max}).

A number of symbolic regression problems are also tackled in the study. For the symbolic regression problems, quality is measured as the sum of the absolute error over the fitness cases defined for the given problem; the quality scores span the interval $[0, \infty)$, whereby lower scores indicate higher quality. For the sake of uniformity with the classification, Boolean and path-finding problems, the transformation function $1 - (x/(x + 1))$ is applied to each symbolic regression quality score, x . Therefore, all reported quality scores span the interval $[0, 1]$, with higher values indicating higher quality.

2. The success rate of GP. This is the proportion of the total number of GP runs that find a global optimum before the maximum number of generations have been reached [217].

Premature convergence is characterized by the inability of GP to make further gains with respect to OF [1, 57, 58]: the phenomenon is thus identified by a plateauing of the generational BS-OF scores for the remaining generations of GP [1, 57, 58]; conversely, the ability to continually improve on OF scores is attributed to non-occurrence, or mitigation (in the case where plateauing occurs in the other experimental treatments) of the phenomenon. In turn, improvements in the quality of the solutions found by GP, as in the case with BP in [5, 148, 149], are detected by higher success rates and higher final BS-OF scores [5, 148, 149].

Generalization

GP is a machine learning (ML) algorithm [218]. An ML technique approximates a given process, by searching through the data associated with the process to look for patterns and/or regularities in the data [219]. At the origination of GP, Koza [1] tackled simplified instances of well-known ML problems¹ by using a single dataset (i.e. a fitness case set) to train GP how to solve a given problem, and reporting results on the same dataset that was used to evaluate the fitness of the candidate solution programs during the course of the evolution. This methodology is justifiable, if we take into account the fact that the problems tackled in [1] are simple problems, used to demonstrate the potential of GP. Nevertheless, ML rarely aims to replicate the training data, but rather to be able to make the correct predictions for data instances that were not involved in the training process [219]. In the ML community, it is a flawed methodology to report only on the training performance. This is because a learning algorithm can overfit the training data, and perform poorly on unseen data from the same process [219, 220]. In this regard, it is also important to evaluate the algorithm's performance on data that was not used in the course of learning. In the literature [219, 221], this is known as the two-datasets methodology, where a training dataset is used in the course of learning, and a test dataset is used to report on the performance of the learning algorithm on unseen data. The two-dataset methodology forms the backbone of empirical study in ML [219]; however the GP community still continues to publish papers that report performance results based on training data alone. Eiben and Jelasity [222], note that this is a methodological problem in evolutionary computing research in general, which should not persist in future.

This study applies the two-dataset methodology. In GP literature, the two-dataset methodology has been applied in studies that explicitly attempt to improve on generalization [151, 223, 224]: here, the use of training and test sets is critical with respect to determining if generalization is actually improved. In chapter 3, it was

¹The symbolic regression problems tackled in [1] are simplified instances of real-world regression problems; the artificial ant problem [1] is a simplified instance of reinforcement learning; furthermore, the Boolean function synthesis problems tackled in [1] are simplified instances of supervised binary classification.

established that a number of the DS measures (e.g. DSS) were formulated to improve on GP's generalization capability. Training and test sets are also used in studies that aim to improve on GP's overall performance [181, 225]. Here, the literature acknowledges the importance of testing GP's generalization capability when it comes to validating improvements on the algorithm [181, 225]. Also, in our case, a fitness measure that is designed to improve some other aspect of GP, may implicitly improve on generalization. For example, in [181], HP, which is designed to steer GP towards higher OF scores, is found to improve on GP's generalization. Overall, despite their diverse motivations, the different fitness measures are compared with respect to their performance on both a training and test set.

The extent of generalization to a test set is determined for the best solution program on each GP generation. This is referred to as the generational BS generalization index. Also, the extent of generalization to the test set is measured for the overall best solution program found by GP and is referred to as the final BS generalization index. The generalization index (GI) is measured as the difference in the OF scores produced on the training and test sets:

$$GI = |F_1(best, m_{t1}) - F_1(best, m_{t2})| \quad (4.10)$$

whereby:

- i. GI is the generalization index.
- ii. $F_1(best, m_{t1})$ is the OF score of the candidate solution program on the training set, m_{t1} .
- iii. $F_1(best, m_{t2})$ is the OF score of the candidate solution program on the test set, m_{t2} .

Population diversity

There is a lack of clear correlation between diversity and solution quality in the literature. Gustafson et. al. [56, 57, 226, 227] argue that there is a danger in assuming that high diversity always leads to better quality solution programs. Rather, different levels of fitness diversity suit different problems [56]. For example, problems that suffer from local optima benefit from high fitness diversity, whereby recombination with the lower-fitness solution programs can help to get GP out of the local optima [56]. On the other hand, lowering the fitness diversity can also be useful. Low fitness diversity may be associated with useful exploitation and convergence; this occurs when the GP population contains solution programs that are in the basin of a global optimum, such that population convergence leads to the optimum [56]. Ultimately, whether or not high diversity is useful depends on the properties of the specific problem being tackled, as well as the phase of GP. High diversity is associated with high selective pressure, which allows GP to distinguish the high-quality solution programs in the explorative initial generations [1]. In turn, low diversity facilitates exploitation in later generations, whereby convergence to high-quality solution programs can lead to a global optimum [1]. The exploration and exploitation phases may span different generations on different problems, such that different problems require different levels of diversity on the different generations.

The fitness measures will be compared based on semantic diversity. In section 3.4 of chapter 3, the semantics of a program, i , is defined as the ordered set of outputs produced by i over the $|m|$ fitness cases input to GP, expressed as a vector, $S(i, x_n) \in \mathbb{R}^{|m|}$. In the literature [228, 229], the pairwise semantic distance between two candidate solution programs is measured as the mean absolute difference between the semantic vectors produced by the programs. The literature does not define a measure for the semantic diversity of a population. Most of the literature has focussed on designing genetic operators to control the semantic diversity of subsequent GP generations [52, 228–231]; as such, the aim has not been to measure the semantic diversity of a given population, but rather to measure the difference in semantics between parent solution programs and their offspring produced by the genetic operators [52, 228–231]. Nevertheless defining a population semantic diversity measure is paramount to the current study. Based on the formulae in table 4.1, FS, DSS, HP and NS1 perpetuate the semantic diversity of GP populations. FS favors candidate solution programs that solve unique fitness cases compared to the rest of the population; DSS and HP force the candidate solution programs to deal with “difficult” fitness cases, curtailing rapid convergence to solution programs that solve

only the “easy” fitness cases; also, NS1 favors candidate solution programs that have the highest semantic distance to their k -nearest neighbours. In this study, semantic diversity is defined as the diversity of the semantic vectors ($S(i, x_n) \in \mathbb{R}^{|m|}$) in the population. The population semantic diversity measure used borrows from the work of Burke et. al. [56, 57, 226, 227]. Here, the authors [56, 57, 226, 227] estimate the structural diversity of a given population by averaging over the pairwise structural distance between each candidate solution program in the population and the best solution program found by GP so far; in similar fashion, the semantic diversity of a given population is estimated by averaging over the pairwise semantic distance between each candidate solution program in the population and the best solution program found by GP so far. The population semantic diversity calculation is shown in equations 4.11 and 4.12.

$$\delta_s(g) = \frac{\sum_{i=1}^N \text{dist}(i, \text{best}(g))}{N} \quad (4.11)$$

whereby:

- i. $\delta_s(g)$ is the semantic diversity of the population on generation g .
- ii. $\text{dist}(i, \text{best}(g))$ is the pairwise semantic distance between the i^{th} candidate solution program in the population and $\text{best}(g)$, the best solution program found by GP so far on generation g . $\text{dist}(i, \text{best}(g))$ is calculated according to equation 4.12.
- iii. $\sum_{i=1}^N$ represents the summation of the $\text{dist}(i, \text{best}(g))$ over all candidate solution programs contained in the population of size N on generation g . The summation is divided by N to compute an average.

$$\text{dist}(i, j) = \frac{\sum_{n=1}^{|m|} \delta(S(i, x_n), S(j, x_n))}{|m|} \quad (4.12)$$

whereby:

- i. $\text{dist}(i, j)$ is the pairwise semantic distance between i and j .
- ii. m is the fitness case set defined for the given problem. $|m|$ is the size of m .
- iii. $S(i, x_n)$ represents the output value returned by i for the n^{th} fitness case in m .
- iv. In quantitative problems: $\delta(S(i, x_n), S(j, x_n)) = 0$ if $|S(i, x_n) - S(j, x_n)| < 0.01$. Otherwise $\delta(S(i, x_n), S(j, x_n)) = 1$.
In qualitative problems: $\delta(S(i, x_n), S(j, x_n)) = 0$ if $S(i, x_n) = S(j, x_n)$. Otherwise $\delta(S(i, x_n), S(j, x_n)) = 1$.

The fitness measures are also compared based on fitness diversity. Fitness diversity refers to the diversity of OF scores in the GP population [56, 57, 226]. Fitness diversity is an important aspect of this study, because most of the fitness measures are OF-based; BP, FS, DSS and HP all modify OF (see table 4.1). Monitoring the fitness diversity informs on the presence (or absence) of a fitness gradient [56, 57, 226]. The entropy measure, defined in [56, 57, 226], will be used as an indicator of the fitness diversity of a population: this measure not only describes the number of unique fitness values in a population, but also how the existing fitness values are distributed over the population [56, 57, 226]. Equation 4.13, taken from [56, 57, 226], depicts the calculation of a population’s entropy:

$$\delta_e(g) = - \sum_k p_k \times \log(p_k) \quad (4.13)$$

whereby:

- i. $\delta_e(g)$ is the fitness diversity (entropy) of the population on generation g .
- ii. p_k is the proportion of the population that occupies the k^{th} partition of the population: in discrete problems, a partition represents each possible OF score; in continuous problems, a partition is defined to include a subset of OF scores.

Structural complexity

Structural complexity refers to the size of the candidate solution programs evolved by GP. Previous studies have indicated that NS and FS mitigate bloat [4, 68, 170]: hence the fitness measures should yield solutions programs of low structural complexity. The following is measured for each experimental treatment:

1. The mean size of the candidate solution programs in the population on each GP generation.
2. The size of the best solution program found by GP.

Time taken

Fitness evaluation is the most time consuming aspect of GP [11, 120, 121]. In this vein, the experiment also measures the time taken (in seconds) to execute a single GP run.

4.2.3 Benchmark suite

This section specifies the benchmark suite used in the experiment. The suite incorporates a number of the problems described at the origination of GP [1]. Nevertheless, White et. al. [232] argue that a benchmark suite that comprises *only* of the problems in [1] is inadequate: the problems in [1] are simple “toy” problems, used to demonstrate the capability of GP. In this vein, the majority of the problems in the suite come from the new benchmarks that have been proposed for GP in [16, 232]. The “older” problems in the suite provide the benefit of familiarity: they are widely implemented in the literature, such that information about the properties of these problems is readily accessible. A number of the new benchmarks are also well documented [151, 181, 223, 229, 233, 234].

The benchmark suite employed in this study consists of 4 classes of problems: symbolic regression problems, supervised classification problems, Boolean function synthesis problems and path-finding problems. Symbolic regression problems are the most common benchmarks tackled in GP literature [232]. Supervised classification problems also popular benchmarks [232]. Also, a number of supervised classification problems are based on challenging real-world datasets [232]. In turn, Boolean function synthesis problems are good tests for techniques designed to exploit functional redundancy in a problem. Functional redundancy refers to the reuse of input or intermediate values and functionality; most Boolean function synthesis problems require the reuse of intermediate modules [1]. BP is relevant to such problems because the BP fitness measure exploits modularity to improve on the performance of GP [5, 66]. Path-finding problems do not deal with the arithmetic (or Boolean) input/output typical of the other 3 listed problem domains; rather, in these problems, each fitness case is an environment for training or testing real or simulated robot behaviors [1]. The path-finding domain is included to ensure that non-arithmetic problems are also tackled. Note that FS, DSS, and HP are formulated to work on problems where a set of fitness cases is defined. In this vein, the problems selected for the study are problems in which a set of fitness cases is defined, rather than a single fitness case.

The ensuing sections detail the symbolic regression, supervised classification, Boolean function synthesis and path-finding problems incorporated in the benchmark suite.

Symbolic regression benchmarks

In the symbolic regression domain, each candidate solution program is a regression model - a functional relationship between system inputs and outputs [1]. The aim of search is to find the solution program that best approximates the target outputs over a set of example inputs [1]. The candidate solution programs are evolved on m fitness cases, whereby each fitness case, (x_n, y_n) , is a real-valued input-output pair. The solution quality is measured as shown in the quantitative case in equation 4.1, where the m real-valued outputs produced by a candidate solution program are compared with real-valued target outputs.

For each regression problem tackled in the study, the arithmetic function set $\{+, -, \times, \%, \sin, \cos, \log, \exp\}$ [1] is employed: here, the division (%) and logarithm (log) functions used are protected, such that the result of division is 1 whenever the denominator is 0, and the argument of the logarithm is always converted to its absolute value [1]. The terminal set used represents the inputs to the given problem, and as such, the number of terminals in the terminal set depends on the number of inputs to the problem. Table 4.3 details the problems tackled in the study. In the table, a target function is specified for the synthetic problems (sextic, Keijzer-6, Nguyen-7, Vladislavleva-4, Pagie-1) a priori; on the other hand, the target function is not known in the real-world case (i.e. Dow-Chemical dataset). For the training and testing sets defined in table 4.3, $U[a, b, c]$ represents c random numbers drawn with uniform probability from the interval $[a, b]$. In turn, $E[a, b, c]$ represents a grid of points evenly spaced with a gap of c , drawn from the interval $[a, b]$. The training and test sets are taken from [1, 181, 232]. A suitable test set is generated in cases where the literature does not specify a test set.

TABLE 4.3: Symbolic regression benchmarks, adapted from [232]

Benchmark problem	Target function	No. of inputs Terminal set	Training set Test set
Sextic [77]	$x^6 - 2x^4 + x^2$	1 {x}	E[-1, 1, 0.04] E[-1, 1, 0.06]
Nguyen-7 [229, 232] (abbrev. Nguyen)	$\ln(x + 1) + \ln(x^2 + 1)$	1 {x}	U[0, 2, 20] U[0, 2, 30]
Pagie-1 [181, 232] (abbrev. Pagie)	$\frac{1}{1+x_0^{-4}} + \frac{1}{1+x_1^{-4}}$	2 { x_0, x_1 }	E[-5, 5, 0.4] E[-5, 5, 0.1]
Keijzer-6 [223, 232] (abbrev. Keijzer)	$\sum_{j=1}^x \frac{1}{j}$	1 {x}	E[1, 50, 1] E[1, 120, 1]
Vladislavleva-4 [151, 232] (abbrev. Vlad.)	$\frac{10}{5 + \sum_{j=0}^4 (x_j - 3)^2}$	5 { x_0, x_1, x_2, x_3, x_4 }	U[0.05, 6.05, 1024] U[-0.25, 6.35, 5000]
Dow Chemical dataset [232] (Real-world problem) (abbrev. Dow)	Not known	57 { $x_0, x_1, x_2, x_3, \dots, x_{56}$ }	747 points 319 points

The problems in table 4.3 vary with respect to the degree of difficulty. The sextic polynomial is a simple problem tackled in the early years of GP [77]. The Nguyen problem is also a simple problem, where OF-GP solves the problem in [229]. In turn, the Pagie problem is a harder problem, which is not solved by OF-GP, but is solved by employing host-parasite coevolution in [181]. The Keijzer problem is considered to be a simple problem, but the difficulty is raised with the test data, whereby a number of the test fitness cases are drawn from outside the interval of the training set inputs, as shown in table 4.3; therefore, the solution programs may perform poorly on the test instances drawn from outside the training set domain [232, 235]. The Vladislavleva problem has a reputation for being difficult to solve with GP [151, 232]; also, as in the Keijzer case, a number of the test fitness cases fall outside the interval of the training set inputs [232]. Finally, the Dow Chemical dataset represents a challenging real-world problem that was the subject of the symbolic regression EvoCompetitions event of the 2010 EvoStar conference [236].

Table 4.3 lists the test sets specified in the literature for the Keijzer [223, 232], Vladislavleva [151, 232], Pagie [15] and Dow Chemical dataset [232, 236] problems. No test sets are specified in the literature for the sextic [77] and Nguyen [232] problems. According to White et. al. [232], in cases where the test sets are not explicitly defined, a testing protocol can still be implemented by using a test set containing data points not used for training. This is done in the case of the sextic and Nguyen problems in table 4.3.

Supervised classification benchmarks

In the supervised classification domain, a candidate solution program predicts the value of a categorical attribute (the class attribute), based on the values of other attributes (predicting attributes) [237, 238]. The aim of search is to find the solution program that best approximates the target class attributes over a set of example predicting attributes [1]. The candidate solution programs are evolved on m fitness cases, whereby each fitness case, (x_n, y_n) , is an input-output pair with real-, integer- and/or categorical-valued inputs and categorical output. The solution quality is measured as shown in the qualitative case in equation 4.1, where the m categorical outputs produced by a candidate solution program are compared with target categorical outputs.

When solving supervised classification problems, an arithmetic parse tree representation is arguably the simplest encoding for the candidate solution programs [238]; here, the problem inputs are first converted into real numbers (e.g. categorical inputs are coded as numerical factors), and then input to an arithmetic tree; subsequently, the real-valued output returned from traversing the tree is translated into a class label [238], as detailed in the ensuing paragraph. Previous investigations have also used GP to generate decision tree classifiers [239, 240]. A decision tree is a tree structure in which each internal node represents a “test” on a predicting attribute, each branch represents the outcome of the test, and each terminal node represents a class label (i.e. the class assignment determined after computing all predicting attributes) [239, 240]. Decision trees adopt a more literal approach to representing classifications; hence it becomes difficult to represent classifications that involve complex interactions between the predicting attributes [238]. Logical trees are another alternative for representing the candidate solution programs [241]. A logical tree is a logical expression that assigns a class label to a given instance, based on how the predicting attributes of the instance conform to the conditions encoded in the tree [242, 243]. Logical trees offer a less constrained representation than decision trees, whereby the internal nodes are Boolean functions (e.g. *IF-THEN-ELSE*, *AND*, *OR*, *NOT*) connecting tests on the predicting attributes, and the terminal nodes are class labels and the different values that the predicting attributes can assume [242]. In supervised classification problems, each data instance should be classified as belonging to one and only one class [243]. However a logical tree may contain conflicting conditional clauses that assign two or more different classes to a single instance [243]. On the other hand an instance that does not satisfy any of the clauses in the tree will not be assigned a class label [243]. Such scenarios call for additional mechanisms to be incorporated into GP to mitigate their occurrence [241–243], thus deviating from canonical GP.

Based on the above arguments, the current study adopts an arithmetic parse tree representation for the tackled supervised classification problems. The term static range selection (SRS) is used to describe the process of converting real-valued output from the arithmetic trees into class labels [238]. In SRS, each candidate solution program is an arithmetic expression combining the standard arithmetic operators (+, −, *, %) and the predicting attributes of the given problem. Hence the function set {+, −, *, %} is used, whereas the terminal set depends on the tackled problem - see table 4.4. A candidate solution program returns a real-valued output for each fitness case, whereby the real-valued output is the result of evaluating the arithmetic expression on the fitness case. SRS calls for arbitrary class boundaries to be defined on the real number line [238]. For example, the following boundaries can be defined for a binary classification problem: $Class1 = [-inf, 0]$, $Class2 = [0, inf]$, whereby *inf* abbreviates infinity. Here, a candidate solution program makes the prediction “Class 1” by returning a value in the interval $[-inf, 0]$, and the prediction “Class2” by returning a value in the interval $[0, inf]$ [238]. Nevertheless, it becomes less intuitive to define such boundaries for a multi-class classification. Loveard and Ciesielski [238] propose the following ranges for a 6-class classification: $Class1 = [-inf, -5]$, $Class2 = [-5, -1]$, $Class3 = [-1, 0]$, $Class4 = [0, 1]$, $Class5 = [1, 5]$, $Class6 = [5, inf]$. In this scenario, the classes are not assigned equal ranges; this may (or may not) affect the classification. Loveard and Ciesielski [238] argue that a classifier will only be able to perform as well as the ranges that are arbitrarily chosen.

In the current study, the class assignment is generated by applying the modulus (*mod*) operator, such that:

$$class_assignment = (real_valued_output) \bmod (number_of_classes)$$

For example, given a problem with 6 classes, the class assignment will take on any of the values 0, 1, 2, 3, 4 or 5, (*real_valued_output*) mod 6, depending on the real-valued output. The modulus operator removes the need to define arbitrary ranges, and ensures uniform distribution of the classes for numeric outputs in the interval $[-inf, inf]$. The author has examined the scale of the attributes for each problem tackled, and ensured that the programs have the ability to output all of the relevant classes.

Most of the supervised classification problems tackled in the study are taken from the CHIRP suite [232, 244]. Table 4.4 lists the problems tackled in the study. In table 4.4, the columns *Ca*, *Int*, *Re* and *Mi* describe the predicting attributes, indicating true (*T*), when a dataset contains categorical attributes (*Ca*), integer attributes (*Int*), real attributes (*Re*) and missing data (*Mi*) respectively, and false (*F*) otherwise.

TABLE 4.4: Supervised classification benchmarks, adapted from [232]

Benchmark problem	No. of predicting attributes Terminal set	No. of classes	Ca	Int	Re	Mi	Size of training set Training set distribution Size of test set Test set distribution
Iris [245]	4 { d_0, d_1, d_2, d_3 }	3	F	F	T	F	105 (35 instances per class) 45 (15 instances per class)
Credit [244]	14 { $d_0, d_1, d_2, \dots, d_{13}$ }	2	T	T	T	T	345 (Class 1: 153, Class 2: 192) 345 (Class 1: 154, Class 2: 191)
Wine [245]	13 { $d_0, d_1, d_2, \dots, d_{12}$ }	3	F	T	T	F	123 (Class 1: 41, Class 2: 50, Class 3: 32) 55 (Class 1: 18, Class 2: 21, Class 3: 16)
Segment [244]	19 { $d_0, d_1, d_2, \dots, d_{18}$ }	7	F	F	T	F	210 (30 instances per class) 2100 (300 instances per class)
Vowel [244]	10 { $d_0, d_1, d_2, \dots, d_9$ }	11	F	T	T	F	528 (48 instances per class) 462 (42 instances per class)
Opt [244]	64 { $d_0, d_1, d_2, \dots, d_{63}$ }	10	F	T	F	F	3823 (\approx 382 instances per class) 1797 (\approx 180 instances per class)

The problems in table 4.4 are selected based on the structural variety of their datasets, where the datasets pose different challenges to classifiers. For example, the iris dataset contains a small number of predicting attributes. Conversely the opt dataset has a large number of predicting attributes. The segment dataset has a small training set in relation to the test set used. The credit dataset contains a mix of categorical and continuous variables; here, the categorical inputs are coded as numerical factors for input into the arithmetic trees. The credit dataset also contains missing data; in the current study, each numerical missing value is replaced with the mean of the corresponding attribute over the fitness cases in the dataset [246]; also, each categorical missing value is replaced with the most common value of the corresponding attribute in the dataset [246]. Importantly, the problems listed in table 4.4 vary with respect to the degree of difficulty. The iris and wine datasets represent “exceedingly” simple problems [232, 245], whereas the segment, opt and credit datasets pose more difficulty to GP [232]. Further information on the datasets is available from the UCI Machine Learning Repository [245]. The repository [245] does not explicitly define test sets for the iris, credit and wine problems; in this case, the training set is delineated as 70% of the complete fitness case set, whereas the remaining 30% makes up the test

set. Here, care is taken to retain the class distribution, whereby both the training and test sets have the same class distribution as the complete fitness case set, as shown in table 4.4.

Boolean function synthesis benchmarks

Boolean function synthesis is an instance of supervised classification [1]. In this problem domain, both the predicting attributes and the class attributes are of the binary data type; that is, the attributes assume the values *TRUE* or *FALSE* [1]. The aim is to find the solution program that can best approximate the target binary class attributes over a set of example binary predicting attributes [1]. The candidate solution programs are evolved on m fitness cases of the form (x_n, y_n) . The solution quality is measured as shown in the qualitative case in equation 4.1, where the m binary outputs produced by a candidate solution program are compared with target binary outputs.

The function and terminal sets used are specific to the tackled problem. In general, the function sets are comprised of Boolean operators (e.g. *AND*, *OR*, *NOT*). In turn, the terminal set used represents the binary inputs to the problem; hence the number of terminals in the terminal set depends on the number of inputs to the problem. The two-datasets methodology is not the standard practice in this domain. Therefore, the following rule is applied to define the training and test sets: for each problem, the training set is generated by drawing 70% of the fitness cases with uniform probability from the complete fitness case set; the remaining 30% of the fitness case set makes up the test set. Table 4.5 details the problems tackled in the study.

TABLE 4.5: Boolean function synthesis benchmarks

Benchmark problem	Function set	No. of inputs Terminal set	Original training set (θ_0) Training set Test set
Even-5 parity [1, 77] (abbrev. Par-5)	{AND, OR, NAND, NOR}	5 { d_0, d_1, d_2, d_3, d_4 }	θ_0 : 2^5 binary numbers $\in [0, 2^{5-1}]$ 70% of θ_0 30% of θ_0
Even-7 parity [1, 77] (abbrev. Par-7)	{AND, OR, NAND, NOR}	7 { $d_0, d_1, d_2, d_3, d_4, d_5, d_6$ }	θ_0 : 2^7 binary numbers $\in [0, 2^{7-1}]$ 70% of θ_0 30% of θ_0
Even-9 parity [1, 77] (abbrev. Par-9)	{AND, OR, NAND, NOR}	9 { $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$ }	θ_0 : 2^9 binary numbers $\in [0, 2^{9-1}]$ 70% of θ_0 30% of θ_0
11-multiplexer [1, 77] (abbrev. Mux-11)	{IF, AND, OR, NOT}	11 { $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, a_0, a_1, a_2$ }	θ_0 : 2^{11} binary numbers $\in [0, 2^{11-1}]$ 70% of θ_0 30% of θ_0
3-bit multiplier [234] (abbrev. Mult-3)	{AND, ANDI, OR, XOR}	6 { $d_0, d_1, d_2, d_3, d_4, d_5$ }	θ_0 : 2^6 binary numbers $\in [0, 2^{6-1}]$ 70% of θ_0 30% of θ_0
4-bit multiplier [234] (abbrev. Mult-4)	{AND, ANDI, OR, XOR}	8 { $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7$ }	θ_0 : 2^8 binary numbers $\in [0, 2^{8-1}]$ 70% of θ_0 30% of θ_0

The problems in table 4.5 vary with respect to the degree of difficulty for GP. In the literature [1, 77], it is common practice to tackle Boolean problems with different input sizes. For example: the Boolean even-5, even-7 and even-9 parity problems are all instances of the same problem; the problems become progressively more difficult for GP as the input size increases [1, 77, 232]. The main differentiator in table 4.5 is that the parity and multiplexer problems are simple problems tackled at the origination of GP [1], while the multiplier problems represent more difficult benchmarks [234]. Modelling digital multipliers is a difficult task for evolutionary techniques, especially when the number of bits in the multiplicands is greater than three [234].

Path-finding benchmarks

In the path-finding domain, a candidate solution program encodes a set of rules to follow in different path-finding scenarios [1, 212, 247–249]. The aim is to find the solution program that achieves the goal path-finding behavior. The candidate solution programs are evolved on m fitness cases, whereby each fitness case, (x_n, y_n) , is an environment for training or testing the behaviors [1, 212, 247–249]. The solution quality is measured as shown in the quantitative case in equation 4.1: here, the quality of a candidate solution program, i , is the sum over the m fitness cases of the negative difference between the number of path-finding points that i earns, $S(i, x_n)$, and the maximum possible path-finding points for the given fitness case, y_n .

The function and terminal sets used are specific to the tackled problem. In the artificial ant problem [1], a candidate solution program controls an agent that collects a number of food pellets located on a trail. Here, the aim is to find a solution program that collects the maximum number of food pellets from the trail within a limited number of time-steps. In the tartarus [212, 247–249] and deceptive tartarus [212] problems, an agent is placed in a cell of an $M \times M$ grid-based world; the grid also contains a number of blocks. In these problems, the aim is to find a solution program that controls the agent to navigate the blocks to the boundary of the world within a limited number of moves [248, 250]. In general, the path-finding terminal sets contain instructions on the actions to be carried out by the agents (e.g. *Move*, *TurnLeft*, *TurnRight*). In turn, the path-finding function sets contain connective functions, whose arguments are executed in order (e.g. *PROGN2*(x, y) denotes executing subtree x , and then y). The function sets also contain problem-specific logical (or branching) functions that instruct on the action that the path-finding agent should take, based on the current state of its environment (e.g. *IF-FOOD-AHEAD*(x, y) denotes executing the subtree x if the given condition is satisfied, otherwise y is executed); in the tartarus and deceptive tartarus problems, the logical functions assume the form *IF-POSITION-IS-STATE*(x, y); whereby *POSITION* refers to a cell in the immediate 3×3 neighbourhood of the tartarus agent e.g. *UR* = upper right, *ML* = middle left, *LM* = lower middle, etc.; and *STATE* refers to the state of the indicated cell i.e. *E* = is empty, *B* = contains a block, and *W* = contains a wall [248, 250]. The tartarus and deceptive tartarus path finding logic is further enhanced by the use of indexed memory to keep track of the agent’s location and orientation on the grid [248, 250]: the read/write functions {*READ0*, *READ1*, *READ2*, *STORE0*, *STORE1*, *STORE2*} provide access to 3 storage locations (0, 1 and 2) [248]. Furthermore, the simple logical functions {*IF*, *NOT*, *EQ*} are incorporated into the function set to influence the agent’s actions based on its current state stored in memory [248]. Table 4.6 details the path-finding problems tackled in the study [248, 250]. The function and terminal sets specified in the table are taken from [1] and [248].

TABLE 4.6: Path-finding benchmarks

Benchmark problem	Function set	Terminal set	Training set Test set
Artificial Ant [1, 77, 251] (abbrev. Ant)	<ol style="list-style-type: none"> Connective functions: {<i>PROGN2</i>, <i>PROGN3</i>}. Problem-specific logical function: {<i>IF-FOOD-AHEAD</i>}. 	<ol style="list-style-type: none"> Actions: {<i>Left</i>, <i>Right</i>, <i>Move</i>}. 	<ol style="list-style-type: none"> 7 San Mateo Trail fitness cases. 2 San Mateo Trail fitness cases.
Tartarus [212, 247–249] (abbrev. Tart)	<ol style="list-style-type: none"> Connective function: {<i>PROGN2</i>}. Problem-specific logical functions: {<i>IF-POSITION-IS-STATE</i>} whereby <i>POSITION</i> \in {<i>UM</i>, <i>UR</i>, <i>MR</i>, <i>LR</i>, <i>LM</i>, <i>LL</i>, <i>ML</i>, <i>UL</i>}, and <i>STATE</i> \in {<i>E</i>, <i>B</i>, <i>W</i>}. 	<ol style="list-style-type: none"> Constants (used for comparison): {<i>Zero</i>, <i>One</i>, <i>Two</i>}. Actions: {<i>TurnLeft</i>, <i>TurnRight</i>, <i>Move-Forward</i>}. 	<ol style="list-style-type: none"> 40 randomly generated grid-worlds. 10 randomly generated grid-worlds.
Deceptive Tartarus [212] (abbrev. Dec. Tart)	<ol style="list-style-type: none"> Simple logical functions: {<i>NOT</i>, <i>EQ</i>, <i>IF</i>}. Read/write functions: {<i>READi</i>, <i>WRITEi</i>} whereby $i \in$ {1, 2, 3}. 		

In this study, the artificial ant agent navigates the San Mateo Trail [77, 251]. According to Koza [77], the San Mateo Trail represents a more difficult problem for GP when compared to the Santa Fe trail used at the

origination of GP in [1]. The San Mateo trail is made up of nine parts (or fitness cases), each consisting of a square 13 by 13 grid containing different irregularities in the sequence of food pellets; for example, some of the fitness cases contain single and double gaps between the food pellets. For each fitness case, the movement of the ant is terminated when either of the following 4 conditions occur [77, 251]: 1) the ant touches the outer boundary of the 13 by 13 grid, 2) the ant has executed a total of 120 right or left turns, 3) the ant has moved forward 80 times, or 4) the ant has consumed all the pellets on the trail. In the current study, GP is trained on 7 of the San Mateo fitness cases, while the other 2 cases make up the test set.

NS2, defined in table 4.1, is implemented on the path-finding problems in the experiment. To implement NS2, a problem-specific definition of behavior is required. In the artificial ant problem, the “behavior” of interest is collecting food pellets: therefore, in a novelty search, a novel individual should do something different with respect to collecting the food pellets. The NS2 behavior descriptor is adapted from [4] and specified as follows: the behavior of a candidate solution program is the number of food pellets consumed by an ant agent executing the program after every 40 time steps. In this study, a single time step elapses each time the ant turns or moves forward. Different configurations for the period of time steps on which to measure behavior were tested, and an interval of 40 gave the best result. Given a training set of 7 fitness cases, the behavior vector of a candidate solution program, i , is the concatenation of 7 vectors, $B(i, x_0), B(i, x_1), \dots, B(i, x_7)$. Here, each vector, $B(i, x_n)$, is of length 5 (the value 5 was obtained by empirical tuning), and represents the number of food pellets consumed after every 40 time steps on the n^{th} fitness case. Therefore, each $B(i, x_n)$ is an ordered vector of 5 numbers $\{B(i, x_n, 40), B(i, x_n, 80), B(i, x_n, 120), B(i, x_n, 160), B(i, x_n, 200)\}$, which represents the number of food pellets consumed at 40, 80, 120, 160 and 200 time steps. Fixing the length of the $B(i, x_n)$ vectors permits comparison of the behavior of different solution programs, whose execution may span different time steps depending on when the termination criteria are met. For each fitness case, the behavior of the ant agent is not recorded after 200 time steps. Also, if the movement of the ant agent is terminated before 200 time steps have elapsed, each of the remaining entries in the corresponding vector $B(i, x_n)$ is set to the total number of food pellets consumed by the ant so far.

The tartarus problem is taken from [248, 250]. As in [248, 250], the dimensions of the path-finding grid-world used are 6×6 . The grid-world is bounded by impenetrable walls, and contains 6 blocks that are randomly placed in its inner 4 by 4 locations; that is, the grid world is initialized such that none of the blocks touch the walls. The agent is also initially placed at a random location in the world one cell away from the boundary. Only one object can occupy a given location at any one time. The aim is to find a solution program that controls the agent to navigate the blocks to the boundary of the world within a limited number of moves: the ideal solution program navigates 1 block to each of the 4 corners (2 points are earned for each block navigated to a corner), and 2 blocks to any other location near the boundary (1 point is earned for each block navigated to any other location near the boundary), to achieve a maximum score of 10. The training set consists of 40 such randomly generated worlds. In turn, the test set consists of 10 such randomly generated worlds. Following [212], the maximum number of moves allowed in each grid world is 80. The tartarus problem is observed to be difficult for GP [248]. Based on the data in [248], the best solution program found by GP achieves only slightly above average fitness.

In the tartarus problem, the “behavior” of interest is navigating the blocks to the desired positions in order to accumulate a high score. The NS2 behavior descriptor for this problem is specified as follows: the behavior of a candidate solution program is the sequence of scores recorded at intervals of 20 time steps; a single time step elapses each time the agent executes one of the instructions in the terminal set (for example: turn, move forward, read register, etc.). The interval value of 20 time steps was obtained by empirical tuning. Given the training set of 40 fitness cases, the behavior vector of a candidate solution program, i , is the concatenation of 40 vectors, $B(i, x_0), B(i, x_1), \dots, B(i, x_{40})$. Here, each vector, $B(i, x_n)$, is of length 4, and represents the score achieved by the agent after every 20 time steps on the n^{th} fitness case. Therefore, each $B(i, x_n)$ is an ordered vector of 4 numbers $\{B(i, x_n, 20), B(i, x_n, 40), B(i, x_n, 60), B(i, x_n, 80)\}$, which represents the scores recorded at 20, 40, 60 and 80 time steps.

The deceptive tartarus problem is identical to the tartarus problem, with the only exception being the way that the candidate solution programs are scored [212]. In the deceptive version, blocks lying against the boundary (but not at a corner) receive a score of -1, and the corners are still worth 2 points each, such that optimal solution programs navigate 4 blocks to the corners, and the other 2 blocks away from the walls to a central position, to achieve a maximum score of 8. Here, the subgoal of navigating a block against the wall is penalized, to make the problem harder [212]. As in the case with the tartarus problem, the training and test sets consist of 40 and 10 such randomly generated worlds respectively. The NS2 behavior descriptor used on this problem is specified in the same way as in the tartarus problem.

4.2.4 Experiment set-up

The fitness measures in table 4.1 are compared based on their performance on the benchmark problems listed in section 4.2.3. Seven treatments (OF, BP1, BP2, FS, DSS, HP and NS1) are run for each of the problems in the symbolic regression, supervised classification and Boolean function synthesis domains. An additional treatment, NS2, is run to make a total of 8 treatments for the problems in the path-finding domain. The following baseline configuration is common to all experimental treatments: each treatment consists of 50 independent runs of GP. Each GP run evolves a population of 500 candidate solutions for 101 generations (i.e. an initial random generation 0 plus 100 additional generations). 101 generations are selected in order to allow GP to improve on the performance criteria over an extended period of time (the convention used in the literature [1, 252] is 51 GP generations). Also, the standard population size is fixed at 500 because from the author's observations, population sizes above 500 did not yield significant improvements in the best solutions found by the treatments. For all treatments, the initial population individuals are generated using the ramped half-and-half method, with the parse trees ranging from depths of 2 to 6, as prescribed in [1].

4.2.5 Technical specifications

The algorithms tested in this study were developed using Java SE (Oracle, version 8; [253]). The `java.util.Random` pseudorandom number generator (PRNG) was used to generate random numbers: to achieve true randomness, the PRNG is seeded using the `java.security.SecureRandom` class². The programs were developed on a computer with the following specifications: Intel(R) Celeron(R) N2840 @ 2.16GHz, 2.00 GB RAM, Windows 8 Enterprise OS. Simulations (trial and final) were run on the Center for High Performance Computing, South Africa³. The analysis of the results was performed using Microsoft Excel 2010 [254] and Wolfram Mathematica 10.0 [255].

4.3 Results and discussion

A discussion of the results ensues. The results are organized according to the criteria specified in section 4.2.2. Overall, the results show that the different fitness measures largely address the intended limitations. Nevertheless, the No Free Lunch (NFL) theorems [14] apply: no one fitness measure achieves the best result on all problems with respect to a given criterion. For example, no one fitness measure achieves the best solution quality on all problems: a fitness measure, f_α , may be designed to improve on solution quality, but the extent to which f_α can achieve this depends on the properties of the specific problem being solved. Ultimately, the benchmark problems exhibit different properties: thus different fitness measures achieve the best results on different problems.

²The `SecureRandom` class achieves true randomness by seeding itself from sources of entropy (or environmental noise) available from the local machine, such as timings of input/output events [253].

³See <https://www.chpc.ac.za/index.php/resources/lengau-cluster> for cluster specifications.

4.3.1 Solution quality

A presentation of the solution quality results from each domain ensues.

Symbolic regression benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows. In the detailed results, a comparison is drawn among the fitness measures on each problem; subsequently, statistical tests are conducted to confirm the significance of the observations.

Performance overview

Figures 4.1 and 4.2 indicate the mean final BS-OF scores achieved on the training and test sets respectively. The mean final BS-OF scores are also listed in tables 4.7 and 4.8, where the best scores achieved on each problem are highlighted. In turn, figure 4.3 shows the mean generational BS-OF scores achieved on the training set. In the figures and tables below, higher scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures generally achieve high training and test scores on the trivial sextic problem, but low scores on the difficult Vlad and Dow problems. In general, the more difficult the problem, the poorer the performance of the fitness measures; a discussion on the relative difficulty of the problems was included in section 4.2.3. Critically, different fitness measures achieve the best result on the different problems. For example, BP2 achieves the highest training and test scores on the sextic problem, but among the lowest test scores on the Pagie, Keijzer, Vlad and Dow problems. Also, HP achieves the best training and test scores on the Pagie problem, as well as the highest test scores on the Vlad and Dow problems, but is only the second best performing fitness measure on the sextic, Nguyen and Keijzer test sets. On the other hand, DSS achieves the highest test score on the Nguyen problem, but only achieves a moderate test score on the Pagie problem. Furthermore, NS1 typically achieves the lowest training and test scores on most problems, but achieves better test scores than BP1 and BP2 on the Keijzer problem. Overall, the performance of a given fitness measure is shown to depend on the particular problem being tackled.

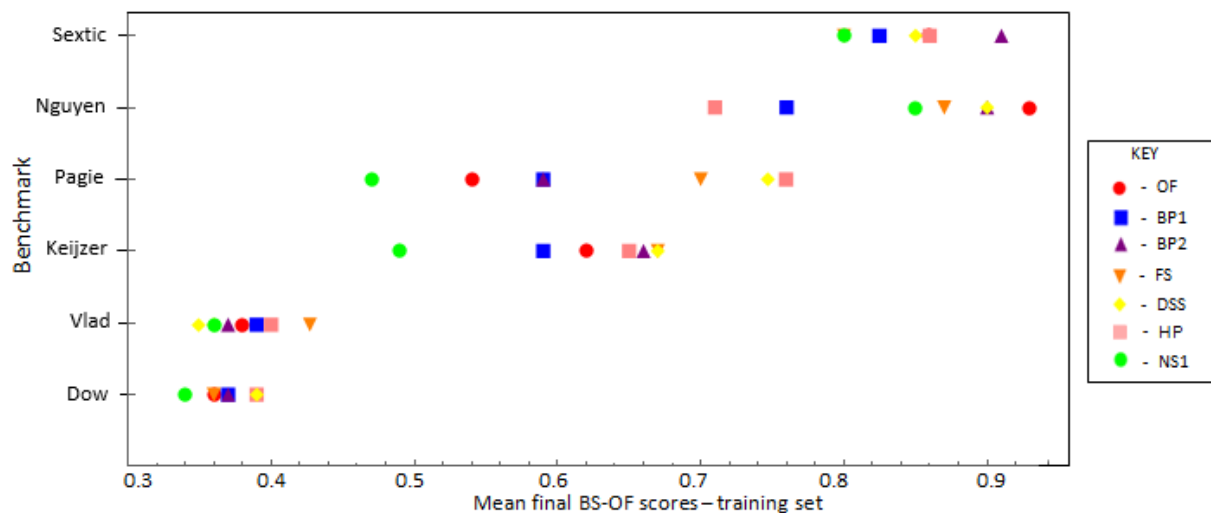


FIGURE 4.1: Symbolic regression benchmarks: Mean final BS-OF scores - training set

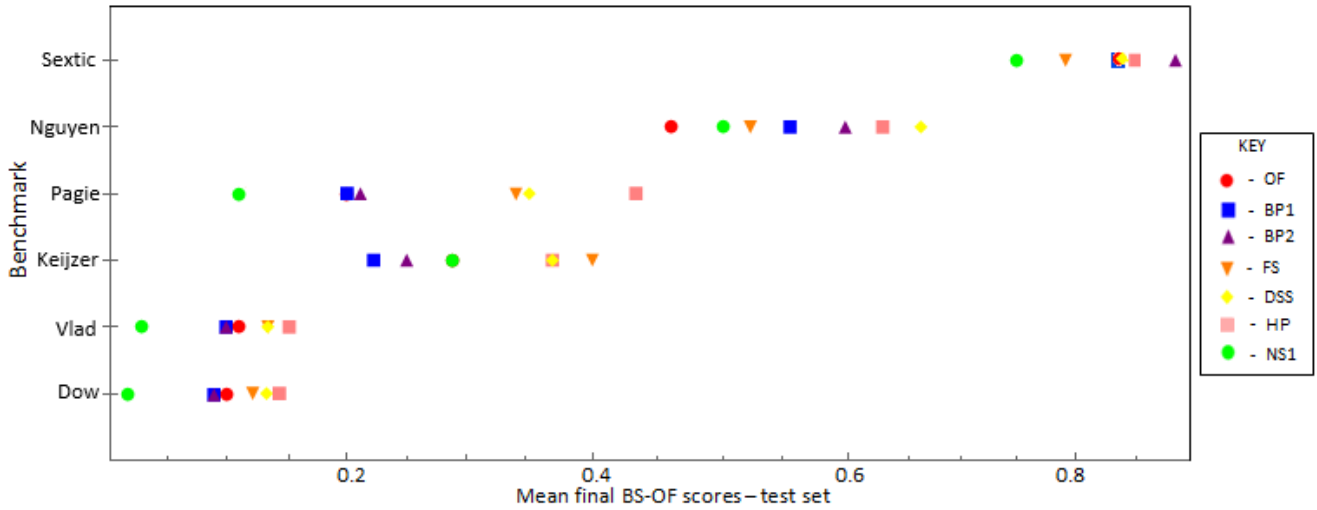


FIGURE 4.2: Symbolic regression benchmarks: Mean final BS-OF scores - test set

TABLE 4.7: Symbolic regression benchmarks: Mean final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	0.86	0.84	0.91	0.80	0.85	0.86	0.80
Nguyen	0.93	0.76	0.90	0.87	0.90	0.71	0.85
Pagie	0.54	0.59	0.59	0.70	0.74	0.75	0.47
Keijzer	0.62	0.59	0.66	0.67	0.67	0.65	0.49
Vlad	0.38	0.39	0.37	0.44	0.35	0.40	0.36
Dow	0.36	0.37	0.37	0.36	0.39	0.39	0.34

TABLE 4.8: Symbolic regression benchmarks: Mean final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	0.84	0.84	0.88	0.79	0.84	0.85	0.75
Nguyen	0.46	0.56	0.60	0.52	0.66	0.63	0.50
Pagie	0.20	0.20	0.21	0.35	0.36	0.45	0.11
Keijzer	0.29	0.22	0.25	0.40	0.37	0.37	0.29
Vlad	0.11	0.10	0.10	0.13	0.13	0.15	0.03
Dow	0.10	0.09	0.09	0.12	0.13	0.14	0.02

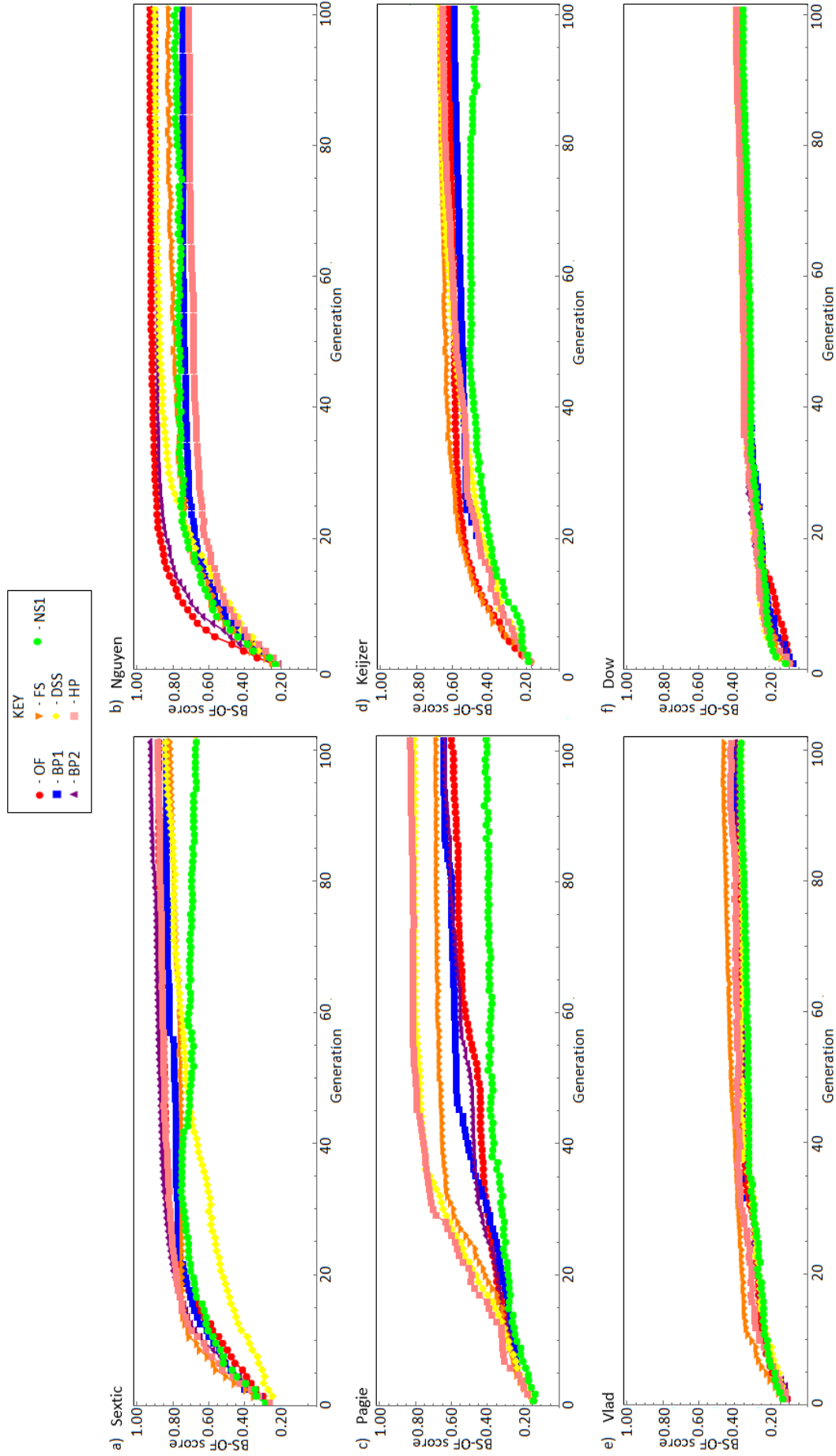


FIGURE 4.3: Symbolic regression benchmarks: Mean generational BS-OF scores

Detailed results

The benchmarks are discussed in the following order: A) Sextic, B) Nguyen, C) Pagie, D) Keijzer, E) Vlad, and F) Dow.

A. Sextic (highest solution quality = BP2)

Tables 4.9 and 4.10 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables. Note that the mean scores shown in tables 4.9 and 4.10 are the same scores shown for the sextic problem in tables 4.7 and 4.8; the analysis in this section looks at the scores achieved on the sextic problem in detail.

TABLE 4.9: Sextic: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.07	0.03	0.13	0.00	0.00	0.00	0.00
b	1.00	1.00	1.00	0.91	0.93	0.90	0.91
μ	0.86	0.84	0.91	0.80	0.85	0.86	0.80
σ	0.03	0.04	0.04	0.04	0.06	0.06	0.06

TABLE 4.10: Sextic: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.92	0.90	0.94	0.84	0.94	0.91	0.82
μ	0.84	0.84	0.88	0.79	0.84	0.85	0.75
σ	0.07	0.06	0.06	0.05	0.04	0.04	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.9 and 4.10. The statistical tests include:

1. A single factor ANOVA [256] conducted on the training BS-OF scores. In the ANOVA, the null hypothesis, H_O , is that $\mu_{OF} = \mu_{BP1} = \mu_{BP2} = \mu_{FS} = \mu_{DSS} = \mu_{HP} = \mu_{NS1}$; the alternate hypothesis, H_A , is that at least one of the means is different. The single factor ANOVA is repeated for the test BS-OF scores.
2. A series of pairwise one-tail z-tests [256] comparing the training BS-OF scores for each pair of the fitness measures. In each pairwise z-test, the null hypothesis H_O is that $\mu_1 = \mu_2$; the alternate hypothesis H_A is that $\mu_1 > \mu_2$, whereby μ_1 is the greater of the two means in the data. The pairwise z-tests are repeated for the test BS-OF scores.

Tables 4.11 and 4.12 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.11: Sextic: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.02						
BP2	0.00	0.00					
FS	0.00	0.01	0.00				
DSS	0.23	0.00	0.00	0.00			
HP	0.25	0.01	0.00	0.00	0.27		
NS1	0.00	0.00	0.00	0.29	0.00	0.00	

TABLE 4.12: Sextic: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.35						
BP2	0.00	0.03					
FS	0.00	0.00	0.00				
DSS	0.30	0.39	0.00	0.00			
HP	0.07	0.11	0.02	0.00	0.17		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that BP2 yields the best solution quality. BP2 achieves significantly higher training and test scores than the other fitness measures. Furthermore, BP2 achieves the highest success rate on the training set. Conversely, NS1 achieves significantly lower training scores than most fitness measures, as well as the lowest test scores on the problem.

B. Nguyen (highest solution quality = OF, DSS)

Tables 4.13 and 4.14 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.13: Nguyen: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.03	0.00	0.00	0.00	0.00	0.00	0.00
b	1.00	0.81	0.95	0.96	0.94	0.82	0.89
μ	0.93	0.76	0.90	0.87	0.90	0.71	0.85
σ	0.04	0.06	0.04	0.06	0.07	0.09	0.03

TABLE 4.14: Nguyen: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.50	0.63	0.68	0.61	0.74	0.70	0.61
μ	0.46	0.56	0.60	0.52	0.66	0.63	0.50
σ	0.07	0.07	0.06	0.08	0.05	0.05	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.13 and 4.14. Tables 4.15 and 4.16 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.15: Nguyen: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.02				
DSS	0.02	0.00	0.29	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.05	0.00	0.00	

TABLE 4.16: Nguyen: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.02	0.00	0.01		
NS1	0.00	0.00	0.00	0.07	0.00	0.00	

The results indicate that OF achieves significantly higher training scores than the other fitness measures, but the lowest test scores. In turn, DSS achieves significantly higher test scores than the other fitness measures on the problem.

C. Pagie (highest solution quality = DSS, HP)

Tables 4.17 and 4.18 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.17: Pagie: Final BS-OF score - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Max	0.59	0.63	0.64	0.73	0.82	0.84	0.55
μ	0.54	0.59	0.59	0.70	0.74	0.75	0.47
σ	0.04	0.03	0.03	0.04	0.05	0.04	0.06

TABLE 4.18: Pagie: Final BS-OF score - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Max	0.28	0.27	0.29	0.41	0.43	0.49	0.20
μ	0.20	0.20	0.21	0.35	0.36	0.45	0.11
σ	0.07	0.06	0.06	0.05	0.07	0.05	0.09

Statistical tests are conducted to verify the significance of the observations in tables 4.17 and 4.18. Tables 4.19 and 4.20 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.19: Pagie: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.34					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.20		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.20: Pagie: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.19					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.22			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that DSS and HP achieve significantly higher training scores than the other fitness measures on the problem. Also, HP achieves the highest test scores. Conversely, NS1 achieves the significantly lower training and test scores than the other fitness measures.

D. Keijzer (highest solution quality = BP2, FS, DSS, HP)

Tables 4.21 and 4.22 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.21: Keijzer: Final BS-OF score - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.69	0.71	0.76	0.76	0.75	0.73	0.61
μ	0.62	0.59	0.66	0.67	0.67	0.65	0.49
σ	0.08	0.09	0.09	0.09	0.07	0.07	0.11

TABLE 4.22: Keijzer: Final BS-OF score - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Max	0.40	0.32	0.34	0.49	0.45	0.46	0.37
μ	0.29	0.22	0.25	0.40	0.37	0.37	0.29
σ	0.10	0.12	0.11	0.10	0.09	0.09	0.12

Statistical tests are conducted to verify the significance of the observations in tables 4.21 and 4.22. Tables 4.23 and 4.24 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.23: Keijzer: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.01						
BP2	0.00	0.00					
FS	0.00	0.00	0.25				
DSS	0.00	0.00	0.22	0.32			
HP	0.04	0.00	0.21	0.19	0.23		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.24: Keijzer: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.01						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.32		
NS1	0.42	0.00	0.00	0.00	0.00	0.00	

The results indicate that BP2, FS, DSS and HP achieve significantly higher training scores than the other fitness measures on the problem. In turn, FS achieves significantly higher test scores than the other fitness measures on the problem.

E. Vlad (highest solution quality = FS, DSS, HP)

Tables 4.25 and 4.26 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.25: Vlad: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.41	0.43	0.40	0.48	0.39	0.45	0.42
μ	0.38	0.39	0.37	0.44	0.35	0.40	0.36
σ	0.03	0.03	0.04	0.03	0.04	0.04	0.07

TABLE 4.26: Vlad: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.14	0.15	0.15	0.18	0.17	0.21	0.08
μ	0.11	0.10	0.10	0.13	0.13	0.15	0.03
σ	0.04	0.07	0.07	0.06	0.07	0.05	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.25 and 4.26. Tables 4.27 and 4.28 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.27: Vlad: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.22						
BP2	0.25	0.13					
FS	0.00	0.00	0.00				
DSS	0.10	0.00	0.17	0.00			
HP	0.15	0.32	0.05	0.00	0.00		
NS1	0.13	0.03	0.19	0.00	0.37	0.00	

TABLE 4.28: Vlad: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.11						
BP2	0.12	0.42					
FS	0.07	0.01	0.01				
DSS	0.07	0.00	0.00	0.41			
HP	0.00	0.00	0.00	0.09	0.09		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that a number of the fitness measures achieve on par training and test scores on this difficult problem. FS achieves significantly higher training scores than the other fitness measures, but performs on par with DSS and HP on the test set.

F. Dow (highest solution quality = FS, DSS, HP)

Tables 4.29 and 4.30 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.29: Dow: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.40	0.40	0.39	0.39	0.42	0.43	0.39
μ	0.36	0.37	0.37	0.36	0.39	0.39	0.34
σ	0.03	0.03	0.02	0.02	0.02	0.03	0.03

TABLE 4.30: Dow: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.19	0.16	0.16	0.17	0.17	0.18	0.09
μ	0.10	0.09	0.09	0.12	0.13	0.14	0.02
σ	0.06	0.07	0.07	0.07	0.07	0.06	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.29 and 4.30. Tables 4.31 and 4.32 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.31: Dow: Statistical test on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.22						
BP2	0.23	0.39					
FS	0.41	0.23	0.29				
DSS	0.01	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.34		
NS1	0.01	0.00	0.00	0.01	0.00	0.00	

TABLE 4.32: Dow: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.03							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.25						
BP2	0.22	0.37					
FS	0.05	0.00	0.00				
DSS	0.00	0.00	0.00	0.09			
HP	0.00	0.00	0.00	0.05	0.13		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that as in the previous problem, a number of the fitness measures achieve on par training and test scores on this difficult problem. HP and DSS achieve significantly higher training scores than the other fitness measures. In turn, FS, DSS and HP achieve the best test scores.

Supervised classification benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.4 and 4.5 indicate the mean final BS-OF scores achieved on the training and test sets respectively. The mean final BS-OF scores are also listed in tables 4.33 and 4.34, where the best scores on each problem are highlighted. In turn, figure 4.6 shows the mean generational BS-OF scores achieved on the training set. In the figures and tables below, higher scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures generally achieve high training and test scores on the trivial iris classification problem, but low scores on the difficult vowel and opt classification problems. In general, the more difficult the problem, the poorer the performance of the fitness measures; a discussion on the relative difficulty of the problems was included in section 4.2.3. Critically, different fitness measures achieve the best result on the different problems. For example, NS1 achieves among the highest training and test scores on the segment problem but performs poorly when compared to the other fitness measures on the remaining problems. Also, FS and DSS achieve among the highest training and test scores on the wine and segment problems, but perform moderately when compared to the other fitness measures on the credit problem. Furthermore, BP1 and BP2 achieve competitive training and test scores on the credit problem, but the lowest training and test scores on the segment problem. Overall, the performance of a given fitness measure is shown to depend on the particular problem being tackled.

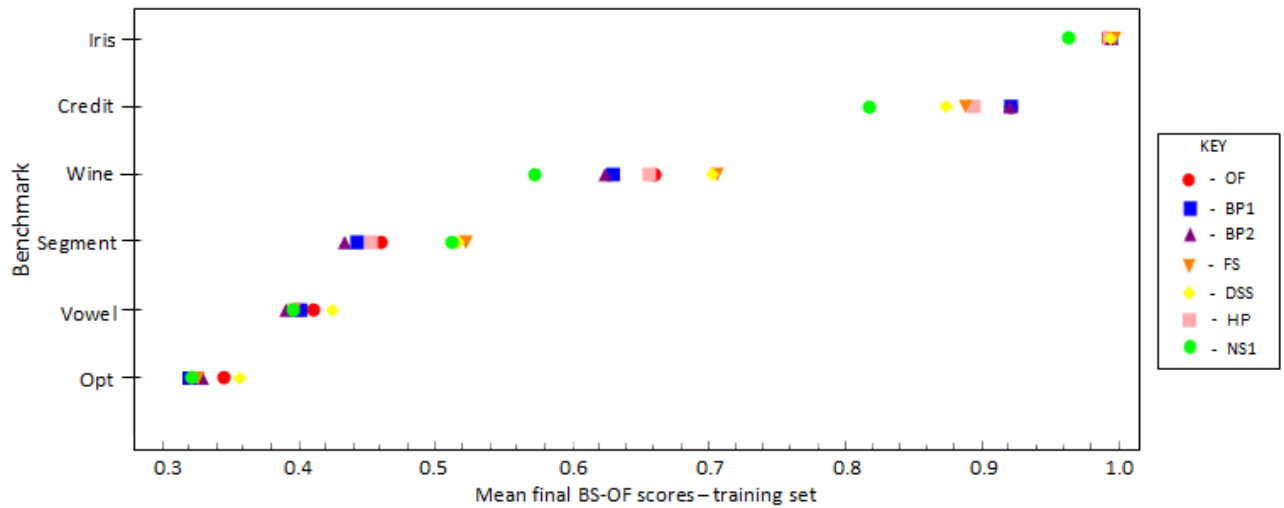


FIGURE 4.4: Supervised classification benchmarks: Mean final BS-OF scores - training set

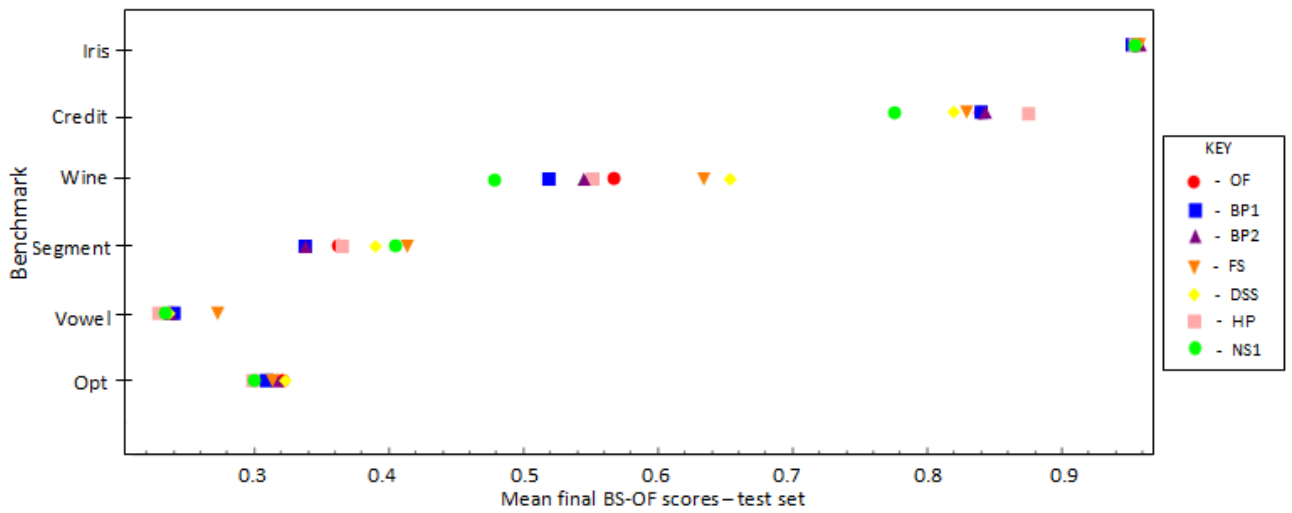


FIGURE 4.5: Supervised classification benchmarks: Mean final BS-OF scores - test set

TABLE 4.33: Supervised classification benchmarks: Mean final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	0.99	1.00	1.00	1.00	1.00	0.99	0.96
Credit	0.92	0.92	0.92	0.88	0.87	0.89	0.82
Wine	0.66	0.63	0.62	0.70	0.70	0.66	0.57
Segment	0.46	0.44	0.43	0.52	0.51	0.45	0.51
Vowel	0.41	0.40	0.39	0.39	0.44	0.40	0.39
Opt	0.34	0.32	0.33	0.32	0.35	0.32	0.32

TABLE 4.34: Supervised classification benchmarks: Mean final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	0.95	0.95	0.96	0.96	0.96	0.95	0.95
Credit	0.84	0.84	0.84	0.83	0.82	0.88	0.78
Wine	0.57	0.52	0.54	0.63	0.65	0.55	0.48
Segment	0.36	0.34	0.34	0.41	0.39	0.36	0.40
Vowel	0.24	0.24	0.24	0.29	0.23	0.23	0.23
Opt	0.32	0.31	0.32	0.31	0.32	0.30	0.30

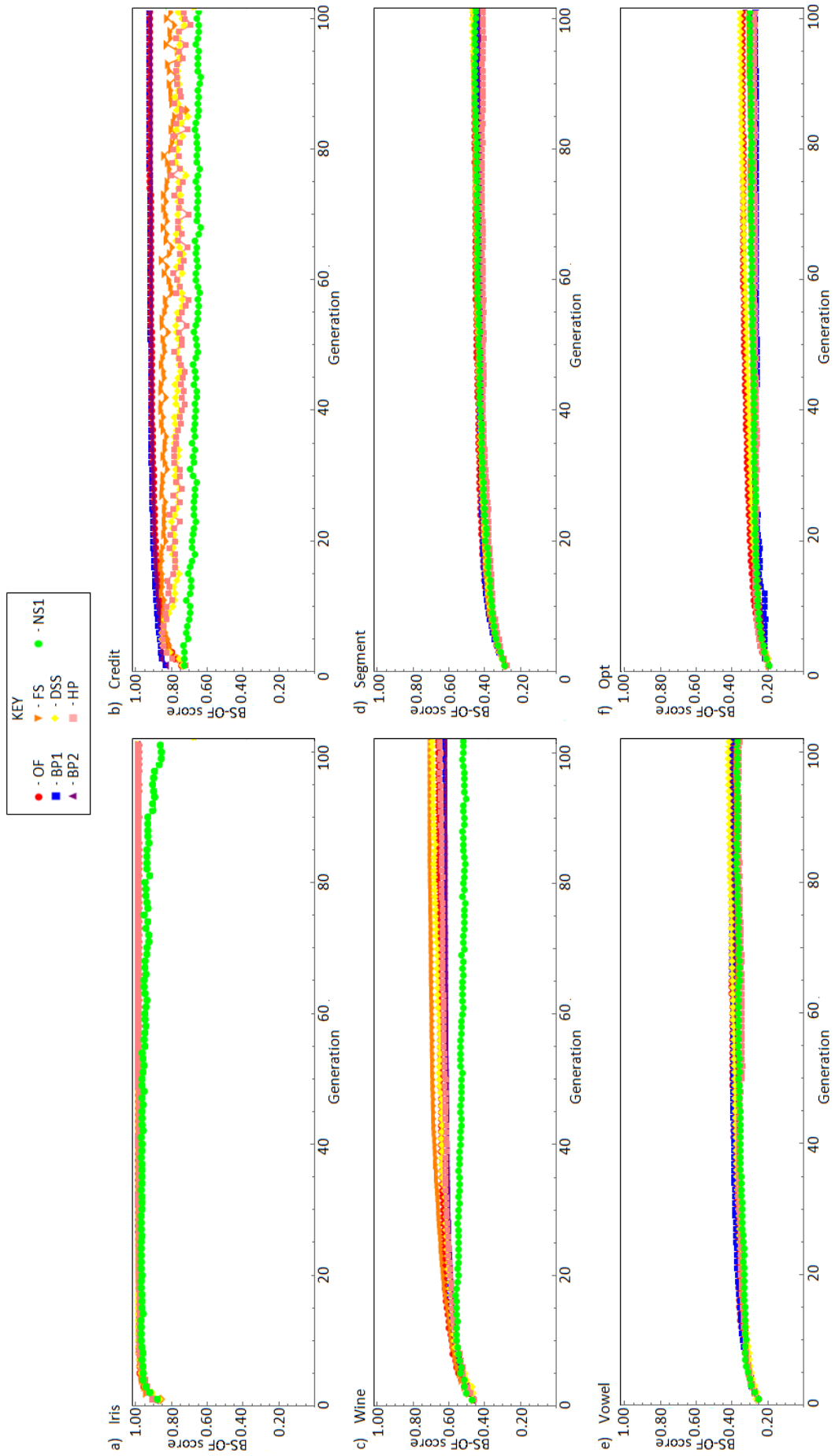


FIGURE 4.6: Supervised classification benchmarks: Mean generational BS-OF scores

Detailed results

The benchmarks are discussed in the following order: A) Iris, B) Credit, C) Wine, D) Segment, E) Vowel, and F) Opt.

A. Iris (highest solution quality = N/A)

Tables 4.35 and 4.36 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. Note that the mean scores shown in tables 4.35 and 4.36 are the same scores shown for the iris problem in tables 4.33 and 4.34; the analysis in this section looks at the scores achieved on the iris problem in detail.

TABLE 4.35: Iris: Final BS-OF scores
- training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.30	0.47	0.57	0.73	0.43	0.47	0.03
b	1.00	1.00	1.00	1.00	1.00	1.00	1.00
μ	0.99	1.00	1.00	1.00	1.00	0.99	0.96
σ	0.02	0.01	0.01	0.01	0.01	0.02	0.01

TABLE 4.36: Iris: Final BS-OF scores
- test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.98	0.98	0.98	0.98	0.98	0.98	0.98
μ	0.95	0.95	0.96	0.96	0.96	0.95	0.95
σ	0.04	0.06	0.04	0.04	0.03	0.05	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.35 and 4.36. Tables 4.37 and 4.38 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.37: Iris: Statistical tests on
final BS-OF scores - training set

One-way ANOVA p-value = 0.07							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.25						
BP2	0.29	0.42					
FS	0.15	0.11	0.13				
DSS	0.27	0.40	0.33	0.15			
HP	0.14	0.20	0.18	0.10	0.27		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.38: Iris: Statistical tests on
final BS-OF scores - test set

One-way ANOVA p-value = 0.72							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.38						
BP2	0.10	0.12					
FS	0.14	0.15	0.42				
DSS	0.32	0.27	0.19	0.25			
HP	0.42	0.33	0.16	0.21	0.42		
NS1	0.50	0.37	0.08	0.12	0.30	0.41	

The results indicate that the fitness measures achieve high and on par training and test scores on this trivial problem. NS1 achieves significantly lower training scores than the other fitness measures. Apart from NS1, the ANOVA p-value of 0.07 (whereby $0.07 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.37 show that the remaining fitness measures achieve on par performance on the training set. Similarly, the ANOVA p-value of 0.72 in table 4.38 indicates that all the fitness measures achieve on par performance on the test set.

B. Credit (highest solution quality = OF, BP1, BP2, HP)

Tables 4.39 and 4.40 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.39 and 4.40. Tables 4.41 and 4.42 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.39: Credit: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.93	0.95	0.95	0.94	0.93	0.94	0.86
μ	0.92	0.92	0.92	0.88	0.87	0.89	0.82
σ	0.03	0.06	0.04	0.07	0.07	0.09	0.08

TABLE 4.40: Credit: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.88	0.86	0.88	0.86	0.86	0.93	0.86
μ	0.84	0.84	0.84	0.83	0.82	0.88	0.78
σ	0.07	0.03	0.05	0.08	0.09	0.10	0.12

TABLE 4.41: Credit: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.47						
BP2	0.38	0.42					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.02			
HP	0.03	0.04	0.08	0.21	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.42: Credit: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.45						
BP2	0.24	0.23					
FS	0.04	0.02	0.01				
DSS	0.00	0.00	0.00	0.07			
HP	0.01	0.01	0.02	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that OF, BP1 and BP2 largely achieve significantly higher training scores than the rest of the fitness measures on the problem. HP achieves the highest test scores. Conversely NS1 achieves significantly lower training and test scores than the other fitness measures.

C. Wine (highest solution quality = FS, DSS)

Tables 4.43 and 4.44 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.43: Wine: Final BS-OF score - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.70	0.68	0.67	0.73	0.73	0.71	0.61
μ	0.66	0.63	0.62	0.70	0.70	0.66	0.57
σ	0.03	0.03	0.02	0.02	0.02	0.03	0.05

TABLE 4.44: Wine: Final BS-OF score - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.69	0.73	0.71	0.71	0.71	0.69	0.59
μ	0.57	0.52	0.54	0.63	0.65	0.55	0.48
σ	0.05	0.06	0.06	0.04	0.04	0.05	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.43 and 4.44. Tables 4.45 and 4.46 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that FS and DSS achieve significantly higher training and test scores than the other fitness measures. On the other hand, NS1 achieves significantly lower training scores than the other fitness measures; also, NS1 achieves significantly lower test scores than FS and DSS.

TABLE 4.45: Wine: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.02						
BP2	0.01	0.32					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.37			
HP	0.40	0.03	0.02	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.46: Wine: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.06	0.01					
FS	0.04	0.00	0.00				
DSS	0.00	0.00	0.00	0.05			
HP	0.25	0.09	0.00	0.01	0.00		
NS1	0.00	0.06	0.00	0.00	0.00	0.00	

D. Segment (highest solution quality = FS, DSS, NS1)

Tables 4.47 and 4.48 list the best (*b*) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.47: Segment: Final BS-OF score - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<i>b</i>	0.55	0.53	0.51	0.70	0.61	0.57	0.61
μ	0.46	0.44	0.43	0.52	0.51	0.45	0.51
σ	0.06	0.06	0.05	0.05	0.06	0.05	0.05

TABLE 4.48: Segment: Final BS-OF score - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<i>b</i>	0.50	0.44	0.41	0.57	0.53	0.46	0.53
μ	0.36	0.34	0.34	0.41	0.39	0.36	0.40
σ	0.07	0.05	0.07	0.07	0.05	0.05	0.06

Statistical tests are conducted to verify the significance of the observations in tables 4.47 and 4.48. Tables 4.49 and 4.50 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.49: Segment: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.07						
BP2	0.01	0.22					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.34			
HP	0.25	0.18	0.03	0.00	0.00		
NS1	0.00	0.00	0.00	0.22	0.35	0.00	

TABLE 4.50: Segment: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.03						
BP2	0.02	0.49					
FS	0.00	0.00	0.00				
DSS	0.03	0.00	0.00	0.06			
HP	0.40	0.02	0.01	0.00	0.03		
NS1	0.00	0.00	0.00	0.30	0.13	0.00	

The results indicate that FS, DSS and NS1 achieve on par training and test scores. Furthermore, FS, DSS and NS1 achieve significantly higher training and test scores than the other fitness measures on the problem.

E. Vowel (highest solution quality = FS, DSS)

Tables 4.51 and 4.52 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.51: Vowel: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.46	0.46	0.45	0.43	0.51	0.47	0.45
μ	0.41	0.40	0.39	0.39	0.44	0.40	0.39
σ	0.06	0.05	0.05	0.04	0.06	0.06	0.05

TABLE 4.52: Vowel: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.31	0.29	0.31	0.35	0.32	0.31	0.29
μ	0.24	0.24	0.24	0.29	0.23	0.23	0.23
σ	0.05	0.07	0.09	0.09	0.10	0.10	0.11

Statistical tests are conducted to verify the significance of the observations in tables 4.51 and 4.52. Tables 4.53 and 4.54 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.53: Vowel: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.11						
BP2	0.01	0.11					
FS	0.01	0.25	0.17				
DSS	0.03	0.00	0.00	0.00			
HP	0.06	0.38	0.15	0.37	0.00		
NS1	0.01	0.24	0.18	0.47	0.00	0.35	

TABLE 4.54: Vowel: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.04							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.43						
BP2	0.47	0.39					
FS	0.03	0.02	0.02				
DSS	0.42	0.32	0.45	0.01			
HP	0.16	0.09	0.19	0.00	0.19		
NS1	0.30	0.22	0.34	0.00	0.36	0.32	

The results indicate that a number of the fitness measures achieve on par training and test scores on this difficult problem. DSS achieves significantly higher training scores than the other fitness measures. In turn, FS achieves significantly higher test scores than the other fitness measures. Apart from FS, all the other fitness measures perform on par on the test set.

F. Opt (highest solution quality = N/A)

Tables 4.55 and 4.56 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved.

TABLE 4.55: Opt: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.40	0.38	0.39	0.38	0.42	0.45	0.36
μ	0.34	0.32	0.33	0.32	0.35	0.32	0.32
σ	0.15	0.11	0.11	0.09	0.13	0.18	0.06

TABLE 4.56: Opt: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.39	0.41	0.41	0.41	0.41	0.45	0.34
μ	0.32	0.31	0.32	0.31	0.32	0.30	0.30
σ	0.14	0.18	0.18	0.11	0.16	0.20	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.55 and 4.56. Tables 4.57 and 4.58 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.57: Opt: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.06							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.01						
BP2	0.06	0.14					
FS	0.03	0.23	0.34				
DSS	0.13	0.00	0.00	0.00			
HP	0.02	0.48	0.17	0.27	0.00		
NS1	0.01	0.40	0.14	0.26	0.00	0.43	

TABLE 4.58: Opt: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.14							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.14						
BP2	0.36	0.25					
FS	0.22	0.34	0.36				
DSS	0.43	0.12	0.31	0.18			
HP	0.03	0.21	0.07	0.10	0.02		
NS1	0.01	0.16	0.03	0.05	0.01	0.47	

The results indicate that most of the fitness measures achieve on par training and test scores on this difficult problem. The ANOVA p-values of 0.06 (whereby $0.06 > \alpha$) and 0.14 (whereby $0.14 > \alpha$) shown in tables 4.57 and 4.58 respectively indicate that the fitness measures largely achieve on par performance on both the training and test sets.

Boolean function synthesis benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.7 and 4.8 indicate the mean final BS-OF scores achieved on the training and test sets respectively. The mean final BS-OF scores are also listed in tables 4.59 and 4.60, where the best scores on each problem are highlighted. In turn, figure 4.9 shows the mean generational BS-OF scores achieved on the training set. In the figures and tables below, higher scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

A key observation made from figures 4.7 and 4.8 is that the fitness measures incur high overfitting on the even-n parity problems: the test BS-OF scores are significantly lower than the training BS-OF scores. This is discussed in detail in section 4.3.2. The results also demonstrate that different fitness measures achieve the best result on the different problems. For example, FS and DSS achieve the highest test scores on the mux-11 problem, whereas BP1 and BP2 achieve the highest test scores on the mult-3 problem. FS and DSS achieve high training and test scores on the mux-11 problem, but low test scores on the par-5, par-7 and par-9 problems. BP1 and BP2 achieve high training and test scores on most problems, but also achieve low test scores on the par-5, par-7 and par-9 problems. Also, HP is shown to achieve among the best test scores on the mult-4 problem, but the worst test score on the mux-11 problem. Overall, the performance of a given fitness measure is shown to depend on the particular problem being tackled.

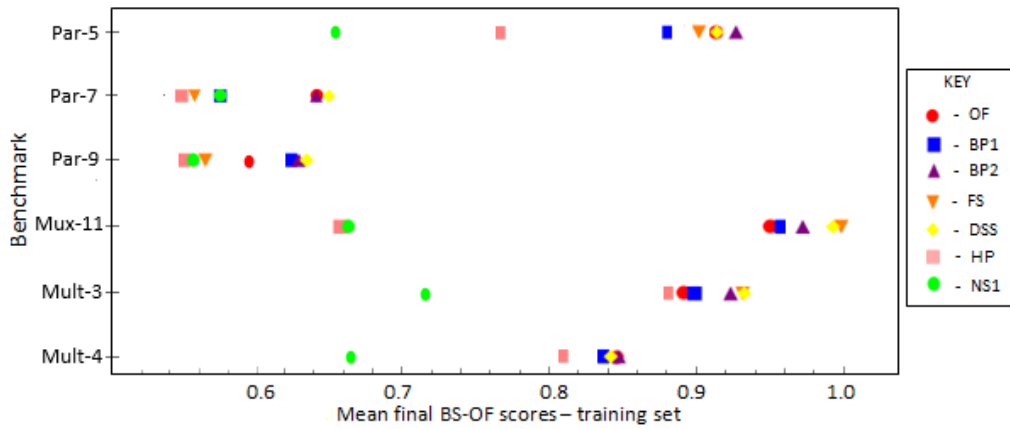


FIGURE 4.7: Boolean function synthesis benchmarks: Mean final BS-OF scores - training set

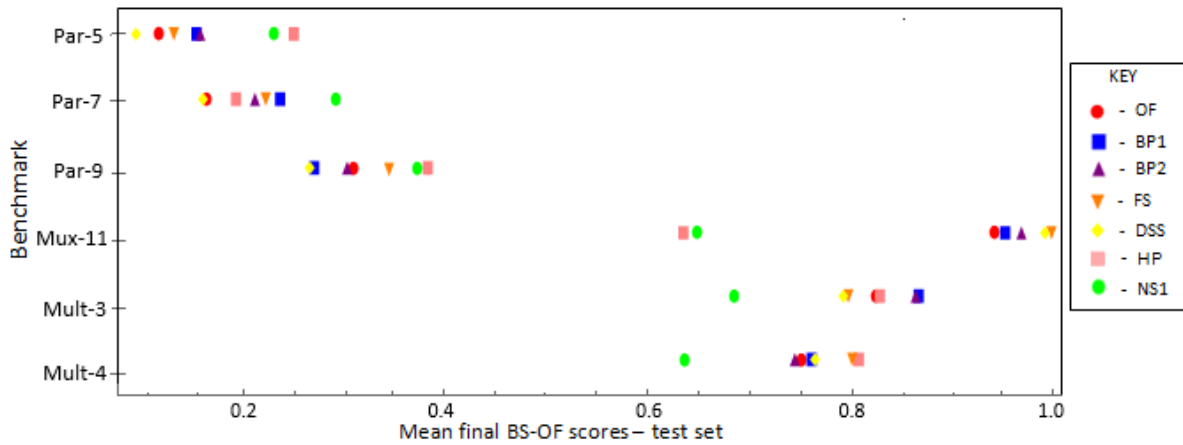


FIGURE 4.8: Boolean function synthesis benchmarks: Mean final BS-OF scores - test set

TABLE 4.59: Boolean function synthesis benchmarks: Mean final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	0.91	0.86	0.93	0.90	0.91	0.77	0.66
Par-7	0.65	0.58	0.65	0.56	0.66	0.54	0.58
Par-9	0.60	0.62	0.63	0.57	0.63	0.55	0.56
Mux-11	0.95	0.96	0.97	1.00	0.99	0.66	0.66
Mult-3	0.89	0.90	0.92	0.93	0.93	0.88	0.71
Mult-4	0.85	0.84	0.85	0.84	0.84	0.81	0.66

TABLE 4.60: Boolean function synthesis benchmarks: Mean final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	0.11	0.15	0.15	0.13	0.09	0.25	0.23
Par-7	0.16	0.24	0.21	0.22	0.15	0.19	0.29
Par-9	0.31	0.27	0.30	0.34	0.26	0.38	0.37
Mux-11	0.94	0.95	0.97	1.00	0.99	0.63	0.65
Mult-3	0.82	0.87	0.86	0.80	0.79	0.83	0.68
Mult-4	0.75	0.76	0.74	0.80	0.76	0.80	0.63

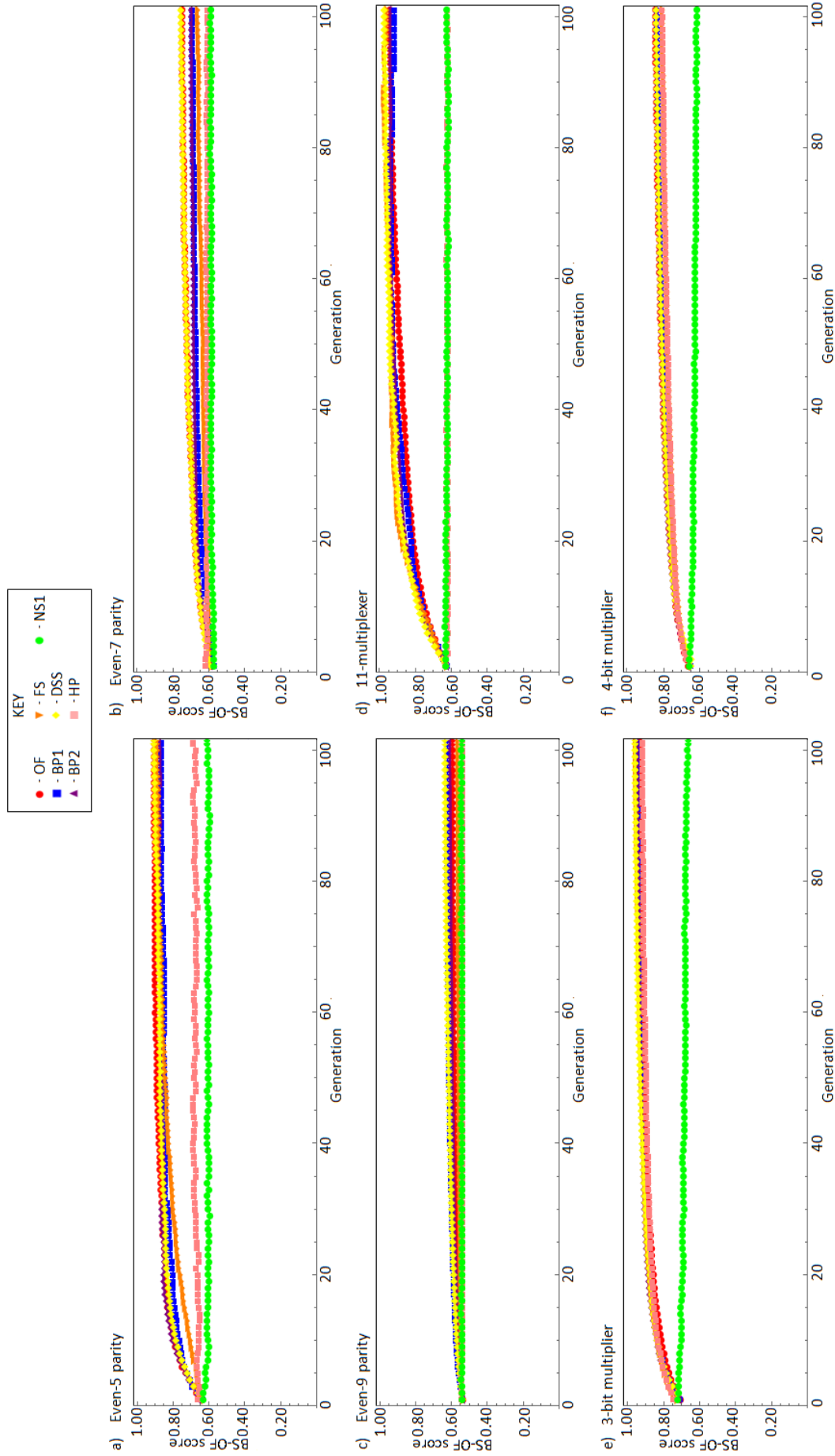


FIGURE 4.9: Boolean function synthesis benchmarks: Mean generational BS-OF scores

Detailed results

The benchmarks are discussed in the following order: A) Even-5 parity, B) Even-7 parity, C) Even-9 parity, D) 11-multiplexer, E) 3-bit multiplier, and F) 4-bit multiplier.

A. Even-5 parity (highest solution quality = BP2, HP, NS1)

Tables 4.61 and 4.62 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables. Note that the mean scores shown in tables 4.61 and 4.62 are the same scores shown for the par-5 problem in tables 4.59 and 4.60; the analysis in this section looks at the scores achieved on the par-5 problem in detail.

TABLE 4.61: Even-5 parity: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.10	0.13	0.43	0.07	0.07	0.00	0.00
b	1.00	1.00	1.00	1.00	1.00	0.91	0.70
μ	0.91	0.86	0.93	0.90	0.91	0.77	0.66
σ	0.04	0.04	0.03	0.02	0.02	0.03	0.02

TABLE 4.62: Even-5 parity: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Max	0.33	0.33	0.22	0.33	0.33	0.67	0.44
μ	0.11	0.15	0.15	0.13	0.09	0.25	0.23
σ	0.09	0.09	0.07	0.10	0.09	0.12	0.09

Statistical tests are conducted to verify the significance of the observations in tables 4.61 and 4.62. Tables 4.63 and 4.64 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.63: Even-5 parity: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.16	0.00					
FS	0.17	0.00	0.03				
DSS	0.50	0.00	0.13	0.12			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.64: Even-5 parity: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.06						
BP2	0.03	0.42					
FS	0.27	0.14	0.08				
DSS	0.12	0.00	0.00	0.04			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.20	

The results indicate that OF, BP2, FS and DSS achieve on par training scores. However, BP2 achieves the highest success rate on the training set. Table 4.63 indicates that NS1 achieves significantly lower training scores than the other fitness measures on the problem. On the other hand, table 4.64 indicates that HP and NS1 achieve significantly higher test scores than the other fitness measures on the problem.

B. Even-7 parity (highest solution quality = OF, BP2, DSS, NS1)

Tables 4.65 and 4.66 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.65 and 4.66. Tables 4.67 and 4.68 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.65: Even-7 parity: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.73	0.62	0.75	0.59	0.72	0.58	0.60
μ	0.65	0.58	0.65	0.56	0.66	0.54	0.58
σ	0.05	0.02	0.05	0.01	0.03	0.01	0.01

TABLE 4.66: Even-7 parity: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.34	0.34	0.29	0.32	0.21	0.26	0.42
μ	0.16	0.24	0.21	0.22	0.15	0.19	0.29
σ	0.04	0.03	0.04	0.05	0.03	0.03	0.07

TABLE 4.67: Even-7 parity: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.46	0.00					
FS	0.00	0.01	0.00				
DSS	0.32	0.00	0.29	0.00			
HP	0.00	0.00	0.00	0.09	0.00		
NS1	0.00	0.47	0.00	0.00	0.00	0.00	

TABLE 4.68: Even-7 parity: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.05	0.14				
DSS	0.41	0.00	0.00	0.00			
HP	0.01	0.00	0.03	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that OF, BP2 and DSS achieve significantly higher training scores than the other fitness measures on the problem. Also, NS1 achieves significantly higher test scores than the other fitness measures on the problem.

C. Even-9 parity (highest solution quality = BP1, BP2, DSS, HP, NS1)

Tables 4.69 and 4.70 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables. Statistical tests

TABLE 4.69: Even-9 parity: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.63	0.66	0.70	0.58	0.69	0.56	0.56
μ	0.60	0.62	0.63	0.57	0.63	0.55	0.56
σ	0.02	0.02	0.04	0.02	0.02	0.01	0.00

TABLE 4.70: Even-9 parity: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.36	0.31	0.33	0.41	0.33	0.39	0.40
μ	0.31	0.27	0.30	0.34	0.26	0.38	0.37
σ	0.03	0.02	0.03	0.03	0.03	0.02	0.01

are conducted to verify the significance of the observations in tables 4.69 and 4.70. Tables 4.71 and 4.72 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that BP1, BP2 and DSS achieve significantly higher training scores than the other fitness measures on the problem. Also, HP and NS1 achieve significantly higher test scores than the other fitness measures on the problem.

TABLE 4.71: Even-9 parity: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.10					
FS	0.00	0.00	0.00				
DSS	0.03	0.29	0.06	0.00			
HP	0.03	0.00	0.00	0.22	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.27	

TABLE 4.72: Even-9 parity: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.35	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.29	0.02	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.20	

D. 11-multiplexer (highest solution quality = FS, DSS)

Tables 4.73 and 4.74 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.73: 11-multiplexer: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.13	0.37	0.50	0.97	0.73	0.00	0.00
b	1.00	1.00	1.00	1.00	1.00	0.69	0.69
μ	0.95	0.96	0.97	1.00	0.99	0.66	0.66
σ	0.03	0.03	0.03	0.01	0.01	0.02	0.03

TABLE 4.74: 11-multiplexer: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.13	0.37	0.50	0.97	0.73	0.00	0.00
b	1.00	1.00	1.00	1.00	1.00	0.68	0.69
μ	0.94	0.95	0.97	1.00	0.99	0.63	0.65
σ	0.04	0.04	0.04	0.02	0.02	0.03	0.03

Statistical tests are conducted to verify the significance of the observations in tables 4.73 and 4.74. Tables 4.75 and 4.76 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.75: 11-multiplexer: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.21						
BP2	0.01	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.27			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.40	

TABLE 4.76: 11-multiplexer: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.21						
BP2	0.01	0.05					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.25			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.12	

The results indicate that FS and DSS achieve significantly higher training and test scores than the other fitness measures on the problem. FS and DSS also achieve the highest success rates on the training and test sets. Conversely, NS1 and HP achieve the lowest training and test scores on the problem.

E. 3-bit multiplier (highest solution quality = BP1, BP2, FS, DSS)

Tables 4.77 and 4.78 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.77: 3-bit multiplier: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.93	0.93	0.96	0.95	0.95	0.92	0.74
μ	0.89	0.90	0.92	0.93	0.93	0.88	0.71
σ	0.02	0.02	0.01	0.01	0.01	0.02	0.02

TABLE 4.78: 3-bit multiplier: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.92	0.89	0.90	0.89	0.86	0.89	0.77
μ	0.82	0.87	0.86	0.80	0.79	0.83	0.68
σ	0.06	0.02	0.04	0.03	0.03	0.03	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.77 and 4.78. Tables 4.79 and 4.80 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.79: 3-bit multiplier: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.10						
BP2	0.00	0.04					
FS	0.00	0.00	0.19				
DSS	0.00	0.00	0.17	0.35			
HP	0.09	0.02	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.80: 3-bit multiplier: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.02	0.37					
FS	0.01	0.00	0.00				
DSS	0.01	0.00	0.00	0.34			
HP	0.32	0.01	0.03	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that BP2, FS and DSS achieve significantly higher training scores than the other fitness measures on the problem. Also, BP1 and BP2 achieve significantly higher test scores than the other fitness measures on the problem.

F. 4-bit multiplier (highest solution quality = OF, BP1, BP2, FS, DSS, HP)

Tables 4.81 and 4.82 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.81: 4-bit multiplier: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.88	0.86	0.88	0.87	0.87	0.85	0.68
μ	0.85	0.84	0.85	0.84	0.84	0.81	0.66
σ	0.06	0.07	0.02	0.03	0.06	0.04	0.06

TABLE 4.82: 4-bit multiplier: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.80	0.81	0.77	0.84	0.81	0.84	0.66
μ	0.75	0.76	0.74	0.80	0.76	0.80	0.63
σ	0.07	0.05	0.02	0.03	0.04	0.04	0.06

Statistical tests are conducted to verify the significance of the observations in tables 4.81 and 4.82. Tables 4.83 and 4.84 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.83: 4-bit multiplier: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.29						
BP2	0.27	0.32					
FS	0.23	0.29	0.27				
DSS	0.18	0.17	0.29	0.22			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.84: 4-bit multiplier: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.27						
BP2	0.29	0.09					
FS	0.03	0.00	0.00				
DSS	0.17	0.30	0.11	0.04			
HP	0.00	0.00	0.00	0.11	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that OF, BP1, BP2, FS and DSS achieve on par and higher training scores compared to HP and NS1. Also, FS and HP achieve significantly higher test scores than the other fitness measures on the problem.

Path-finding benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.10 and 4.11 indicate the mean final BS-OF scores achieved on the training and test sets respectively. The mean final BS-OF scores are also listed in tables 4.85 and 4.86, where the best scores on each problem are highlighted. In turn, figure 4.12 shows the mean generational BS-OF scores achieved on the training set. In the figures and tables below, higher scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures generally achieve high training and test scores on the trivial ant problem, and lower scores on the more difficult tart. and dec. tart. problems. Critically, different fitness measures achieve the best result on the different problems. For example, BP1 and BP2 achieve the highest training and test scores on the dec. tart. problem, but achieve only moderate training and test scores on the tart. problem. Conversely, NS2 achieves the highest training and test scores on the tart. problem, but achieves only moderate training and test scores on the dec. tart. problem. Overall, the performance of a given fitness measure is shown to depend on the particular problem being tackled.

TABLE 4.85: Path-finding benchmarks: Mean final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	0.98	0.98	0.98	0.95	0.98	0.96	0.95	0.98
Tart.	0.51	0.52	0.51	0.14	0.57	0.48	0.14	0.59
Dec. tart.	0.49	0.66	0.67	0.46	0.50	0.48	0.46	0.56

TABLE 4.86: Path-finding benchmarks: Mean final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	0.90	0.90	0.90	0.90	0.92	0.90	0.89	0.90
Tart.	0.39	0.41	0.37	0.11	0.46	0.42	0.11	0.48
Dec. tart.	0.46	0.55	0.58	0.46	0.47	0.45	0.45	0.51

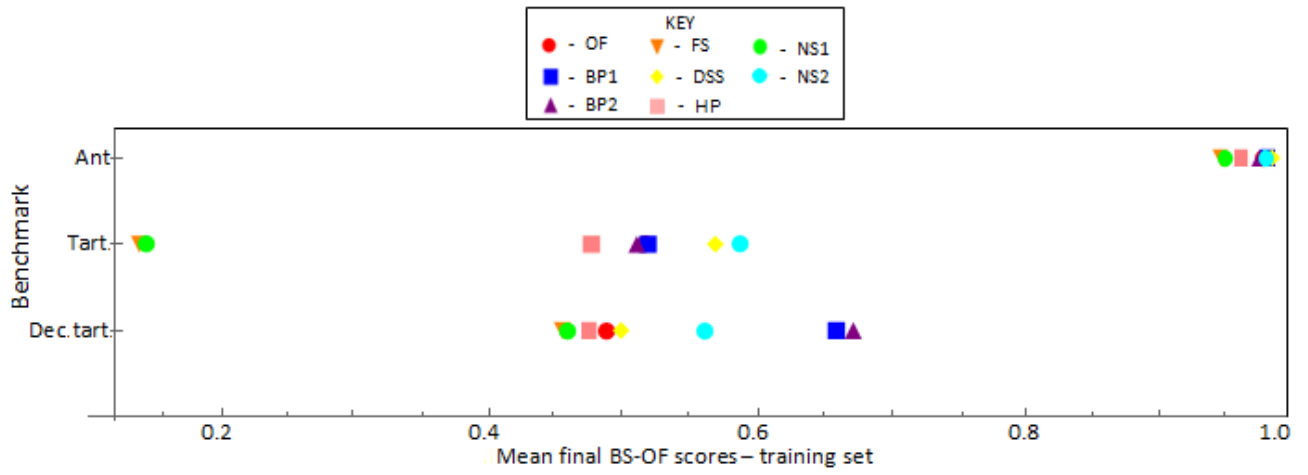


FIGURE 4.10: Path-finding benchmarks: Mean final BS-OF scores - training set

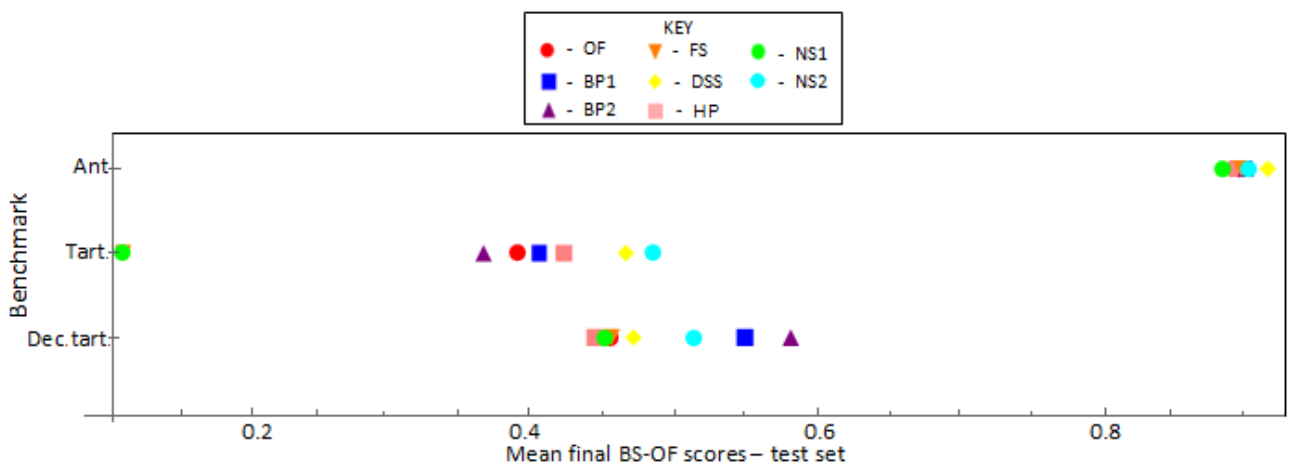


FIGURE 4.11: Path-finding benchmarks: Mean final BS-OF scores - test set

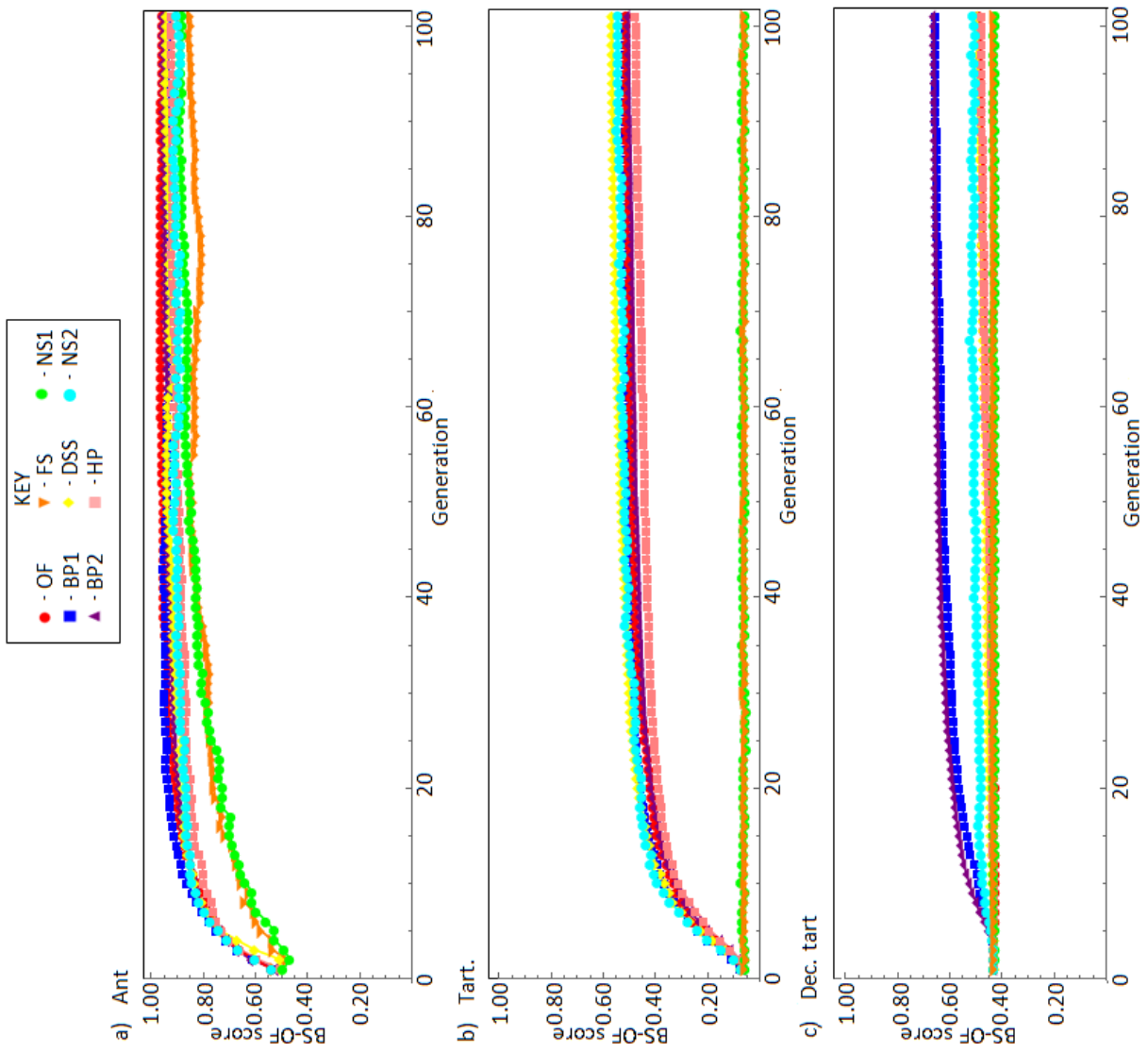


FIGURE 4.12: Path-finding benchmarks: Mean generational BS-OF scores

Detailed results

The benchmarks are discussed in the following order: A) Artificial Ant, B) Tartarus, and C) Deceptive Tartarus.

A. Artificial Ant (highest solution quality = OF, BP1, BP2, DSS, NS2)

Tables 4.87 and 4.88 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables. Note that the mean scores shown in tables 4.87 and 4.88 are the same scores shown for the ant problem in tables 4.85 and 4.86; the analysis in this section looks at the scores achieved on the ant problem in detail.

TABLE 4.87: Artificial Ant: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.50	0.73	0.57	0.57	0.67	0.30	0.40	0.40
b	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
μ	0.98	0.98	0.98	0.95	0.98	0.96	0.95	0.98
σ	0.03	0.03	0.04	0.07	0.03	0.05	0.05	0.05

TABLE 4.88: Artificial Ant: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
b	0.96	0.96	0.96	0.96	0.99	0.95	0.94	1.00
μ	0.90	0.90	0.90	0.90	0.92	0.90	0.89	0.90
σ	0.04	0.05	0.05	0.05	0.04	0.05	0.05	0.05

Statistical tests are conducted to verify the significance of the observations in tables 4.87 and 4.88. Tables 4.89 and 4.90 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.89: Artificial Ant: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.01								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.38							
BP2	0.38	0.29						
FS	0.03	0.03	0.05					
DSS	0.23	0.36	0.16	0.02				
HP	0.03	0.02	0.07	0.23	0.01			
NS1	0.02	0.01	0.03	0.45	0.01	0.24		
NS2	0.38	0.48	0.27	0.02	0.30	0.01	0.01	

TABLE 4.90: Artificial Ant: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.09								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.34							
BP2	0.26	0.43						
FS	0.40	0.44	0.37					
DSS	0.02	0.06	0.07	0.04				
HP	0.45	0.30	0.23	0.36	0.02			
NS1	0.06	0.05	0.02	0.06	0.00	0.08		
NS2	0.25	0.39	0.44	0.33	0.12	0.22	0.04	

The results indicate that OF, BP1, BP2, DSS and NS2 achieve on par training scores; the 5 fitness measures outperform FS, HP and NS1 on the training set. In turn, the ANOVA p-value of 0.09 ($0.09 > \alpha$) in table 4.90 indicates that the fitness measures achieve on par performance on the test set; the ant problem is trivial [232], such that all the fitness measures achieve high quality on the test set.

B. Tartarus (highest solution quality = DSS, NS2)

Tables 4.91 and 4.92 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.91 and 4.92. Tables 4.93 and 4.94 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.91: Tartarus: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.67	0.64	0.66	0.30	0.74	0.65	0.25	0.68
μ	0.51	0.52	0.51	0.14	0.57	0.48	0.14	0.59
σ	0.06	0.09	0.06	0.02	0.06	0.07	0.02	0.04

TABLE 4.93: Tartarus: Statistical tests on final BS-OF scores - training set

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.47							
BP2	0.41	0.35						
FS	0.00	0.00	0.00					
DSS	0.01	0.00	0.00	0.00				
HP	0.07	0.02	0.05	0.00	0.00			
NS1	0.00	0.00	0.00	0.35	0.00	0.00		
NS2	0.00	0.00	0.00	0.00	0.14	0.00	0.00	

TABLE 4.92: Tartarus: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
b	0.69	0.55	0.65	0.25	0.70	0.64	0.34	0.63
μ	0.39	0.41	0.37	0.11	0.46	0.42	0.11	0.48
σ	0.07	0.09	0.07	0.04	0.08	0.08	0.04	0.05

TABLE 4.94: Tartarus: Statistical tests on final BS-OF scores - test set

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.26							
BP2	0.25	0.03						
FS	0.00	0.00	0.00					
DSS	0.02	0.02	0.00	0.00				
HP	0.13	0.22	0.01	0.00	0.03			
NS1	0.00	0.00	0.00	0.48	0.00	0.00		
NS2	0.00	0.00	0.00	0.00	0.28	0.02	0.00	

The results indicate that DSS and NS2 achieve significantly higher training and test scores than the other fitness measures. Conversely FS and NS1 achieve significantly lower training and test scores than the other fitness measures on the problem.

C. Deceptive Tartarus (highest solution quality = BP1, BP2)

Tables 4.95 and 4.96 list the best (b) and mean (μ) final BS-OF scores achieved on the training and test sets respectively; the standard deviation of the final BS-OF scores (σ) is also shown. In addition, the tables list the success rates (S_R) achieved. The best performing fitness measures are highlighted in the tables.

TABLE 4.95: Deceptive Tartarus: Final BS-OF scores - training set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.59	0.76	0.78	0.52	0.65	0.61	0.52	0.68
μ	0.49	0.66	0.67	0.46	0.50	0.48	0.46	0.56
σ	0.04	0.03	0.03	0.02	0.03	0.04	0.02	0.03

TABLE 4.96: Deceptive Tartarus: Final BS-OF scores - test set

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
S_R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
b	0.54	0.69	0.75	0.54	0.58	0.52	0.51	0.64
μ	0.46	0.55	0.58	0.46	0.47	0.45	0.45	0.51
σ	0.05	0.04	0.04	0.03	0.05	0.05	0.02	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.95 and 4.96. Tables 4.97 and 4.98 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that BP1 and BP2 achieve significantly higher training and test scores than the other fitness measures. In turn, FS and NS1 achieve the lowest training scores, and among the lowest test scores.

TABLE 4.97: Deceptive Tartarus:
Statistical tests on final BS-OF
scores - training set

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.29						
FS	0.00	0.00	0.00					
DSS	0.31	0.00	0.00	0.00				
HP	0.24	0.00	0.00	0.01	0.11			
NS1	0.00	0.00	0.00	0.32	0.00	0.03		
NS2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.98: Deceptive Tartarus:
Statistical tests on final BS-OF
scores - test set

One-way ANOVA p-value = 0.04								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.04						
FS	0.49	0.00	0.00					
DSS	0.14	0.00	0.00	0.14				
HP	0.16	0.00	0.00	0.15	0.02			
NS1	0.36	0.00	0.00	0.35	0.06	0.21		
NS2	0.00	0.01	0.00	0.00	0.00	0.00	0.00	

Summary of the solution quality results

The results show that the NFL theorems apply: no one fitness measure achieves the best solution quality on all the tackled problems. The tackled problems differ with respect to the challenges posed to GP. Therefore, different fitness measures are suited to different problems. In some cases, the best performing fitness measure on a problem, A , is the worst performing fitness measure on another problem, B ; the results demonstrate that this can occur even if A and B come from the same problem domain.

While results on training and test sets are included in the study, the following conclusions are based on performance on the test set:

- i) OF, prescribed at the origination of GP, achieves competitive performance on trivial problems, such as the sextic, iris, par-5 and ant problems.
- ii) BP1 and BP2 achieve high performance on a number of the problems, including modular problems. In section 3.3 of chapter 3, and in [1, 158], a modular problem is defined a problem in which the global optima are comprised of useful lower-order functions (or modules) that are clearly delineated from the optima. The sextic problem is modular, because useful lower-order functions, such as x^2 , are reused to construct the sextic function [1]. The Boolean function synthesis problems are also modular: Koza [1] asserts that the Boolean even- n parity functions are characteristically comprised of lower-order parity functions; furthermore, the Boolean 11-multiplexer function is comprised of the lower-order 3-multiplexer and 6-multiplexer functions [1]; also, the data in [257] indicates that the Boolean n -bit multiplier functions are comprised of lower-order functions. Section 3.3.2 of chapter 3 established that BP assesses fitness on both the program and subprogram level, such that the loss of useful subprograms is mitigated. As a result, the subprograms that represent useful modules are retained, which has positive implications for the search efficiency of GP on modular problems. BP2 can improve the search even further, whereby the archive supplied mutation operator is used to substitute the useful subprograms discovered by the search into the candidate solution programs. The above arguments justify the high test scores achieved by BP1 and BP2 on the sextic problem and on the mux-11, mult-3 and mult-4 problems. On the other hand, all the fitness measures are shown to achieve poor test scores on the even- n parity problems due to overfitting, which discussed in detail in section 4.3.2.

BP1 and BP2 also achieve the highest test scores on the deceptive tartarus problem. This version of the tartarus problem is deceptive because the intermediate steps to achieving the goal behavior are penalized. By scoring programs based on useful intermediate results, BP1 and BP2 retain useful modules which would otherwise be discarded by OF measures on the deceptive problem.

- iii) FS also achieves high performance on a number of problems, including problems susceptible to premature convergence. According to the literature [67], the 11-multiplexer problem is highly susceptible to

premature convergence. The wine problem is also susceptible to premature convergence: this is because the wine dataset contains both separable and non-separable classes [245]; when solving classification problems using supervised learning, a separable class is a classification of the fitness cases clearly demarcated by a boundary that separates the predicting attributes of the class from the remaining classes [258]; this property simplifies a classifier's task of identifying all of the fitness cases belonging to the already demarcated class [258]. The wine classification problem is susceptible to premature convergence because GP may converge rapidly to solution programs that only distinguish the separable classes (that is, solution programs that solve only the "easy" fitness cases in the dataset). Section 3.4 of chapter 3 established that FS promotes variety in the fitness cases solved, curtailing rapid convergence to solution programs that solve only the "easy" fitness cases. The results show that FS achieves superior test performance on the wine and 11-multiplexer problems.

- iv) DSS and HP also achieve high performance on a number of problems. Sections 3.5.1 and 3.6.1 of chapter 3 established that DSS and HP increase GP's focus on "difficult" fitness cases, curtailing rapid convergence to solution programs that solve only the "easy" fitness cases. Furthermore, the DSS and HP strategy of resampling the training set during the course of GP mitigates overfitting; hence the expectation is that DSS and HP retain the performance advantage on the test set. DSS and HP are seen to achieve high test performance on the Pagie, wine and 11-multiplexer problems.
- v) NS1 is shown to perform well on the segment problem from the supervised classification domain. Section 3.7 of chapter 3 established that NS is suited to difficult problems, where there is motivation for exploring the behavior space. The segment, vowel and opt problems from the supervised classification domain are part of the CHIRP suite (see [244]), recommended in [232] to pose more of a challenge to GP. Based on the NS1 behavior descriptor defined in table 4.1 of section 4.2.1: given a training set with $|m|$ fitness cases, the behavior space is the space of all possible behavior vectors of length $|m|$ on the tackled problem; hence the size of the behavior space is proportional to $|m|$. Due to a smaller training set (see table 4.4 in section 4.2.3), a smaller behavior space is demarcated on the segment problem compared to the vowel and opt problems. The results obtained from the supervised classification benchmarks show that NS1 is more effective on the segment problem, compared to the vowel and opt problems. This is in line with the literature, whereby NS's reliability is argued to depend on the size of the behavior space [202, 205, 212]; the results indicate that the fitness measure performs well on the segment problem but does not scale well to the larger behavior spaces demarcated on the vowel and opt problems.

NS1 is also shown to perform poorly in the path-finding domain, whereas NS2 performs well in the domain. NS2 performs better than NS1 in the path-finding domain, because former fitness measure applies problem-specific behavior descriptors: each NS2 descriptor takes into account the specific path-finding behavior that is relevant to increasing the OF scores on the problem being tackled. Conversely, the universal behavior descriptor used in NS1 does not incorporate the specific behaviors that can lead to quality improvements in the path-finding problems. The discrepancy in the performance between NS1 and NS2 is justified by the premise that NS's reliability depends on the choice of a behavior descriptor that is relevant to the search objective [202, 211].

4.3.2 Generalization

In the current study, generalization is observed with similar training and test quality scores. Equation 4.10 in section 4.2.2 shows that given a best of generation (or overall best) solution program, the generalization index (GI) is measured as the absolute difference between the OF scores achieved by the solution program on the training and test sets, both of which span the interval $[0, 1]$. Hence the GI in turn spans the interval $[0, 1]$, with lower scores indicating a better result. A presentation of the generalization results from each problem domain ensues.

Symbolic regression benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows. In the detailed results, a comparison is drawn among the fitness measures on each problem; subsequently, statistical tests are conducted to confirm the significance of the observations.

Performance overview

Figure 4.13 shows the mean final BS GI. The mean final BS GI scores are also listed in table 4.99, where the best results achieved on each problem are highlighted. In turn, figure 4.14 shows the mean generational BS GI. In the figures and tables below, lower scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures generalize well on the sextic problem; this follows because the sextic problem is trivial, such that all the fitness measures achieve high training and test quality scores on the problem, as shown in the solution quality results in section 4.3.1. The fitness measures do not generalize as well on the remaining problems. The training and test sets in these problems are deliberately set up to make it difficult for GP to generalize to the test data [232]; this is done so as to determine the approaches that perform as well as possible on difficult problems, rather than simply rewarding good performance on trivial problems [232]. For example, generalization is a challenge on the Nguyen and Pagie problems, due to relatively sparse training sets that do not contain as much information as the test sets with respect to the target function being approximated. In turn, generalization is a challenge on the Keijzer and Vlad problems due to some of the test fitness cases being drawn from outside the training set domain. Furthermore, generalization is a challenge on the Dow problem as a result of the noisy real-world training data. The training and test sets used for the problems are shown in table 4.3 of section 4.2.3.

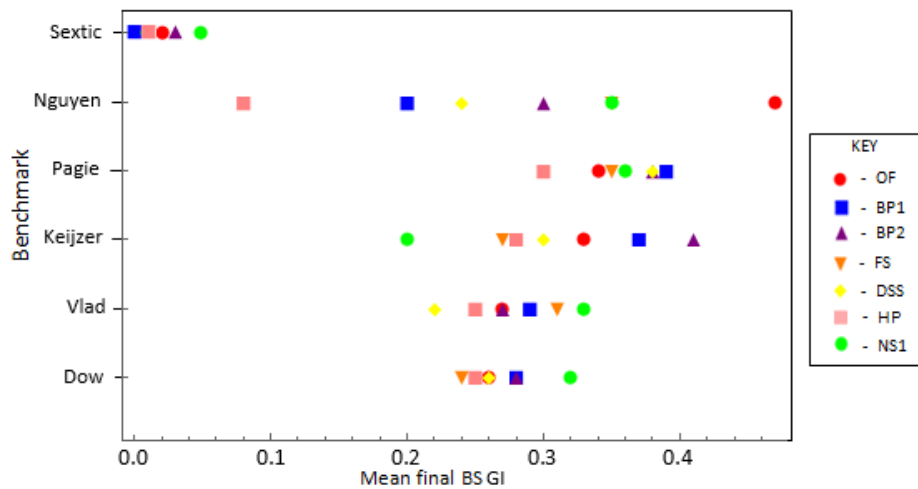


FIGURE 4.13: Symbolic regression benchmarks: Mean final BS GI

TABLE 4.99: Symbolic regression benchmarks: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	0.02	0.00	0.03	0.01	0.01	0.01	0.05
Nguyen	0.47	0.20	0.30	0.35	0.24	0.08	0.35
Pagie	0.34	0.39	0.38	0.35	0.38	0.30	0.36
Keijzer	0.33	0.37	0.41	0.27	0.30	0.28	0.20
Vlad	0.27	0.29	0.27	0.31	0.22	0.25	0.33
Dow	0.26	0.28	0.28	0.24	0.26	0.25	0.32

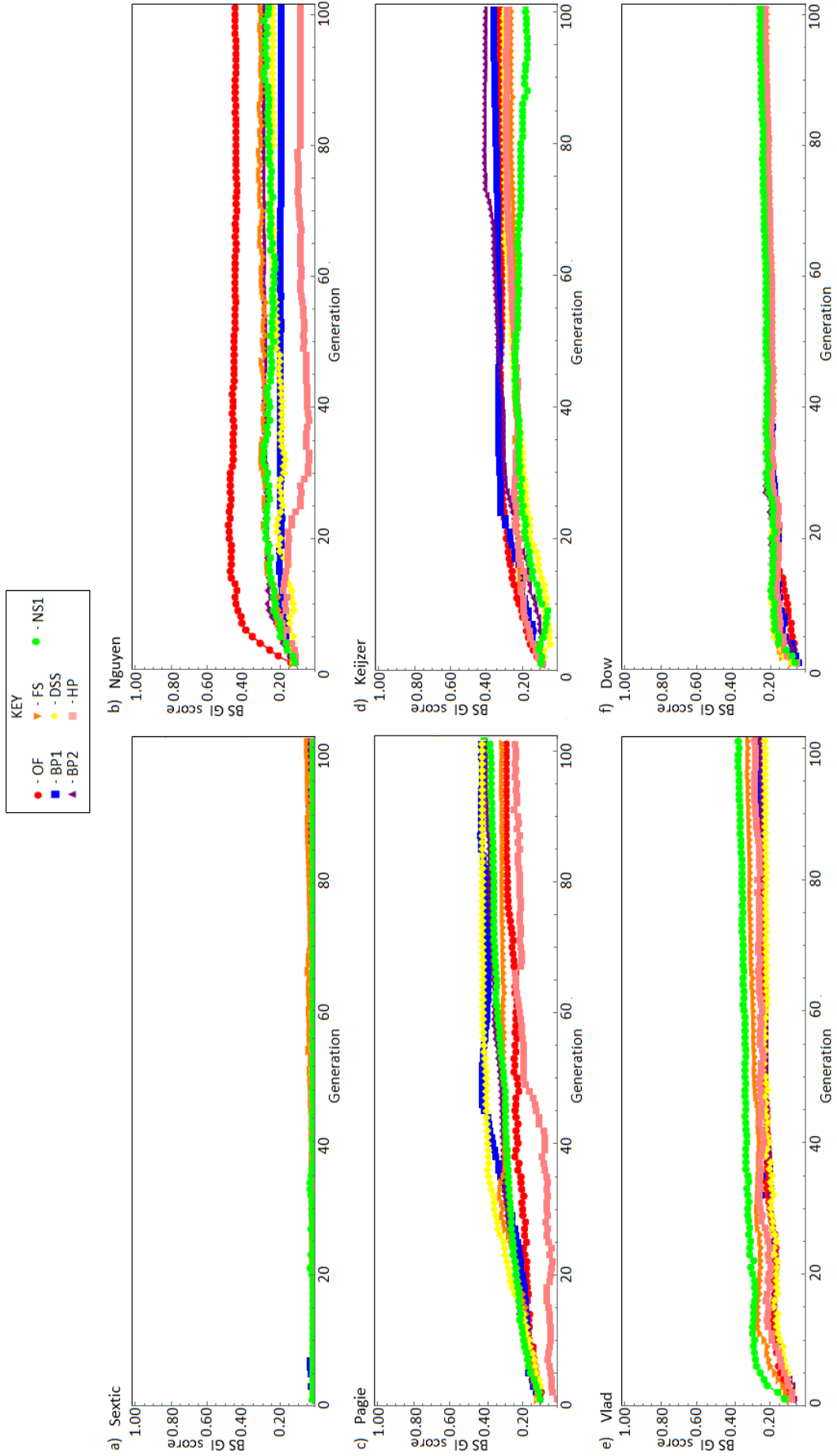


FIGURE 4.14: Symbolic regression benchmarks: Mean generational BS GI

The results show that HP generalizes well on the Nguyen problem, despite the sparse training set used. HP also achieves the best performance on the Pagie problem. Overall, HP is among the best performing fitness measures on most problems, including the Keijzer problem. NS1 appears to achieve the best generalization on the Keijzer problem. Nevertheless, based on the solution quality results shown in section 4.3.1, the low GI score reported for NS1 is a result of the fitness measure achieving low quality scores on the training set, such that the low training scores are similar to the low test scores achieved. HP generalizes better than NS1 because the fitness measure achieves competitive quality scores on both the training and test sets on the problem. FS and DSS also generalize well on the Keijzer problem, whereby the fitness measures achieve competitive scores on both the training and test sets on the problem. A further observation made is the low spread in the GI scores achieved on the Vlad and Dow problems. The solution quality results in section 4.3.1 show that this is due to all the fitness measures achieving low training and test scores on the difficult problems.

Overall, HP is shown to generalize well on the problems in this domain. HP, FS and DSS also generalize better than the other fitness measures on the Keijzer problem.

Detailed results

The benchmarks are discussed in the following order: A) Sextic, B) Nguyen, C) Pagie, D) Keijzer, E) Vlad, and F) Dow.

A. Sextic (best GI = N/A)

Table 4.100 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. Note that the mean scores shown in table 4.100 are the same scores shown for the sextic problem in table 4.99; the analysis in this section looks at the scores achieved on the sextic problem in detail.

TABLE 4.100: Sextic: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.01	0.00	0.01	0.00	0.01	0.00	0.02
μ	0.02	0.00	0.03	0.01	0.01	0.01	0.05
σ	0.01	0.01	0.01	0.01	0.02	0.02	0.01

Statistical tests, similar to the tests done on the solution quality results, are conducted to verify the significance of the observations in table 4.100. Table 4.101 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.101: Sextic: Statistical tests on final BS GI

One-way ANOVA p-value = 0.20							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.05						
BP2	0.17	0.00					
FS	0.15	0.17	0.05				
DSS	0.20	0.19	0.05	0.47			
HP	0.18	0.16	0.06	0.49	0.49		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The ANOVA p-value of 0.20 (whereby $0.20 > \alpha$) in table 4.101 indicates that the fitness measures largely achieve on par generalization. All the fitness measures achieve low GI scores on the trivial sextic problem.

B. Nguyen (best GI = HP)

Table 4.102 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.102: Nguyen: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.40	0.15	0.22	0.27	0.16	0.01	0.24
μ	0.47	0.20	0.30	0.35	0.24	0.08	0.35
σ	0.03	0.02	0.02	0.02	0.03	0.04	0.04

Statistical tests are conducted to verify the significance of the observations in table 4.102. Table 4.103 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.103: Nguyen: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.01	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.38	0.00	0.00	

The results indicate that HP achieves the best generalization on the problem. In turn, OF is the worst performing fitness measure on the problem.

C. Pagie (best GI = HP)

Table 4.104 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.104: Pagie: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.28	0.30	0.30	0.27	0.28	0.19	0.25
μ	0.34	0.39	0.38	0.35	0.38	0.30	0.36
σ	0.02	0.03	0.03	0.03	0.03	0.02	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.104. Table 4.105 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that HP achieves the best generalization on the problem.

TABLE 4.105: Pagie: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.15					
FS	0.19	0.00	0.00				
DSS	0.00	0.17	0.40	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.05	0.00	0.03	0.19	0.03	0.00	

D. Keijzer (best GI = NS1)

Table 4.106 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.106: Keijzer: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.28	0.29	0.30	0.19	0.22	0.21	0.12
μ	0.33	0.37	0.41	0.27	0.30	0.28	0.20
σ	0.03	0.05	0.04	0.03	0.02	0.02	0.04

Statistical tests are conducted to verify the significance of the observations in table 4.106. Table 4.107 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.107: Keijzer: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.17	0.05		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 achieves the lowest GI scores on the problem.

E. Vlad (best GI = DSS)

Table 4.108 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.108: Vlad: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.20	0.21	0.20	0.22	0.15	0.17	0.23
μ	0.27	0.29	0.27	0.31	0.22	0.25	0.33
σ	0.02	0.04	0.03	0.03	0.01	0.02	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.108. Table 4.109 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.109: Vlad: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.01						
BP2	0.49	0.02					
FS	0.00	0.01	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.03	0.00	0.05	0.00	0.01		
NS1	0.00	0.00	0.00	0.05	0.00	0.00	

The results indicate that DSS achieves the lowest GI scores on the problem.

F. Dow (best GI = OF, FS, DSS, HP)

Table 4.110 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.110: Dow: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.18	0.20	0.18	0.15	0.17	0.18	0.25
μ	0.26	0.28	0.28	0.24	0.26	0.25	0.32
σ	0.03	0.04	0.05	0.05	0.05	0.03	0.06

Statistical tests are conducted to verify the significance of the observations in table 4.110. Table 4.111 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.111: Dow: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.05						
BP2	0.05	0.47					
FS	0.06	0.00	0.00				
DSS	0.37	0.05	0.06	0.05			
HP	0.20	0.00	0.00	0.23	0.19		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that a number of the fitness measures achieve on par generalization. Nevertheless, OF, FS, DSS and HP largely achieve better generalization compared to the remaining fitness measures on the problem.

Supervised classification benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.15 shows the mean final BS GI. The mean final BS GI scores are also listed in table 4.112, where the best results achieved on each problem are highlighted. In turn, figure 4.16 shows the mean generational BS GI. In the figures and tables below, lower scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures largely achieve better generalization on the problems in this domain compared to the previous domain. The fitness measures generalize well on the iris problem; this follows because the iris problem is trivial, such that all the fitness measures achieve high training and test quality scores on the problem, as shown in the solution quality results in section 4.3.1. The fitness measures also achieve low GI scores on the opt problem; however, the results in section 4.3.1 show that this is due to low quality scores achieved on both the training and test sets on the difficult opt problem.

NS1 appears to achieve the best generalization on the iris problem. Nevertheless, based on the solution quality results shown in section 4.3.1, the low GI score reported for NS1 occurs due the fitness measure achieving relatively low quality scores on the training set, such that the lower training scores are similar to the low test scores achieved. On the other hand, HP achieves the best generalization on the credit problem, whereby the low GI score shown is a result of high quality scores achieved on both the training and test sets. In turn, DSS achieves the best generalization on the wine problem, with high scores also achieved on both the training and test sets. FS achieves the best generalization on the vowel problem. The solution quality results in section 4.3.1 show that when compared to the other fitness measures, FS achieves competitive quality scores on the training set, and the highest test scores on the problem.

Overall, the fitness measures are shown to generalize well on the supervised classification benchmarks, with FS, DSS and HP achieving superior performance on the vowel, wine and credit problems respectively.

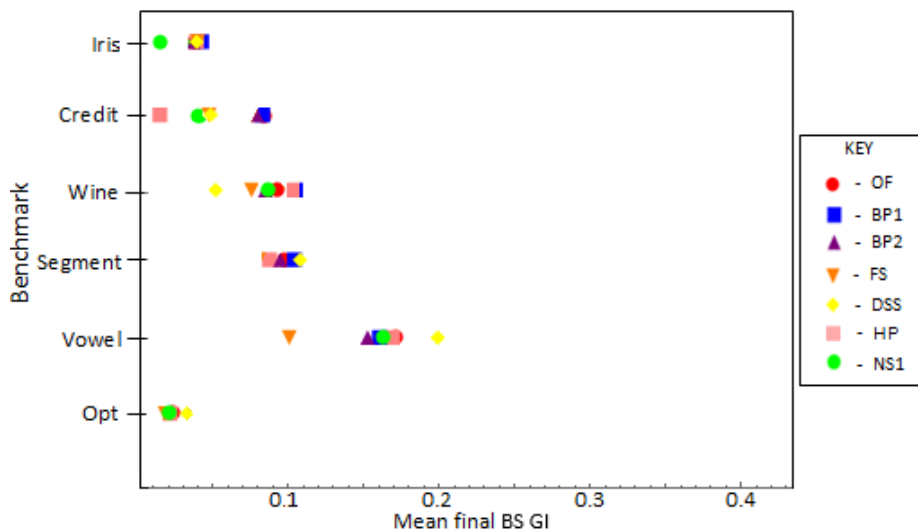


FIGURE 4.15: Supervised classification benchmarks: Mean final BS GI

TABLE 4.112: Supervised classification benchmarks: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	0.04	0.05	0.04	0.04	0.04	0.04	0.01
Credit	0.08	0.08	0.08	0.05	0.05	0.01	0.04
Wine	0.09	0.11	0.08	0.07	0.05	0.11	0.09
Segment	0.10	0.10	0.09	0.11	0.12	0.09	0.11
Vowel	0.17	0.16	0.15	0.10	0.21	0.17	0.16
Opt	0.02	0.01	0.01	0.01	0.03	0.02	0.02

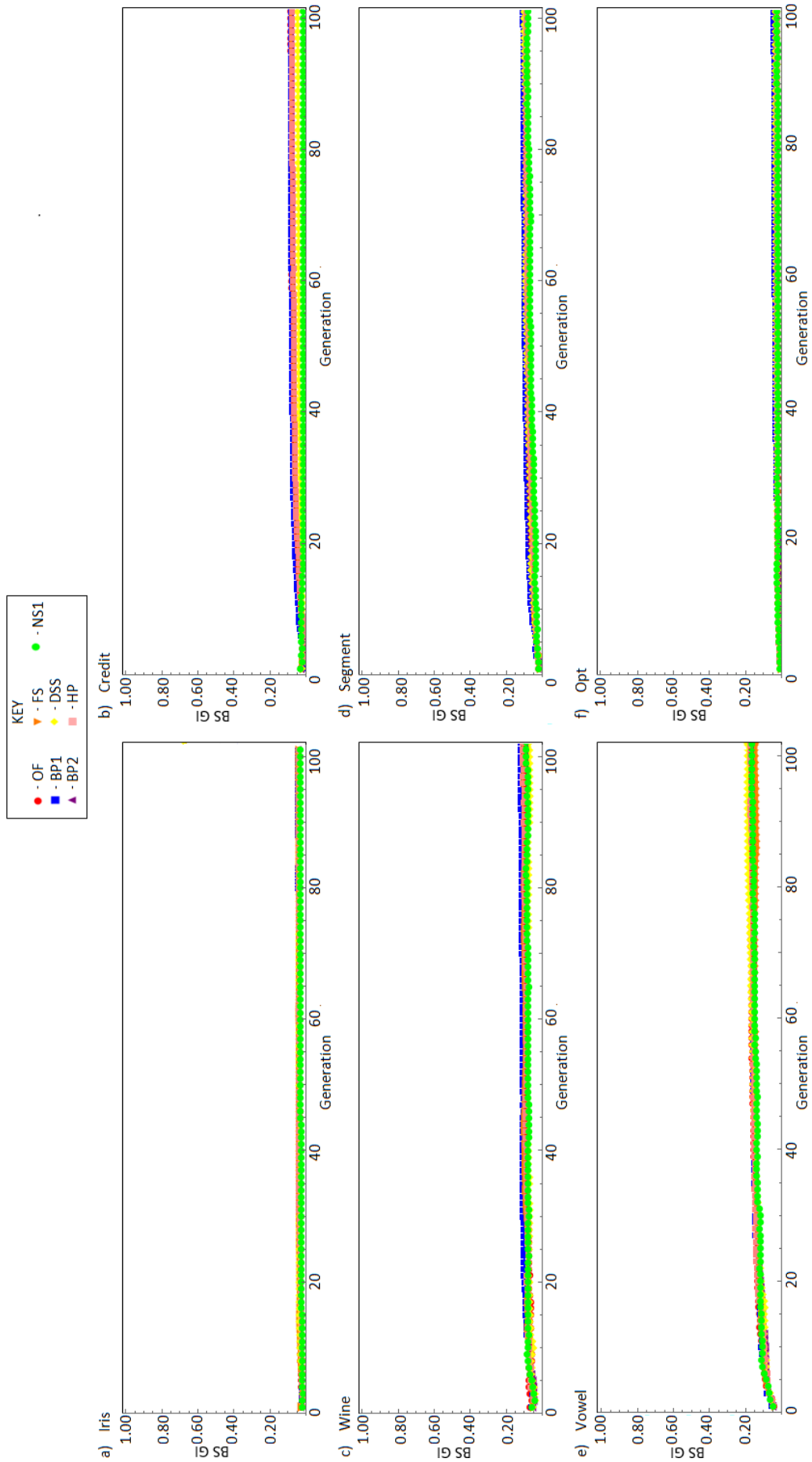


FIGURE 4.16: Supervised classification benchmarks: Mean generational BS GI

Detailed results

The benchmarks are discussed in the following order: A) Iris, B) Credit, C) Wine, D) Segment, E) Vowel and, F) Opt.

A. Iris (best GI = NS1)

Table 4.113 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown.

TABLE 4.113: Iris: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.00	0.01	0.01	0.02	0.01	0.01	0.00
μ	0.04	0.05	0.04	0.04	0.04	0.04	0.01
σ	0.02	0.02	0.02	0.01	0.01	0.02	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.113. Table 4.114 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. $p\text{-val} < 0.05$) are highlighted in the table.

TABLE 4.114: Iris: Statistical tests on final BS GI

One-way ANOVA p-value = 0.21							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.18						
BP2	0.40	0.13					
FS	0.29	0.29	0.19				
DSS	0.35	0.25	0.24	0.42			
HP	0.38	0.24	0.28	0.40	0.48		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results show that NS1 achieves significantly lower GI scores compared to all the other fitness measures on the problem. Apart from NS1, the ANOVA p-value of 0.21 (whereby $0.21 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.114 indicate that the remaining fitness measures achieve on par generalization on this trivial problem.

B. Credit (best GI = HP)

Table 4.115 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.115: Credit: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.05	0.04	0.03	0.00	0.00	0.00	0.02
μ	0.08	0.08	0.08	0.05	0.05	0.01	0.04
σ	0.04	0.03	0.04	0.04	0.03	0.01	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.115. Table 4.116 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. $p\text{-val} < 0.05$) are highlighted in the table.

The results indicate that HP achieves the best generalization on the problem. HP achieves significantly lower GI scores than the other fitness measures on the problem.

TABLE 4.116: Credit: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.43						
BP2	0.20	0.25					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.24			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.13	0.16	0.00	

C. Wine (best GI = DSS)

Table 4.117 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.117: Wine: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.03	0.02	0.02	0.01	0.00	0.01	0.01
μ	0.09	0.11	0.08	0.07	0.05	0.11	0.09
σ	0.03	0.04	0.03	0.02	0.02	0.02	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.117. Table 4.118 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.118: Wine: Statistical tests on final BS GI

One-way ANOVA p-value = 0.03							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.05						
BP2	0.28	0.02					
FS	0.27	0.12	0.11				
DSS	0.03	0.00	0.04	0.05			
HP	0.20	0.23	0.09	0.37	0.00		
NS1	0.35	0.04	0.45	0.18	0.00	0.14	

The results indicate that DSS achieves the best generalization. FS achieves on par generalization to DSS. Otherwise DSS generalizes better than the other fitness measures on the problem.

D. Segment (best GI = N/A)

Table 4.119 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

Statistical tests are conducted to verify the significance of the observations in table 4.119. Table 4.120 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.119: Segment: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.05	0.03	0.03	0.02	0.04	0.01	0.03
μ	0.10	0.10	0.09	0.11	0.12	0.09	0.11
σ	0.06	0.04	0.06	0.05	0.04	0.04	0.04

TABLE 4.120: Segment: Statistical tests on final BS GI

One-way ANOVA p-value = 0.42							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.23						
BP2	0.40	0.18					
FS	0.05	0.07	0.08				
DSS	0.38	0.33	0.29	0.29			
HP	0.13	0.05	0.21	0.22	0.08		
NS1	0.05	0.07	0.06	0.39	0.31	0.12	

The ANOVA p-value of 0.42 (whereby $0.42 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.120 indicate that the fitness measures achieve on par generalization on the problem.

E. Vowel (best = FS)

Table 4.121 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.121: Vowel: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.09	0.09	0.10	0.03	0.13	0.09	0.08
μ	0.17	0.16	0.15	0.11	0.21	0.17	0.16
σ	0.10	0.09	0.06	0.06	0.05	0.07	0.08

Statistical tests are conducted to verify the significance of the observations in table 4.119. Table 4.120 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.122: Vowel: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.18						
BP2	0.06	0.24					
FS	0.00	0.00	0.00				
DSS	0.09	0.01	0.00	0.00			
HP	0.41	0.20	0.05	0.00	0.03		
NS1	0.22	0.43	0.17	0.01	0.01	0.25	

The results indicate that FS achieves the best generalization on the problem. FS is shown to achieve significantly lower GI scores than the other fitness measures.

F. Opt (best GI = N/A)

Table 4.123 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.123: Opt: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>Max</i>	0.00	0.00	0.00	0.00	0.00	0.01	0.01
μ	0.02	0.01	0.01	0.01	0.03	0.02	0.02
<i>Min</i>	0.01	0.02	0.02	0.01	0.01	0.01	0.01

Statistical tests are conducted to verify the significance of the observations in table 4.123. Table 4.124 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.124: Opt: Statistical tests on final BS GI

One-way ANOVA p-value = 0.57							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.16						
BP2	0.12	0.49					
FS	0.41	0.07	0.10				
DSS	0.33	0.19	0.22	0.22			
HP	0.39	0.11	0.10	0.33	0.29		
NS1	0.39	0.09	0.07	0.39	0.27	0.47	

The ANOVA p-value of 0.57 (whereby $0.57 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.124 indicate that the fitness measures achieve on par generalization on the problem. This is because all the fitness measures achieve low training and test scores on the difficult opt problem.

Boolean function synthesis benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.17 shows the mean final BS GI incurred. The mean final BS GI scores are also listed in table 4.125, where the best outcomes on each problem are highlighted. In turn, figure 4.18 shows the mean generational BS GI. In the figures and tables below, lower scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures achieve poor generalization on the even-n parity problems. From the discussion on the benchmark suite in section 4.2.3, recall that the two-datasets methodology is not the standard practice in the Boolean function synthesis domain. For this reason, the training and test sets used on each Boolean problem were drawn with uniform probability from the complete set of fitness cases. A possible reason for poor generalization on the even-n parity problems is that the complete set of fitness cases on each of these problems contains disparate information, such that partitioning such a set results in dissimilar training and test sets, where the training sets do not sufficiently represent the information contained in the test sets, and vice versa. Poor generalization is inevitable with dissimilar training and test sets. On the other

hand, the results show that the fitness measures generalize better on the mux-11, mult-3 and mult-4 problems, indicative of similar training and test sets on these problems.

NS1 appears to generalize well on the even-n parity and n-bit multiplier problems. Nevertheless, based on the solution quality results shown in section 4.3.1, the low GI scores reported for NS1 occur due to low quality scores achieved on both the training and test sets. In turn, HP achieves the best generalization on the mult-4 problem. The solution quality results in section 4.3.1 show that the low GI score reported for HP is a result of high quality scores achieved on both the training and test sets on the mult-4 problem.

Overall, no one fitness measure achieves the best generalization on all problems. The extent to which a fitness measure can improve on GP's generalization is also influenced by the similarity between the training and the test sets, which differs for the different problems.

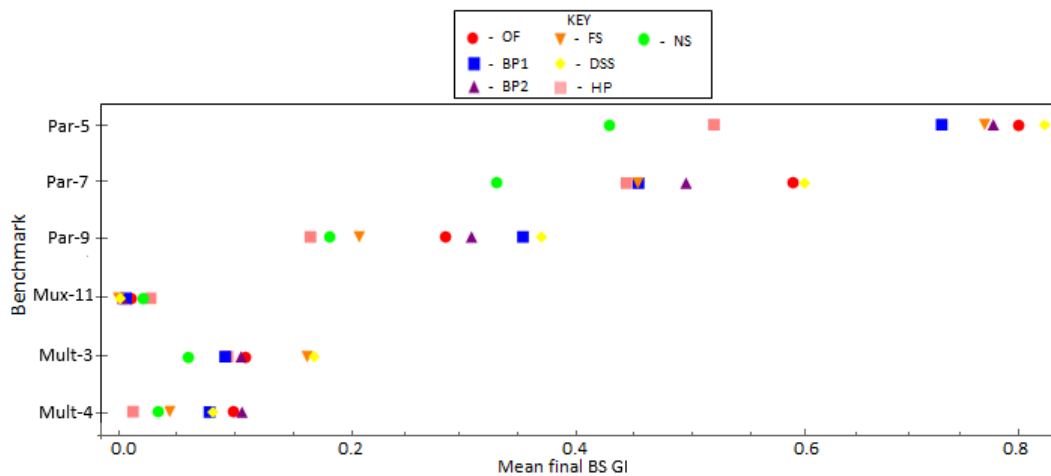


FIGURE 4.17: Boolean function synthesis benchmarks: Mean final BS GI

TABLE 4.125: Boolean function synthesis benchmarks: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	0.80	0.73	0.78	0.77	0.82	0.52	0.43
Par-7	0.59	0.45	0.49	0.45	0.60	0.44	0.33
Par-9	0.29	0.35	0.31	0.21	0.37	0.16	0.18
Mux-11	0.01	0.01	0.00	0.00	0.00	0.03	0.02
Mult-3	0.11	0.09	0.11	0.16	0.17	0.09	0.06
Mult-4	0.10	0.08	0.11	0.04	0.08	0.01	0.03

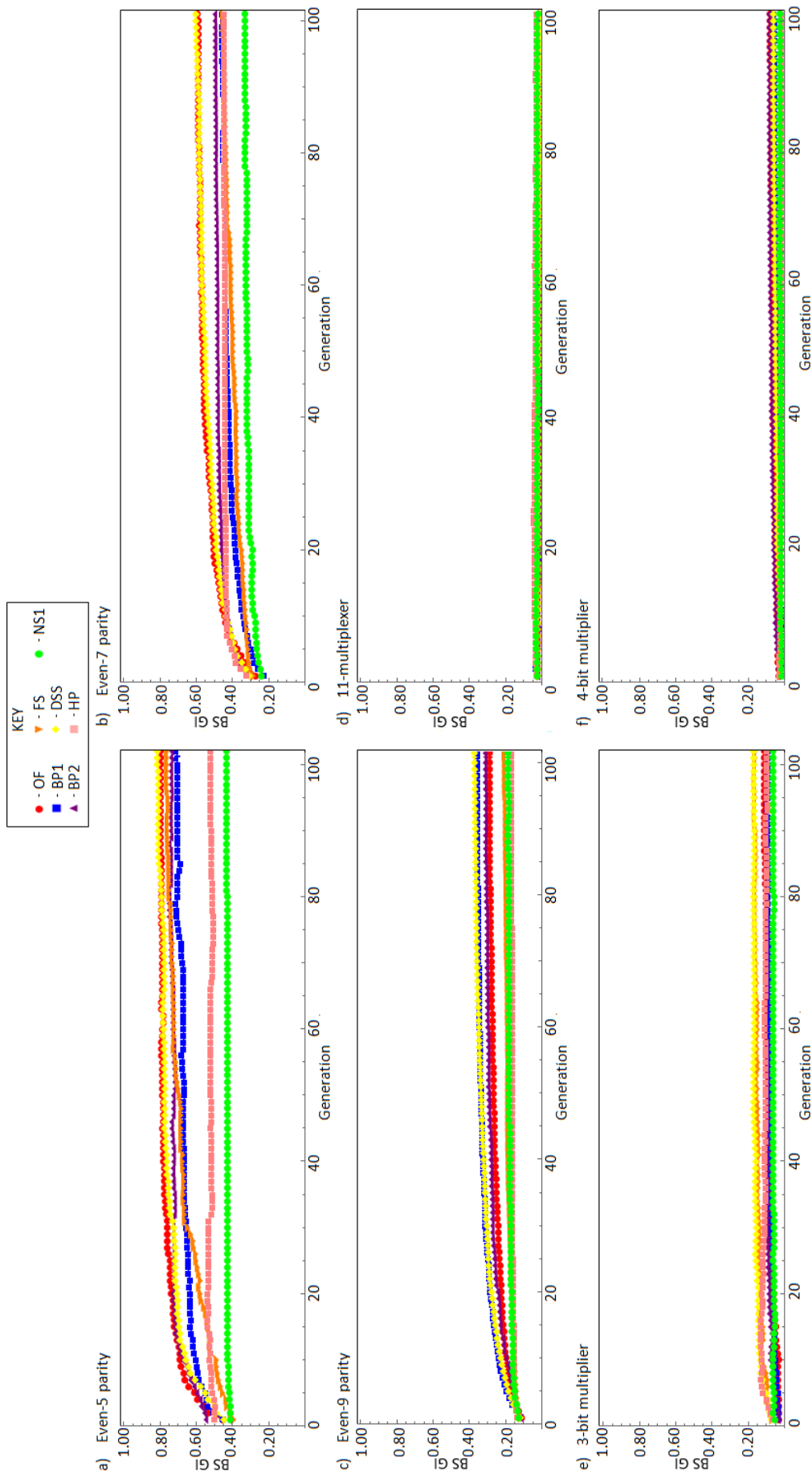


FIGURE 4.18: Boolean function synthesis benchmarks: Mean generational BS GI

Detailed results

The benchmarks are discussed in the following order: A) Even-5 parity, B) Even-7 parity, C) Even-9 parity, D) 11-multiplexer, E) 3-bit multiplier and, F) 4-bit multiplier.

A. Even-5 parity (best GI = NS1)

Table 4.126 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.126: Even-5 parity: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.57	0.45	0.56	0.58	0.54	0.25	0.21
μ	0.80	0.73	0.78	0.77	0.82	0.52	0.43
σ	0.10	0.09	0.09	0.11	0.09	0.13	0.08

Statistical tests are conducted to verify the significance of the observations in table 4.126. Table 4.127 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.127: Even-5 parity: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.02						
BP2	0.21	0.06					
FS	0.15	0.10	0.39				
DSS	0.22	0.00	0.05	0.03			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 achieves the lowest GI scores on the problem.

B. Even-7 parity (best GI = NS1)

Table 4.128 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.128: Even-7 parity: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.34	0.27	0.40	0.38	0.45	0.37	0.20
μ	0.59	0.45	0.49	0.45	0.60	0.44	0.33
σ	0.05	0.03	0.04	0.05	0.04	0.04	0.07

Statistical tests are conducted to verify the significance of the observations in table 4.128. Table 4.129 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.129: Even-7 parity: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.48	0.00				
DSS	0.35	0.00	0.00	0.00			
HP	0.00	0.23	0.00	0.14	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 achieves the lowest GI scores on the problem.

C. Even-9 parity (best GI = HP, NS1)

Table 4.130 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.130: Even-9 parity: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.20	0.29	0.28	0.15	0.28	0.15	0.16
μ	0.29	0.35	0.31	0.21	0.37	0.16	0.18
σ	0.04	0.03	0.04	0.05	0.04	0.02	0.02

Statistical tests are conducted to verify the significance of the observations in table 4.130. Table 4.131 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.131: Even-9 parity: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.17	0.01					
FS	0.00	0.00	0.00				
DSS	0.00	0.09	0.01	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.15	

The results indicate that HP and NS1 achieve the lowest GI scores on the problem.

D. 11-multiplexer (best GI = N/A)

Table 4.132 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

Statistical tests are conducted to verify the significance of the observations in table 4.132. Table 4.133 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.132: 11-multiplexer: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS
b	0.00	0.00	0.00	0.00	0.00	0.00	0.00
μ	0.01	0.01	0.00	0.00	0.00	0.03	0.02
σ	0.01	0.01	0.01	0.01	0.01	0.01	0.01

TABLE 4.133: 11-multiplexer: Statistical tests on final BS GI

One-way ANOVA p-value = 0.20							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS
OF							
BP1	0.49						
BP2	0.22	0.20					
FS	0.23	0.19	0.20				
DSS	0.22	0.13	0.47	0.42			
HP	0.17	0.10	0.14	0.13	0.14		
NS	0.19	0.19	0.16	0.10	0.19	0.27a	

The ANOVA p-value of 0.20 (whereby $0.20 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.133 indicate that the fitness measures achieve on par generalization on the problem.

E. 3-bit multiplier (best GI = NS1)

Table 4.134 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.134: 3-bit multiplier: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.03	0.06	0.03	0.06	0.10	0.03	0.00
μ	0.11	0.09	0.11	0.16	0.17	0.09	0.06
σ	0.05	0.03	0.04	0.03	0.03	0.03	0.05

Statistical tests are conducted to verify the significance of the observations in table 4.134. Table 4.135 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.135: 3-bit multiplier: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.03						
BP2	0.32	0.05					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.29			
HP	0.04	0.37	0.08	0.00	0.00		
NS1	0.00	0.03	0.00	0.00	0.00	0.02	

The results indicate that NS1 achieves the lowest GI scores on the problem.

F. 4-bit multiplier (best GI = HP)

Table 4.136 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.136: 4-bit multiplier: Final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.04	0.03	0.05	0.00	0.02	0.00	0.00
μ	0.10	0.08	0.11	0.04	0.08	0.01	0.03
σ	0.05	0.04	0.02	0.03	0.04	0.04	0.05

Statistical tests are conducted to verify the significance of the observations in table 4.136. Table 4.137 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.137: 4-bit multiplier: Statistical tests on final BS GI

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.12	0.00					
FS	0.00	0.00	0.00				
DSS	0.02	0.24	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.13	0.02	0.00	

The results indicate that HP achieves the lowest GI scores on the problem.

Path-finding benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.19 shows the mean final BS GI. The mean final BS GI scores are also listed in table 4.138, where the best results achieved on each problem are highlighted. In turn, figure 4.20 shows the mean generational BS GI. In the figures and tables below, lower scores indicate better performance of GP. All scores are averaged over the 30 GP runs.

The results show that the fitness measures achieve on par generalization on the artificial ant problem. The ant problem is trivial, such that all the fitness measures achieve high training and test quality scores on the problem, as shown in the solution quality results in section 4.3.1. FS and NS1 appear to generalize well on the tartarus and deceptive tartarus problems. Nevertheless, based on the solution quality results shown in section 4.3.1, the low GI scores reported for FS and NS1 occur due to low training and test scores. DSS and NS2 generalize better than FS and NS1. The solution quality results in section 4.3.1 show that DSS and NS2 achieve the highest quality scores on both the training and test sets on the tartarus problem. HP also generalizes well on the tartarus problem, whereby competitive quality scores are achieved on both the training and test sets. Furthermore, NS2 generalizes well on the deceptive tartarus problem, whereby the fitness measure achieves competitive quality scores on both the training and test sets.

Overall, the fitness measures generalize well on the path-finding tasks. Most of the fitness measures that achieve performance advantages on the training sets maintain the same on the test sets.

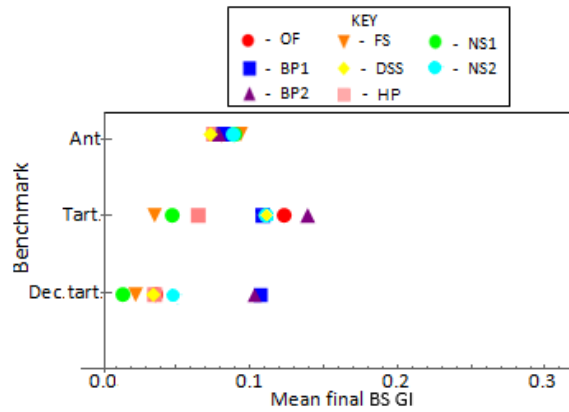


FIGURE 4.19: Path-finding benchmarks: Mean final BS GI

TABLE 4.138: Path-finding benchmarks: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	0.08	0.08	0.08	0.09	0.07	0.08	0.09	0.09
Tart.	0.12	0.11	0.14	0.04	0.11	0.07	0.05	0.11
Dec. tart.	0.04	0.11	0.11	0.02	0.03	0.04	0.01	0.05

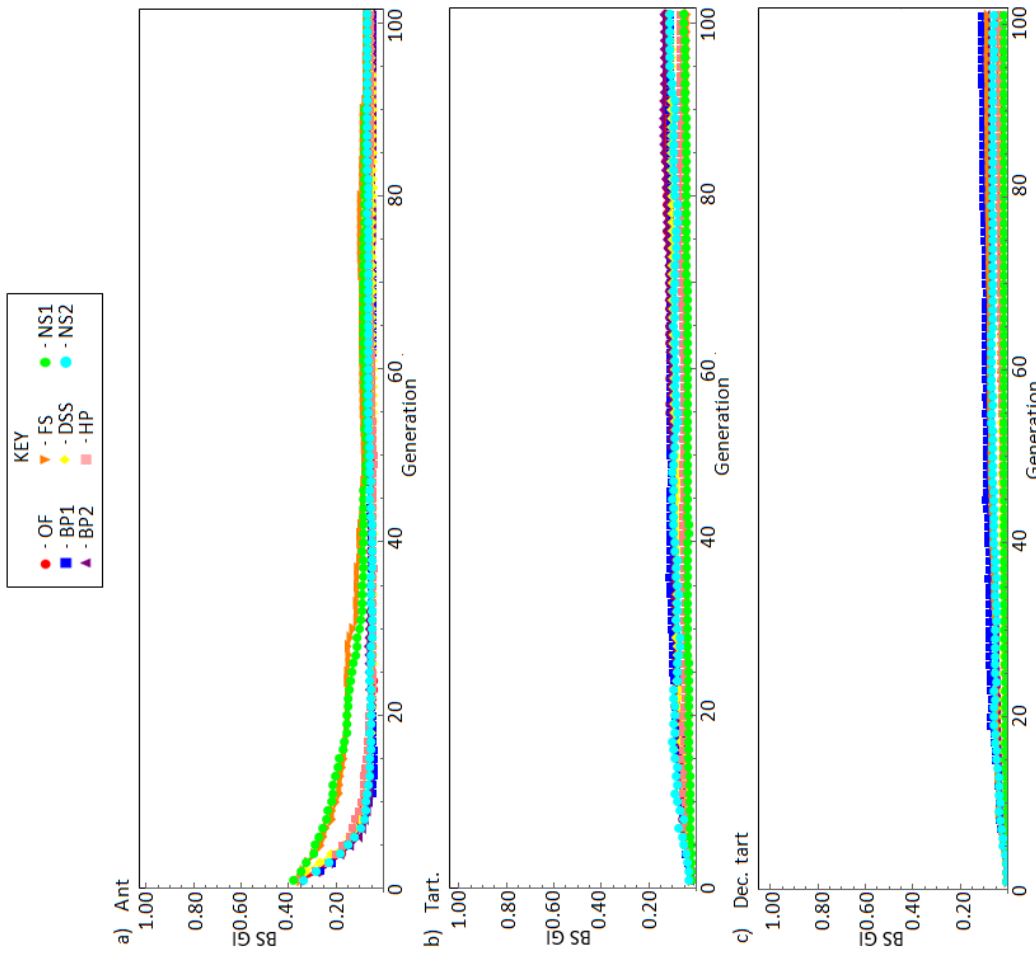


FIGURE 4.20: Path-finding benchmarks: Mean generational BS GI

Detailed results

The benchmarks are discussed in the following order: A) Artificial Ant, B) Tartarus, and C) Deceptive Tartarus.

A. Artificial Ant (best GI = N/A)

Table 4.139 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown.

TABLE 4.139: Artificial Ant: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	0.01	0.01	0.00	0.01	0.01	0.00	0.00	0.00
μ	0.08	0.08	0.08	0.09	0.07	0.08	0.09	0.09
σ	0.02	0.02	0.02	0.04	0.02	0.02	0.02	0.02

Statistical tests are conducted to verify the significance of the observations in table 4.139. Table 4.140 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.140: Artificial Ant: Statistical tests on mean final BS GI

One-way ANOVA p-value = 0.52								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.45							
BP2	0.42	0.38						
FS	0.15	0.18	0.12					
DSS	0.22	0.19	0.31	0.05				
HP	0.30	0.27	0.38	0.08	0.42			
NS1	0.19	0.24	0.16	0.39	0.06	0.10		
NS2	0.24	0.28	0.20	0.36	0.09	0.13	0.46	

The ANOVA p-value of 0.52 (whereby $0.52 > \alpha$) and the p-values resulting from the pairwise comparisons in table 4.140 indicate that the fitness measures achieve on par generalization on the trivial problem.

B. Tartarus (best GI = FS, NS1)

Table 4.141 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.141: Tartarus: Mean final BS GI

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.01
μ	0.12	0.11	0.14	0.04	0.11	0.07	0.05	0.11
σ	0.04	0.03	0.04	0.01	0.04	0.02	0.02	0.04

Statistical tests are conducted to verify the significance of the observations in table 4.141. Table 4.142 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that FS and NS1 achieve the lowest GI scores on the problem.

TABLE 4.142: Tartarus: Statistical tests on mean final BS GI

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.21							
BP2	0.23	0.05						
FS	0.00	0.00	0.00					
DSS	0.23	0.46	0.06	0.00				
HP	0.00	0.00	0.00	0.00	0.00			
NS1	0.00	0.00	0.00	0.04	0.00	0.04		
NS2	0.28	0.45	0.10	0.00	0.48	0.00	0.00	

C. Deceptive tartarus (best GI = FS, NS1)

Table 4.143 lists the best (b) and mean (μ) final BS GI; the standard deviation of the final BS GI (σ) is also shown. The best performing fitness measures are highlighted in the table.

TABLE 4.143: Deceptive Tartarus: Mean BS GI

	OF	BP1	BP2	FS	DSS	HP	NS	NS2
b	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00
μ	0.04	0.11	0.11	0.02	0.03	0.04	0.01	0.05
σ	0.03	0.04	0.04	0.01	0.03	0.03	0.01	0.03

Statistical tests are conducted to verify the significance of the observations in table 4.143. Table 4.144 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.144: Deceptive Tartarus: Statistical tests on mean BS GI

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS	NS2
OF								
BP1	0.00							
BP2	0.00	0.40						
FS	0.03	0.00	0.00					
DSS	0.42	0.00	0.00	0.04				
HP	0.47	0.00	0.00	0.04	0.46			
NS	0.00	0.00	0.00	0.02	0.00	0.00		
NS2	0.02	0.00	0.00	0.00	0.01	0.02	0.00	

The results indicate that FS and NS1 achieve the lowest GI scores on the problem.

Summary of the generalization results

The results indicate that no one fitness measure achieves the best generalization on all benchmarks. The extent to which a fitness measure can improve on GP's generalization is also influenced by the similarity between the training and the test sets, which differs for the different problems. A number of the synthetic problems from the symbolic regression domain were deliberately set up with disparate training and test sets. In addition, the even-n parity problems from the Boolean function synthesis domain were also observed to contain disparate training and test sets. Overall, the aim is to determine the approaches that generalize as well as possible on difficult problems. The following general observations are made:

- i) DSS and HP achieve the best generalization on a number of the problems. Sections 3.5.1 and 3.6.1 of chapter 3 established that DSS and HP repeatedly resample the training set during the course of GP, such that the solution programs being evolved capture the underlying relationships in the training set without overfitting it. DSS generalizes well on the credit, wine and mux-11 problems. HP generalizes well on the Nguyen, Pagie, credit and mult-4 problems.
- ii) FS also demonstrates the capability to improve on GP's generalization. A possible justification for this is that FS mitigates overfitting the training set as a result of the niching strategy used. Section 3.4 of chapter 3 established that the niching strategy promotes diversity in the fitness cases solved by the solution programs; the ability to solve diverse fitness cases improves on the chances of solving unseen fitness cases. FS generalizes the best on the vowel problem. FS also achieves competitive generalization results on the Keijzer and wine problems.
- iii) DSS, HP and FS do not always reliably improve on GP's generalization. The extent of generalization achieved is also influenced by the similarity between the training and the test sets, which differs for the different problems. Poor generalization is inevitable with dissimilar training and test sets. For example, none of the fitness measures generalizes well on the even-n parity problems.

4.3.3 Population diversity

This section reports on the semantic and fitness diversity levels maintained by the fitness measures. Due to the lack of clear correlation between diversity and solution quality, both the lowest and highest diversity scores achieved on each problem are highlighted in the results. A presentation of the results from each domain ensues.

Symbolic regression benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.21 and 4.22 show the mean semantic diversity and entropy of the final GP populations; tables 4.145 and 4.146 contain the same data, whereby the highest and lowest diversity scores are highlighted in the tables. Figures 4.23 and 4.24 show the change in the mean population semantic diversity and the mean population entropy through the generations of GP. The values in the figures and tables below are averaged over the 30 GP runs.

The results show that FS and NS1 exhibit among the highest semantic diversity and entropy scores on most problems in the domain. DSS exhibits relatively high entropy on the sextic problem, and relatively high semantic diversity on the sextic and Nguyen problems. Also, HP exhibits the highest semantic diversity and entropy on the Pagie problem. Conversely, BP1 and BP2 exhibit among the lowest semantic diversity and entropy scores on most problems. Overall, FS, DSS, HP and NS1 are shown to exhibit relatively high diversity while BP1 and BP2 exhibit low diversity.

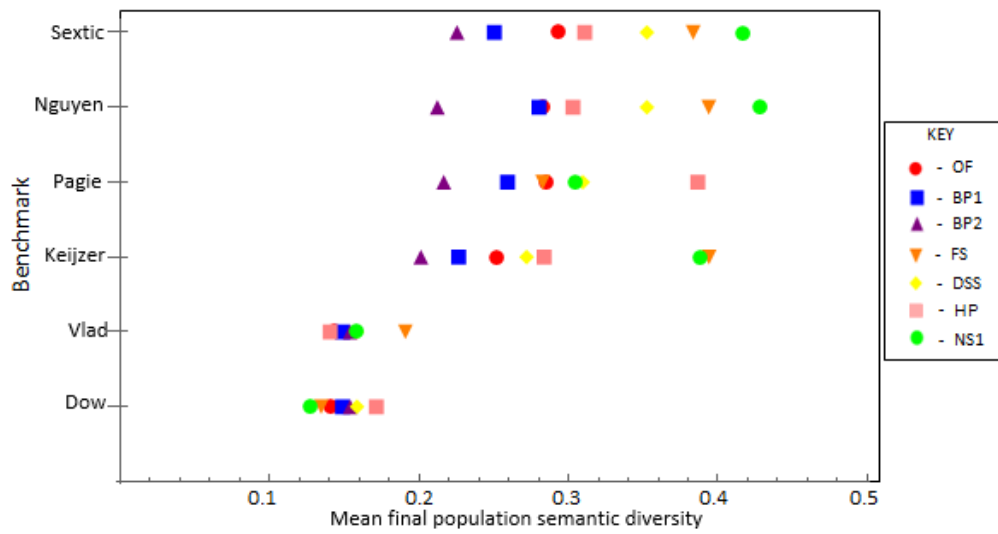


FIGURE 4.21: Symbolic regression benchmarks: Mean final population semantic diversity

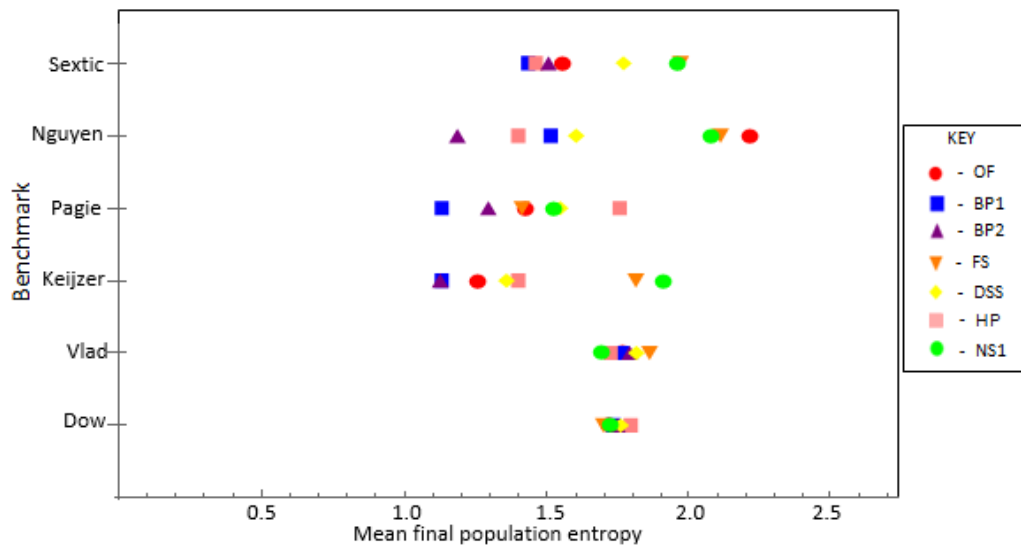


FIGURE 4.22: Symbolic regression benchmarks: Mean final population entropy

TABLE 4.145: Symbolic regression benchmarks: Mean final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	0.29	0.25	0.23	0.38	0.35	0.31	0.43
Nguyen	0.29	0.28	0.21	0.39	0.35	0.30	0.44
Pagie	0.29	0.26	0.22	0.28	0.31	0.39	0.30
Keijzer	0.25	0.23	0.20	0.39	0.27	0.28	0.39
Vlad	0.14	0.15	0.15	0.20	0.16	0.14	0.16
Dow	0.14	0.15	0.15	0.13	0.16	0.17	0.13

TABLE 4.146: Symbolic regression benchmarks: Mean final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	1.55	1.43	1.50	1.97	1.77	1.46	1.96
Nguyen	2.21	1.51	1.19	2.11	1.60	1.40	2.08
Pagie	1.43	1.13	1.29	1.41	1.55	1.75	1.52
Keijzer	1.26	1.13	1.13	1.81	1.36	1.40	1.90
Vlad	0.74	0.74	0.77	0.84	0.79	0.70	0.67
Dow	0.35	0.37	0.38	0.33	0.40	0.43	0.36

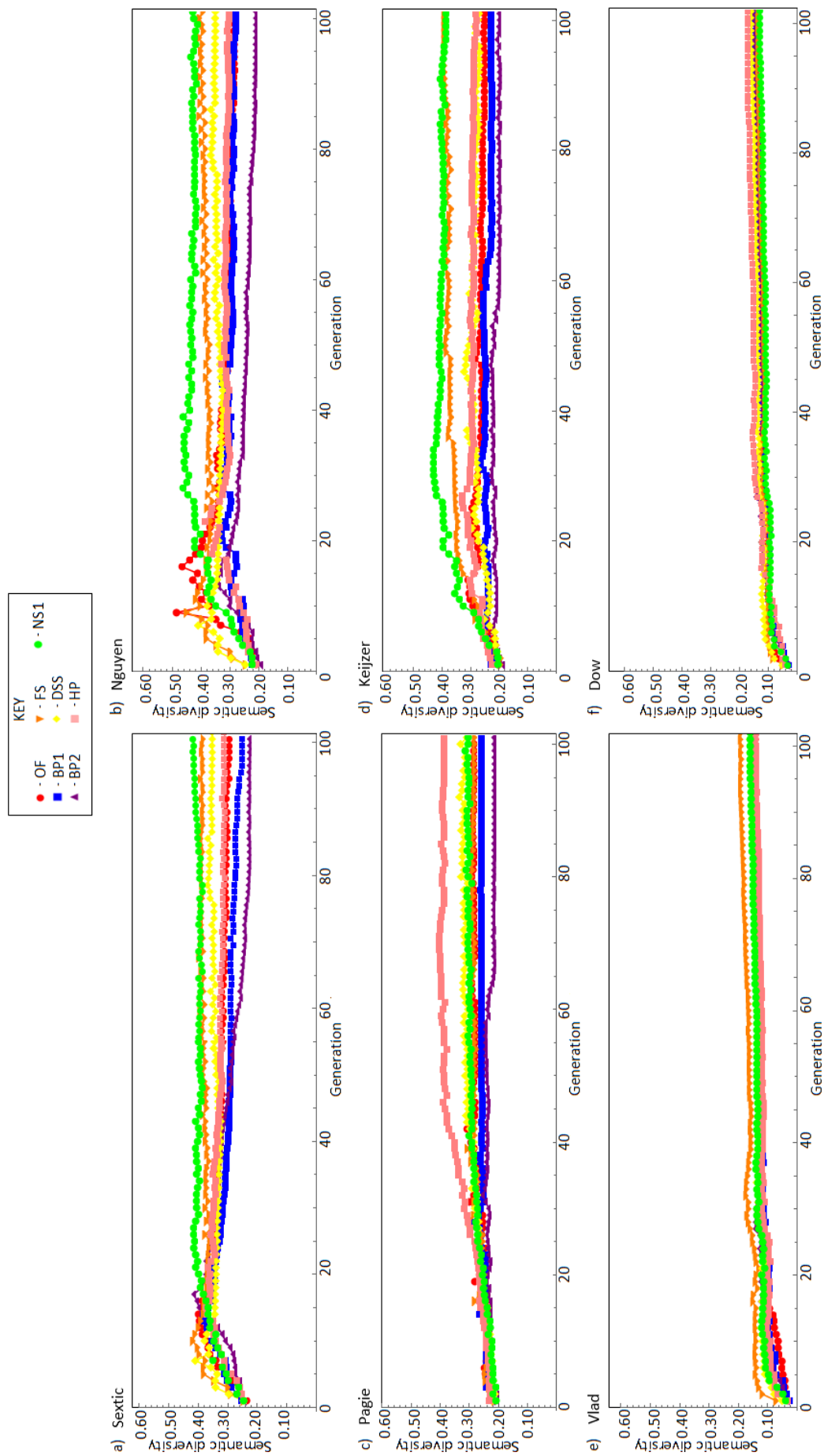


FIGURE 4.23: Symbolic regression benchmarks: Generational population semantic diversity

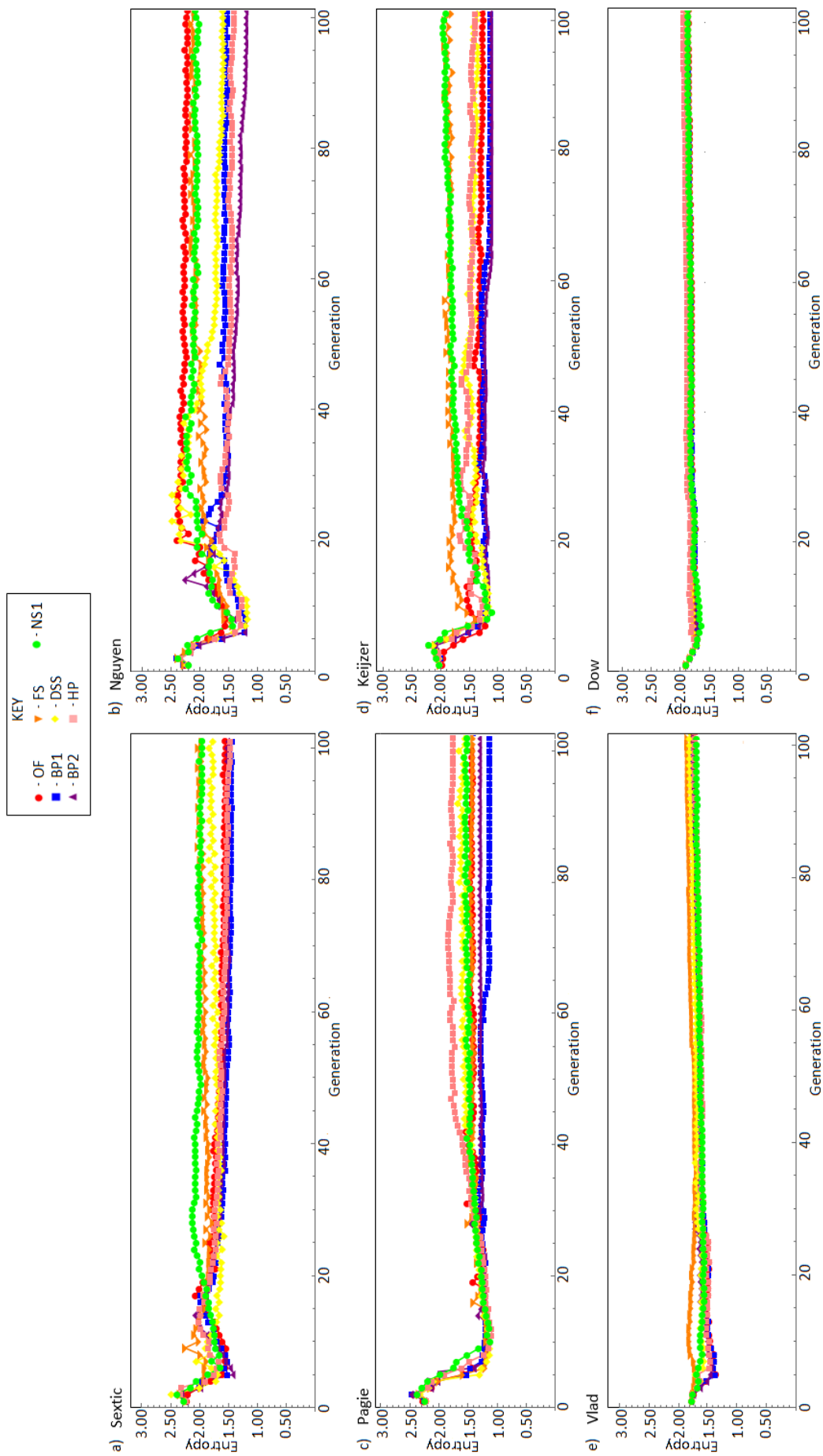


FIGURE 4.24: Symbolic regression benchmarks: Generational population entropy

Detailed results

The benchmarks are discussed in the following order: A) Sextic, B) Nguyen, C) Pagie, D) Keijzer, E) Vlad, and F) Dow.

A. Sextic

(highest semantic diversity = NS1; lowest semantic diversity = BP2; highest entropy = FS, NS1; lowest entropy = OF, BP1, BP2, HP)

Table 4.147 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, table 4.148 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables. Note that the mean scores shown in tables 4.147 and 4.148 are the same scores shown for the sextic problem in tables 4.145 and 4.146 respectively; the analysis in this section looks at the scores achieved on the sextic problem in detail.

TABLE 4.147: Sextic: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.44	0.39	0.36	0.50	0.52	0.45	0.60
μ	0.29	0.25	0.23	0.38	0.35	0.31	0.43
σ	0.05	0.04	0.03	0.04	0.04	0.04	0.03

TABLE 4.148: Sextic: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	2.13	2.05	2.19	2.59	2.25	1.99	2.62
μ	1.55	1.43	1.50	1.97	1.77	1.46	1.96
σ	0.07	0.06	0.06	0.05	0.04	0.04	0.07

Statistical, similar to the tests done on the solution quality results, tests are conducted to verify the significance of the observations in tables 4.147 and 4.148. Tables 4.149 and 4.150 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. $p\text{-value} < 0.05$) are highlighted in the tables.

TABLE 4.149: Sextic: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.05	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.03	0.00	0.00	

TABLE 4.150: Sextic: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.05						
BP2	0.17	0.12					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.06	0.26	0.22	0.00	0.00		
NS1	0.00	0.00	0.00	0.48	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity than the other fitness measures. In turn, FS and NS1 exhibit the highest entropy. Furthermore, BP2 exhibits the lowest semantic diversity, whereas OF, BP1, BP2 and HP exhibit on par entropy on the problem.

B. Nguyen

(highest semantic diversity = NS1; lowest semantic diversity = BP2; highest entropy = OF, FS, NS1; lowest entropy = BP2)

Table 4.151 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.152 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.151: Nguyen: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.44	0.46	0.36	0.53	0.52	0.48	0.56
μ	0.29	0.28	0.21	0.39	0.35	0.30	0.44
σ	0.06	0.06	0.05	0.04	0.05	0.05	0.03

TABLE 4.152: Nguyen: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	2.42	1.78	1.44	2.40	1.81	1.64	2.39
μ	2.21	1.51	1.19	2.11	1.60	1.40	2.08
σ	0.06	0.05	0.06	0.06	0.04	0.04	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.151 and 4.152. Tables 4.153 and 4.154 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.153: Nguyen: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.42						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.154: Nguyen: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.05	0.00	0.00				
DSS	0.00	0.06	0.00	0.00			
HP	0.00	0.05	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.32	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity than the other fitness measures. In turn, OF, FS and NS1 exhibit the highest entropy. Furthermore, BP2 exhibits significantly lower semantic diversity and entropy than the other fitness measures on the problem.

C. Pagie

(highest semantic diversity = HP; lowest semantic diversity = BP2; highest entropy = HP; lowest entropy = BP1)

Table 4.155 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.156 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.155: Pagie: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.45	0.41	0.40	0.44	0.48	0.57	0.52
μ	0.29	0.26	0.22	0.28	0.31	0.39	0.30
σ	0.05	0.07	0.06	0.04	0.05	0.06	0.05

TABLE 4.156: Pagie: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	1.61	1.31	1.44	1.70	1.75	2.05	1.71
μ	1.43	1.13	1.29	1.41	1.55	1.75	1.52
σ	0.06	0.05	0.05	0.07	0.05	0.07	0.05

Statistical tests are conducted to verify the significance of the observations in tables 4.155 and 4.156. Tables 4.157 and 4.158 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that HP exhibits significantly higher semantic diversity and entropy than the other fitness measures. In turn, BP2 exhibits the lowest semantic diversity while BP1 exhibits the lowest entropy.

TABLE 4.157: Pagie: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.46	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.04	0.00	0.00	0.00	0.40	0.00	

TABLE 4.158: Pagie: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.43	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.05	0.00	0.00	0.00	0.46	0.00	

D. Keijzer

(highest semantic diversity = FS, NS1; lowest semantic diversity = BP2; highest entropy = FS, NS1; lowest entropy = BP1, BP2)

Table 4.159 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.160 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.159: Keijzer: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.44	0.44	0.38	0.51	0.52	0.49	0.55
μ	0.25	0.23	0.20	0.39	0.27	0.28	0.39
σ	0.09	0.07	0.06	0.07	0.08	0.07	0.05

TABLE 4.160: Keijzer: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	1.67	1.40	1.39	2.10	1.57	1.89	2.19
μ	1.26	1.13	1.13	1.81	1.36	1.40	1.90
σ	0.10	0.09	0.07	0.07	0.09	0.11	0.07

Statistical tests are conducted to verify the significance of the observations in tables 4.159 and 4.160. Tables 4.161 and 4.162 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.161: Keijzer: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.05		
NS1	0.00	0.00	0.00	0.36	0.00	0.00	

TABLE 4.162: Keijzer: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.49					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.23		
NS1	0.00	0.00	0.00	0.05	0.00	0.00	

The results indicate that FS and NS1 exhibit significantly higher semantic diversity and entropy than the other fitness measures. In turn, BP2 exhibits the lowest semantic diversity, while BP1 and BP2 exhibit the lowest entropy.

E. Vlad

(highest semantic diversity = FS; lowest semantic diversity = N/A; highest entropy = N/A; lowest entropy = N/A)

Table 4.163 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.164 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.163: Vlad: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.25	0.25	0.25	0.35	0.30	0.27	0.22
μ	0.14	0.15	0.15	0.20	0.16	0.14	0.16
σ	0.03	0.03	0.03	0.04	0.04	0.03	0.02

TABLE 4.164: Vlad: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.89	0.87	0.90	0.99	0.94	0.85	0.75
μ	0.74	0.74	0.77	0.84	0.79	0.70	0.67
σ	0.04	0.04	0.03	0.05	0.04	0.04	0.03

Statistical tests are conducted to verify the significance of the observations in tables 4.163 and 4.164. Tables 4.165 and 4.166 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.165: Vlad: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.47							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.47						
BP2	0.46	0.50					
FS	0.00	0.00	0.00				
DSS	0.39	0.41	0.34	0.00			
HP	0.49	0.42	0.37	0.00	0.25		
NS1	0.30	0.41	0.39	0.00	0.27	0.20	

TABLE 4.166: Vlad: Statistical tests on final population entropy

One-way ANOVA p-value = 0.52							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.50						
BP2	0.47	0.41					
FS	0.04	0.04	0.05				
DSS	0.26	0.26	0.32	0.05			
HP	0.25	0.25	0.17	0.00	0.03		
NS1	0.05	0.05	0.03	0.00	0.00	0.29	

The results indicate that FS exhibits significantly higher semantic diversity than the other fitness measures. Apart from FS, the remaining fitness measures exhibit on par semantic diversity. In turn, FS exhibits significantly higher entropy than OF, BP1, HP and NS1. Otherwise, the fitness measures largely exhibit on par entropy on the problem.

F. Dow

(highest semantic diversity = N/A; lowest semantic diversity = N/A; highest entropy = N/A; lowest entropy = N/A)

Table 4.167 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.168 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.167: Dow: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.27	0.27	0.27	0.22	0.29	0.29	0.22
μ	0.14	0.15	0.15	0.13	0.16	0.17	0.13
σ	0.03	0.03	0.03	0.03	0.04	0.04	0.03

TABLE 4.168: Dow: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	1.61	1.31	1.44	1.70	1.75	2.05	1.71
μ	0.35	0.37	0.38	0.33	0.40	0.43	0.36
σ	0.06	0.05	0.05	0.07	0.05	0.07	0.05

Statistical tests are conducted to verify the significance of the observations in tables 4.167 and 4.168. Tables 4.169 and 4.170 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.169: Dow: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.77							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.47						
BP2	0.47	0.50					
FS	0.45	0.40	0.40				
DSS	0.41	0.45	0.45	0.47			
HP	0.37	0.42	0.42	0.37	0.40		
NS1	0.45	0.40	0.40	0.50	0.40	0.37	

TABLE 4.170: Dow: Statistical tests on final population entropy

One-way ANOVA p-value = 0.79							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.47						
BP2	0.46	0.49					
FS	0.41	0.40	0.39				
DSS	0.41	0.45	0.47	0.39			
HP	0.33	0.40	0.42	0.37	0.46		
NS1	0.48	0.47	0.47	0.47	0.40	0.38	

The results indicate that the fitness measures exhibit on par semantic diversity and entropy on the difficult Dow problem.

Supervised classification benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.25 and 4.26 show the mean semantic diversity and entropy of the final GP populations; tables 4.171 and 4.172 contain the same data, whereby the highest and lowest diversity scores are highlighted in the tables. Figures 4.27 and 4.28 show the change in the mean population semantic diversity and the mean population entropy through the generations of GP. The values in the figures and tables below are averaged over the 30 GP runs.

As in the previous domain, the results show that FS and NS1 exhibit among the highest semantic diversity and entropy scores on most problems in the domain. OF exhibits the highest entropy on the credit problem, while DSS exhibits the highest entropy on the wine problem. In turn, BP1 and BP2 exhibit the lowest semantic diversity and entropy scores on most problems in the domain. Also, HP exhibits the lowest entropy on the credit problem. Overall, FS, DSS and NS1 are shown to exhibit relatively high diversity, while BP1 and BP2 exhibit low diversity.

TABLE 4.171: Supervised classification benchmarks: Mean final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	0.21	0.11	0.12	0.20	0.19	0.16	0.53
Credit	0.07	0.02	0.02	0.19	0.23	0.19	0.50
Wine	0.18	0.10	0.13	0.23	0.28	0.17	0.38
Segment	0.10	0.05	0.05	0.37	0.22	0.21	0.42
Vowel	0.09	0.03	0.02	0.35	0.13	0.16	0.36
Opt	0.08	0.01	0.01	0.29	0.13	0.15	0.29

TABLE 4.172: Supervised classification benchmarks: Mean final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	3.15	2.39	2.50	3.09	3.23	2.81	3.91
Credit	3.33	1.94	1.80	1.61	1.44	1.08	2.35
Wine	3.19	2.71	2.81	3.40	3.60	3.09	3.38
Segment	3.26	2.52	2.30	3.82	3.45	3.46	4.04
Vowel	3.86	1.92	1.48	4.67	3.54	3.24	4.77
Opt	4.49	2.05	1.89	5.56	5.13	4.76	5.61

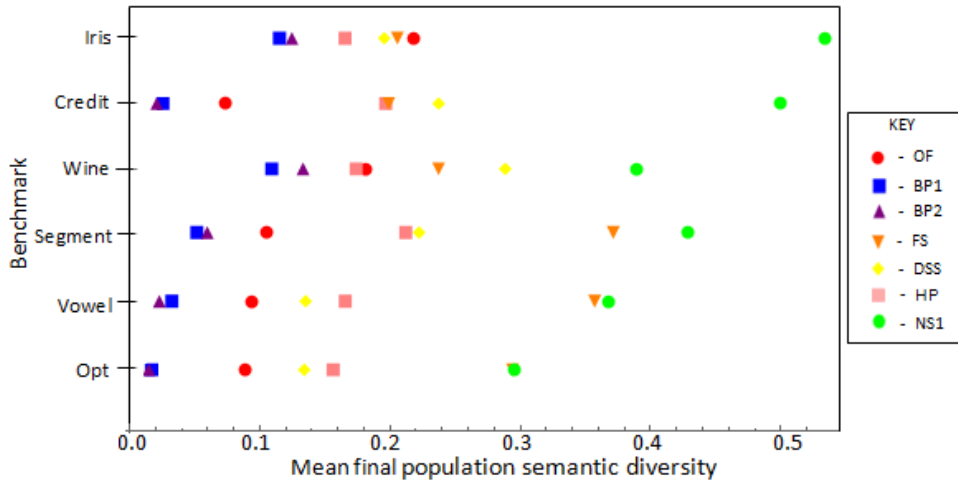


FIGURE 4.25: Supervised classification benchmarks: Mean final population semantic diversity

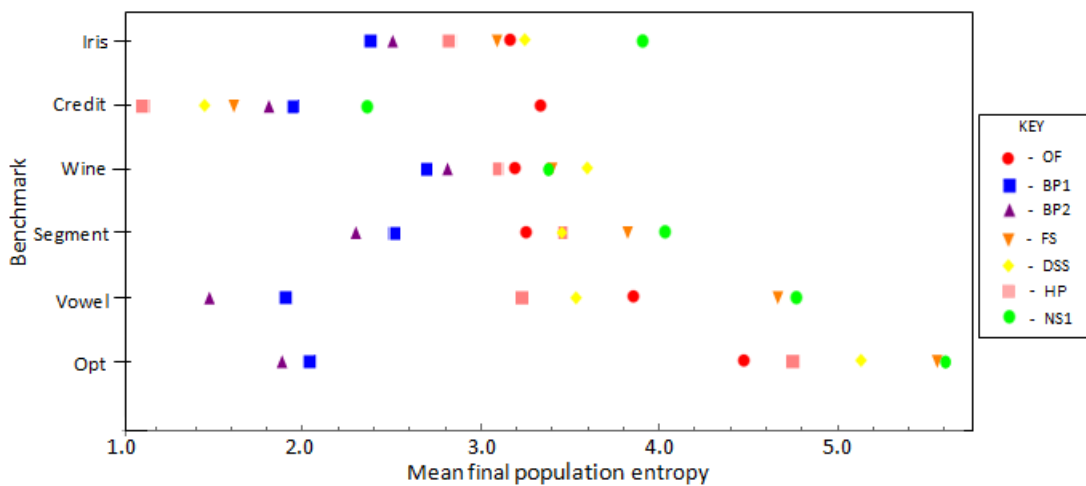


FIGURE 4.26: Supervised classification benchmarks: Mean final population entropy

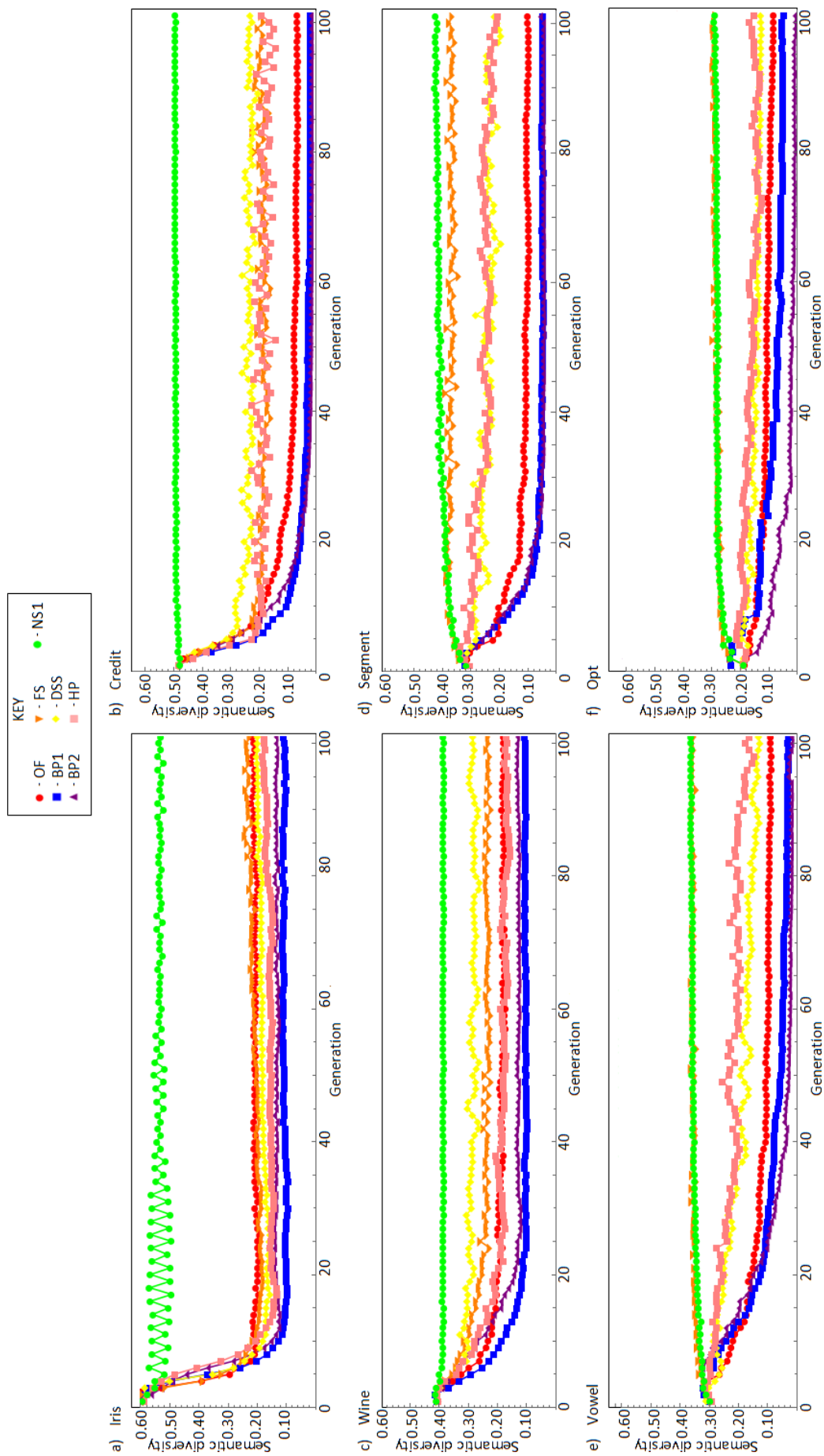


FIGURE 4.27: Supervised classification benchmarks: Generational population semantic diversity

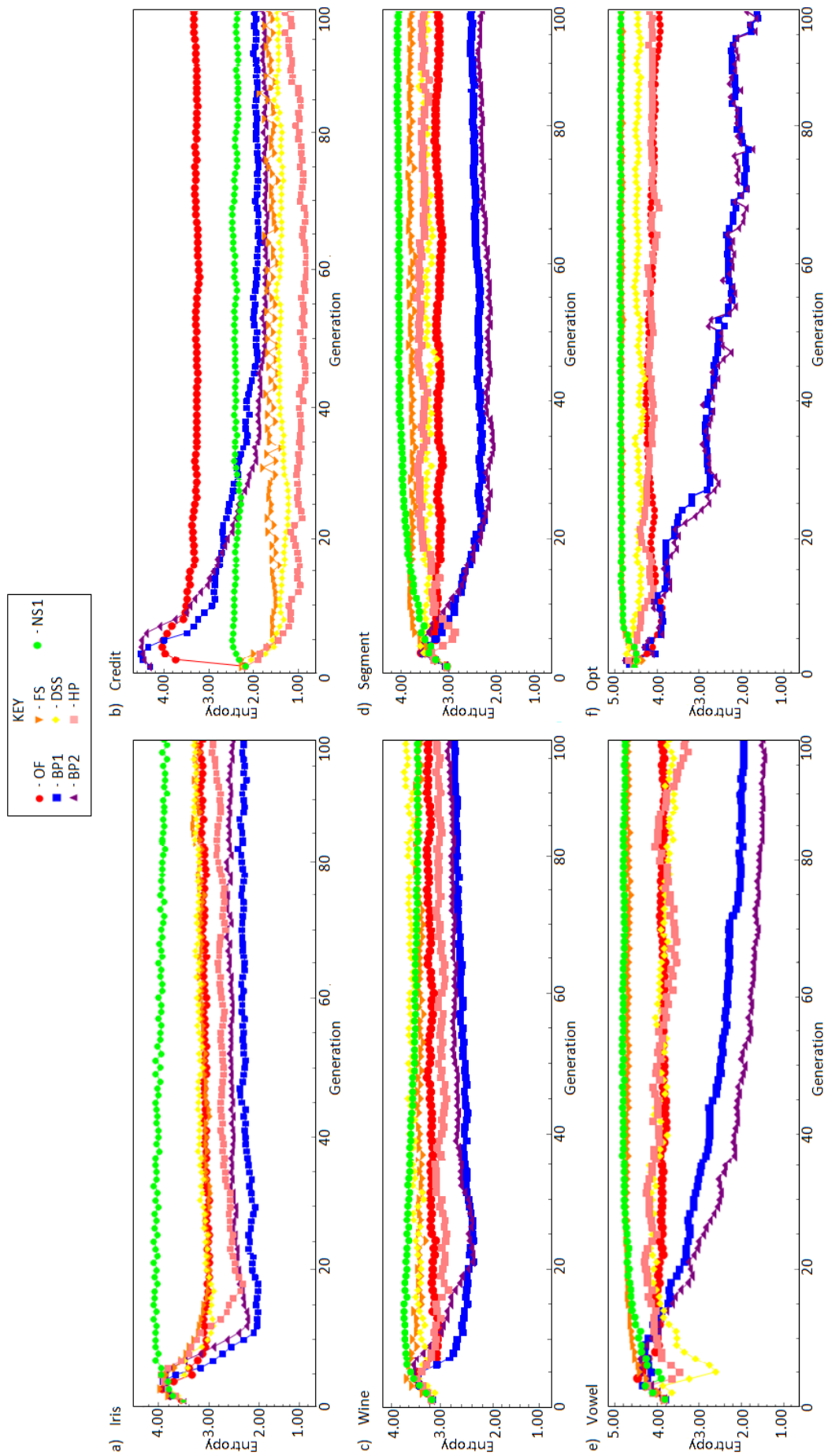


FIGURE 4.28: Supervised classification benchmarks: Generational population entropy

Detailed results

The benchmarks are discussed in the following order: A) Iris, B) Credit, C) Wine, D) Segment, E) Vowel, and F) Opt.

A. Iris

(highest semantic diversity = NS1; lowest semantic diversity = BP1, BP2; highest entropy = NS1; lowest entropy = BP1, BP2)

Table 4.173 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.174 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.173: Iris: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.31	0.21	0.19	0.48	0.29	0.28	0.59
μ	0.21	0.11	0.12	0.20	0.19	0.16	0.53
σ	0.06	0.05	0.05	0.11	0.06	0.07	0.04

TABLE 4.174: Iris: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	3.59	3.05	3.21	4.01	3.61	3.51	4.25
μ	3.15	2.39	2.50	3.09	3.23	2.81	3.91
σ	0.05	0.05	0.04	0.06	0.05	0.05	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.173 and 4.174. Tables 4.175 and 4.176 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.175: Iris: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.19					
FS	0.26	0.00	0.00				
DSS	0.04	0.00	0.00	0.29			
HP	0.00	0.00	0.00	0.03	0.02		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.176: Iris: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.11					
FS	0.28	0.00	0.00				
DSS	0.14	0.00	0.00	0.07			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity and entropy than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity and entropy.

B. Credit

(highest semantic diversity = NS1; lowest semantic diversity = BP1, BP2; highest entropy = OF; lowest entropy = HP)

Table 4.177 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.178 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.177 and 4.178. Tables 4.179 and 4.180 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that NS1 exhibits significantly higher semantic diversity than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity. OF exhibits the highest entropy. Conversely, HP exhibits the lowest entropy.

TABLE 4.177: Credit: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.11	0.04	0.03	0.47	0.50	0.53	0.51
μ	0.07	0.02	0.02	0.19	0.23	0.19	0.50
σ	0.02	0.01	0.01	0.13	0.09	0.15	0.01

TABLE 4.178: Credit: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	3.79	2.78	2.52	3.36	3.78	3.58	2.89
μ	3.33	1.94	1.80	1.61	1.44	1.08	2.35
σ	0.03	0.05	0.06	0.09	0.10	0.11	0.04

TABLE 4.179: Credit: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.15			
HP	0.00	0.00	0.00	0.48	0.19		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.180: Credit: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.13					
FS	0.00	0.09	0.23				
DSS	0.00	0.05	0.15	0.33			
HP	0.00	0.00	0.01	0.06	0.19		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

C. Wine

(highest semantic diversity = NS1; lowest semantic diversity = BP1, BP2; highest entropy = DSS; lowest entropy = BP1, BP2)

Table 4.181 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.182 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.181: Wine: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.28	0.18	0.20	0.32	0.40	0.30	0.40
μ	0.18	0.10	0.13	0.23	0.28	0.17	0.38
σ	0.06	0.03	0.04	0.06	0.07	0.03	0.01

TABLE 4.182: Wine: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS
<i>b</i>	3.72	3.23	3.63	3.95	3.94	3.63	3.67
μ	3.19	2.71	2.81	3.40	3.60	3.09	3.38
σ	0.04	0.03	0.05	0.05	0.05	0.04	0.02

Statistical tests are conducted to verify the significance of the observations in tables 4.181 and 4.182. Tables 4.183 and 4.184 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.183: Wine: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.28	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.184: Wine: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.09					
FS	0.02	0.00	0.00				
DSS	0.00	0.00	0.00	0.01			
HP	0.11	0.00	0.00	0.00	0.00		
NS1	0.01	0.00	0.00	0.38	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity than the other fitness measures. Also DSS exhibits the highest entropy. Furthermore, BP1 and BP2 exhibit significantly lower semantic diversity and entropy than the other fitness measures.

D. Segment

(highest semantic diversity = NS1; lowest semantic diversity = BP1, BP2; highest entropy = NS1; lowest entropy = BP2)

Table 4.185 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.186 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.185: Segment: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.17	0.08	0.36	0.47	0.51	0.39	0.48
μ	0.10	0.05	0.05	0.37	0.22	0.21	0.42
σ	0.04	0.02	0.12	0.07	0.14	0.11	0.03

TABLE 4.186: Segment: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	4.03	3.04	2.88	4.23	4.09	4.06	4.25
μ	3.26	2.52	2.30	3.82	3.45	3.46	4.04
σ	0.05	0.04	0.04	0.03	0.06	0.08	0.01

Statistical tests are conducted to verify the significance of the observations in tables 4.185 and 4.186. Tables 4.187 and 4.188 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.187: Segment: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.24					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.33		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.188: Segment: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.00	0.00				
DSS	0.03	0.00	0.00	0.00			
HP	0.04	0.00	0.00	0.00	0.48		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity and entropy than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity, while BP2 exhibits the lowest entropy.

E. Vowel

(highest semantic diversity = FS, NS1; lowest semantic diversity = BP1, BP2; highest entropy = FS, NS1; lowest entropy = BP2)

Table 4.189 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.190 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.189 and 4.190. Tables 4.191 and 4.192 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.189: Vowel: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.18	0.08	0.09	0.39	0.27	0.38	0.38
μ	0.09	0.03	0.02	0.35	0.13	0.16	0.36
σ	0.03	0.01	0.01	0.02	0.05	0.10	0.01

TABLE 4.190: Vowel: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	4.43	3.39	2.88	4.94	4.64	4.69	4.97
μ	3.86	1.92	1.48	4.67	3.54	3.24	4.77
σ	0.04	0.01	0.06	0.02	0.06	0.03	0.01

TABLE 4.191: Vowel: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.03					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.07		
NS1	0.00	0.00	0.00	0.01	0.00	0.00	

TABLE 4.192: Vowel: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.03					
FS	0.00	0.00	0.00				
DSS	0.01	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.10		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that FS and NS1 exhibit significantly higher semantic diversity and entropy than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity, while BP2 exhibits the lowest entropy.

F. Opt

(highest semantic diversity = FS, NS1; lowest semantic diversity = BP1, BP2; highest entropy = FS, NS1; lowest entropy = BP1, BP2)

Table 4.193 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.194 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.193: Opt: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.16	0.03	0.02	0.37	0.17	0.45	0.33
μ	0.08	0.01	0.01	0.29	0.13	0.15	0.29
σ	0.03	0.01	0.01	0.05	0.03	0.10	0.03

TABLE 4.194: Opt: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	5.49	2.72	2.72	5.80	5.54	5.30	5.74
μ	4.49	2.05	1.89	5.56	5.13	4.76	5.61
σ	0.01	0.04	0.05	0.02	0.03	0.01	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.193 and 4.194. Tables 4.195 and 4.196 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that FS and NS1 exhibit significantly higher semantic diversity and entropy than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity and entropy.

TABLE 4.195: Opt: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.38					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.08		
NS1	0.00	0.00	0.00	0.42	0.00	0.00	

TABLE 4.196: Opt: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.24					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.07	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.06	0.00	0.00	

Boolean function synthesis benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.29 and 4.30 show the mean semantic diversity and entropy of the final GP populations; tables 4.197 and 4.198 contain the same data, whereby the highest and lowest diversity scores are highlighted in the tables. Figures 4.31 and 4.32 show the change in the mean population semantic diversity and the mean population entropy through the generations of GP. The values in the figures and tables below are averaged over the 30 GP runs.

The results show that NS1 exhibits the highest semantic diversity and entropy scores on most problems. HP also exhibits among the highest semantic diversity scores on the par-5, par-7, par-9 and mux-11 problems. In turn, FS exhibits relatively high semantic diversity and entropy on the par-7, par-9 and mult-4 problems. Conversely, BP1 and BP2 exhibit the lowest semantic diversity and entropy on most problems. Also, OF exhibits among the lowest semantic diversity and entropy scores on the mult-3 and mult-4 problems. Overall, FS, HP and NS1 are shown to exhibit relatively high diversity, while BP1 and BP2 exhibit low diversity.

TABLE 4.197: Boolean function synthesis benchmarks: Mean final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	0.12	0.07	0.05	0.23	0.09	0.39	0.49
Par-7	0.06	0.02	0.01	0.41	0.05	0.47	0.48
Par-9	0.07	0.01	0.00	0.45	0.04	0.47	0.49
Mux-11	0.08	0.08	0.07	0.09	0.07	0.51	0.49
Mult-3	0.01	0.01	0.01	0.08	0.01	0.04	0.47
Mult-4	0.01	0.01	0.16	0.22	0.02	0.05	0.46

TABLE 4.198: Boolean function synthesis benchmarks: Mean final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	1.99	1.53	1.14	1.86	1.76	1.34	1.25
Par-7	2.26	1.04	1.02	2.25	1.90	2.34	2.29
Par-9	2.59	0.95	0.51	2.69	2.29	2.59	2.94
Mux-11	4.12	3.49	3.67	3.88	3.87	4.76	5.01
Mult-3	1.46	1.58	1.71	3.07	1.85	2.68	3.83
Mult-4	2.61	2.51	2.40	4.36	2.93	3.62	4.83

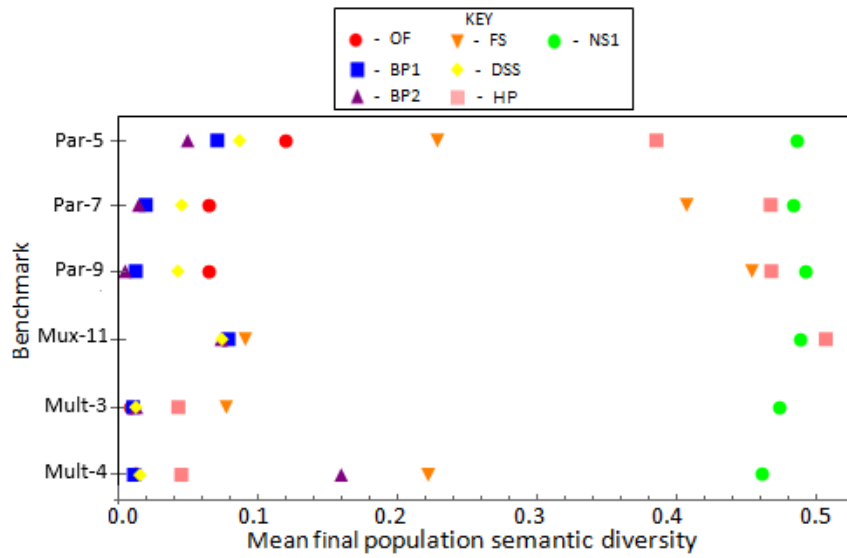


FIGURE 4.29: Boolean function synthesis benchmarks: Mean final population semantic diversity

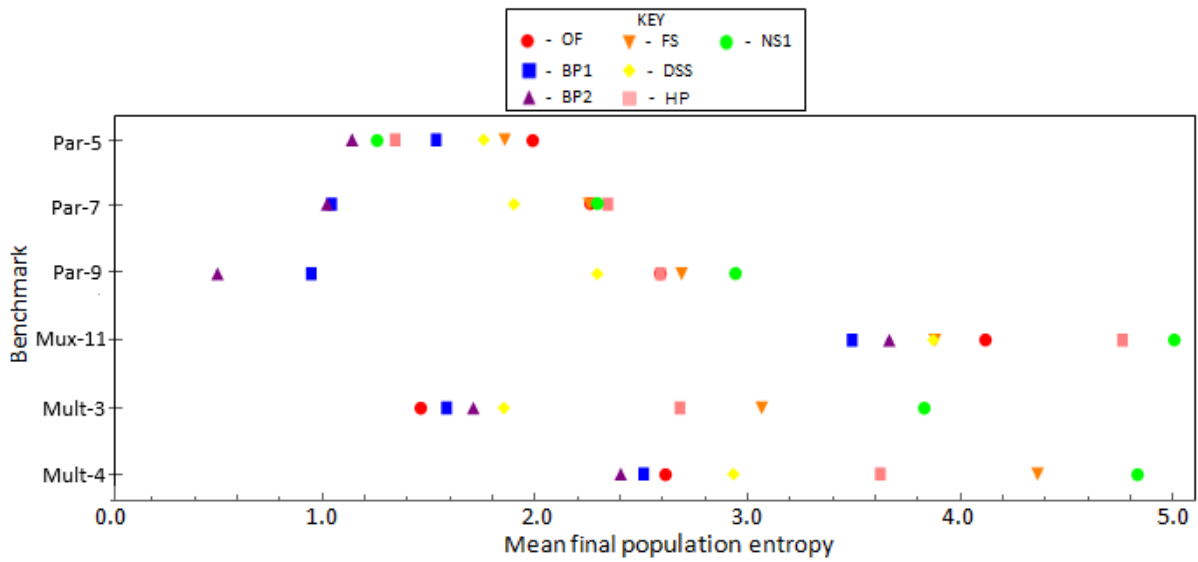


FIGURE 4.30: Boolean function synthesis benchmarks: Mean final population entropy

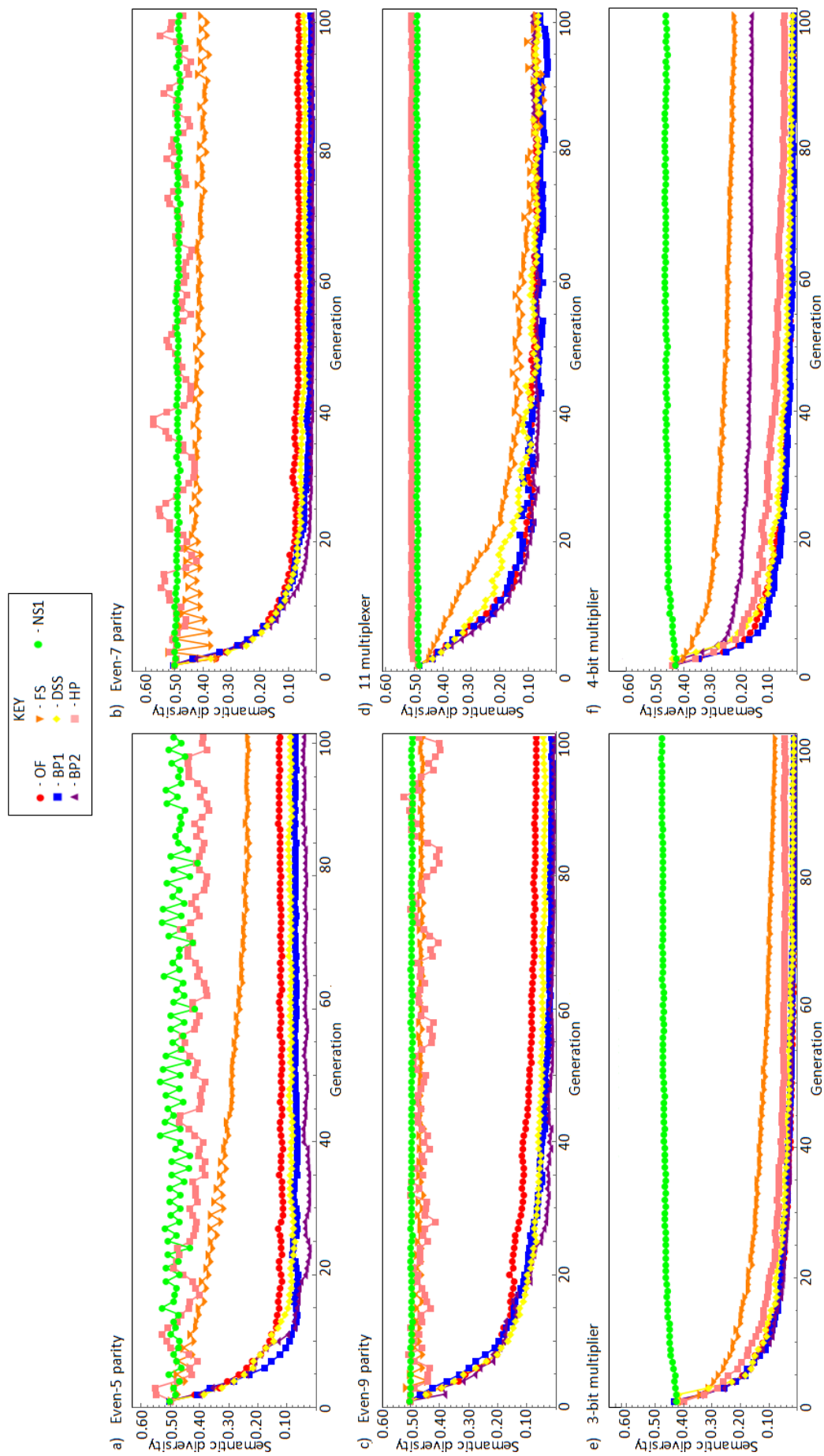


FIGURE 4.31: Boolean function synthesis benchmarks: Generational population semantic diversity

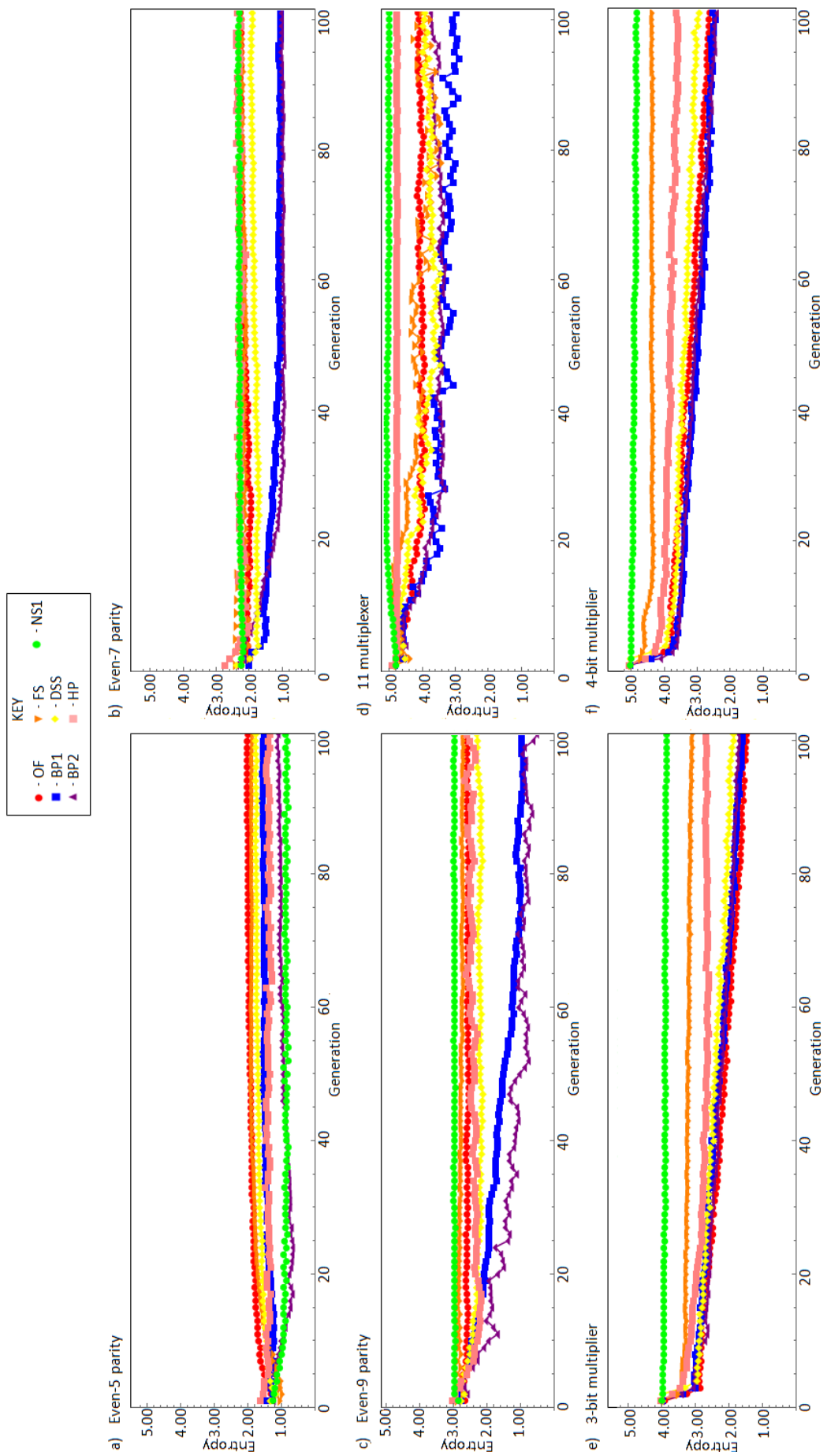


FIGURE 4.32: Boolean function synthesis benchmarks: Generational population entropy

Detailed results

The benchmarks are discussed in the following order: A) Even-5 parity, B) Even-7 parity, C) Even-9 parity, D) 11-multiplexer, E) 3-bit multiplier, and F) 4-bit multiplier.

A. Even-5 parity

(highest semantic diversity = NS1; lowest semantic diversity = BP2; highest entropy = OF; lowest entropy = BP2)

Table 4.199 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.200 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.199: Even-5 parity: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.15	0.11	0.14	0.36	0.15	0.77	0.90
μ	0.12	0.07	0.05	0.23	0.09	0.39	0.49
σ	0.03	0.03	0.04	0.07	0.03	0.19	0.20

TABLE 4.200: Even-5 parity: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	2.19	1.94	1.57	2.25	1.88	1.82	1.81
μ	1.99	1.53	1.14	1.86	1.76	1.34	1.25
σ	0.01	0.03	0.04	0.02	0.01	0.03	0.03

Statistical tests are conducted to verify the significance of the observations in tables 4.199 and 4.200. Tables 4.201 and 4.202 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.201: Even-5 parity: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.01	

TABLE 4.202: Even-5 parity: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.01	0.01	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity than the other fitness measures. Also, OF exhibits the highest entropy. Furthermore, BP2 exhibits the lowest semantic diversity and entropy.

B. Even-7 parity

(highest semantic diversity = HP, NS1; lowest semantic diversity = BP1, BP2; highest entropy = OF, FS, HP, NS1; lowest entropy = BP1, BP2)

Table 4.203 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.204 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

Statistical tests are conducted to verify the significance of the observations in tables 4.203 and 4.204. Tables 4.205 and 4.206 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.203: Even-7 parity: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.10	0.04	0.04	0.51	0.14	0.73	0.51
μ	0.06	0.02	0.01	0.41	0.05	0.47	0.48
σ	0.02	0.01	0.01	0.07	0.03	0.19	0.02

TABLE 4.204: Even-7 parity: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	2.64	1.60	1.50	2.56	2.32	2.74	2.47
μ	2.26	1.04	1.02	2.25	1.90	2.34	2.29
σ	0.03	0.03	0.04	0.02	0.03	0.03	0.01

TABLE 4.205: Even-7 parity: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.03	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.30	

TABLE 4.206: Even-7 parity: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.39					
FS	0.43	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.09	0.00	0.00	0.05	0.00		
NS1	0.22	0.00	0.00	0.12	0.00	0.15	

The results indicate that HP and NS1 exhibit significantly higher semantic diversity than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity and entropy. In turn, OF, FS, HP and NS1 exhibit on par entropy on the problem.

C. Even-9 parity

(highest semantic diversity = FS, HP, NS1; lowest semantic diversity = BP1, BP2; highest entropy = NS1; lowest entropy = BP2)

Table 4.207 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.208 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.207: Even-9 parity: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.10	0.03	0.01	0.49	0.12	0.70	0.50
μ	0.07	0.01	0.00	0.45	0.04	0.47	0.49
σ	0.01	0.00	0.00	0.03	0.03	0.11	0.01

TABLE 4.208: Even-9 parity: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	2.91	2.03	1.05	2.88	2.83	3.18	3.09
μ	2.59	0.95	0.51	2.69	2.29	2.59	2.94
σ	0.03	0.05	0.15	0.01	0.04	0.04	0.01

Statistical tests are conducted to verify the significance of the observations in tables 4.207 and 4.208. Tables 4.209 and 4.210 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

The results indicate that FS, HP and NS1 exhibit significantly higher semantic diversity than the other fitness measures. Also, NS1 exhibits significantly higher entropy than the other fitness measures. Furthermore, BP1 and BP2 exhibit the lowest semantic diversity, while BP2 exhibits the lowest entropy.

TABLE 4.209: Even-9 parity: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.02					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.29	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.17	

TABLE 4.210: Even-9 parity: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.03					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.49	0.00	0.00	0.08	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

D. 11-multiplexer

(highest semantic diversity = HP, NS1; lowest semantic diversity = OF, BP1, BP2, FS, DSS; highest entropy = NS1; lowest entropy = BP1, BP2)

Table 4.211 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.212 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.211: 11-multiplexer: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	0.15	0.19	0.19	0.18	0.12	0.52	0.50
μ	0.08	0.08	0.07	0.09	0.07	0.51	0.49
σ	0.04	0.04	0.04	0.04	0.03	0.01	0.01

TABLE 4.212: 11-multiplexer: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
<i>b</i>	4.52	4.50	4.37	4.97	4.45	4.85	5.11
μ	4.12	3.49	3.67	3.88	3.87	4.76	5.01
σ	0.05	0.04	0.05	0.03	0.03	0.04	0.04

Statistical tests are conducted to verify the significance of the observations in tables 4.211 and 4.212. Tables 4.213 and 4.214 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.213: 11-multiplexer: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.40						
BP2	0.40	0.34					
FS	0.05	0.17	0.05				
DSS	0.39	0.34	0.48	0.03			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.27	

TABLE 4.214: 11-multiplexer: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.10						
BP2	0.00	0.30					
FS	0.01	0.20	0.08				
DSS	0.00	0.20	0.07	0.47			
HP	0.00	0.01	0.00	0.00	0.00		
NS1	0.00	0.01	0.00	0.00	0.00	0.00	

The results indicate that HP and NS1 exhibit significantly higher semantic diversity than the other fitness measures. In turn, NS1 exhibits significantly higher entropy than the other fitness measures. OF, BP1, BP2, FS and DSS exhibit on par semantic diversity. On the other hand, BP1 and BP2 exhibit the lowest entropy.

E. 3-bit multiplier

(highest semantic diversity = NS1; lowest semantic diversity = OF, BP1, BP2, DSS; highest entropy = NS1; lowest entropy = OF, BP1, BP2)

Table 4.215 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.216 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.215: 3-bit multiplier: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.02	0.02	0.02	0.10	0.02	0.07	0.50
μ	0.01	0.01	0.01	0.08	0.01	0.04	0.47
σ	0.00	0.00	0.00	0.01	0.00	0.01	0.01

TABLE 4.216: 3-bit multiplier: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	2.25	2.27	2.06	3.35	2.44	3.03	3.98
μ	1.46	1.58	1.71	3.07	1.85	2.68	3.83
σ	0.04	0.05	0.03	0.02	0.04	0.02	0.03

Statistical tests are conducted to verify the significance of the observations in tables 4.215 and 4.216. Tables 4.217 and 4.218 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.217: 3-bit multiplier: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.47						
BP2	0.45	0.47					
FS	0.00	0.00	0.00				
DSS	0.42	0.39	0.35	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.218: 3-bit multiplier: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.08						
BP2	0.00	0.05					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.02	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity and entropy than the other fitness measures. OF, BP1, BP2 and DSS exhibit the lowest semantic diversity. Also, OF, BP1 and BP2 exhibit the lowest entropy.

F. 4-bit multiplier

(highest semantic diversity = NS1; lowest semantic diversity = OF, BP1, DSS; highest entropy = NS1; lowest entropy = OF, BP1, BP2)

Table 4.219 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.220 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.219: 4-bit multiplier: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1
b	0.03	0.02	0.17	0.27	0.04	0.08	0.48
μ	0.01	0.01	0.16	0.22	0.02	0.05	0.46
σ	0.00	0.01	0.15	0.20	0.01	0.02	0.42

TABLE 4.220: 4-bit multiplier: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1
b	3.13	3.30	2.90	4.59	3.51	4.08	5.00
μ	2.61	2.51	2.40	4.36	2.93	3.62	4.83
σ	1.97	1.84	1.76	4.21	2.21	3.06	4.66

Statistical tests are conducted to verify the significance of the observations in tables 4.219 and 4.220. Tables 4.221 and 4.222 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.221: 4-bit multiplier: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.16						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.02	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.222: 4-bit multiplier: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.13						
BP2	0.01	0.12					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 exhibits significantly higher semantic diversity and entropy than the other fitness measures. OF, BP1 and DSS exhibit the lowest semantic diversity. Also, OF, BP1 and BP2 exhibit the lowest entropy.

Path-finding benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figures 4.33 and 4.34 show the mean semantic diversity and entropy of the final GP populations; tables 4.223 and 4.224 contain the same data, whereby the highest and lowest diversity scores are highlighted in the tables. Figures 4.35 and 4.36 show the change in the mean population semantic diversity and the mean population entropy through the generations of GP. The values in the figures and tables below are averaged over the 30 GP runs.

The results show that unlike in the previous domains, FS and NS1 exhibit the lowest semantic diversity and entropy on the tartarus and deceptive tartarus problems. On the other hand, NS2, which employs task-specific behavior descriptors, exhibits the highest semantic diversity and entropy on these problems. BP1 and BP2 also exhibit higher semantic diversity and entropy than FS and NS1 on the tartarus and deceptive tartarus problems. Overall, NS2 is seen to exhibit high semantic diversity and entropy, whereas FS and NS1 exhibit low semantic diversity and entropy on the tartarus and deceptive tartarus problems.

TABLE 4.223: Path-finding benchmarks: Mean final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	0.38	0.48	0.48	0.58	0.46	0.47	0.56	0.47
Tart.	0.06	0.06	0.02	0.00	0.07	0.03	0.00	0.22
Dec. tart.	0.02	0.19	0.11	0.00	0.02	0.02	0.00	0.31

TABLE 4.224: Path-finding benchmarks: Mean final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	2.96	3.26	3.41	3.32	3.27	3.42	3.45	3.73
Tart.	3.75	3.78	1.67	0.96	3.42	3.63	1.07	4.50
Dec. tart.	1.85	3.35	2.08	1.01	2.12	1.75	1.00	3.39

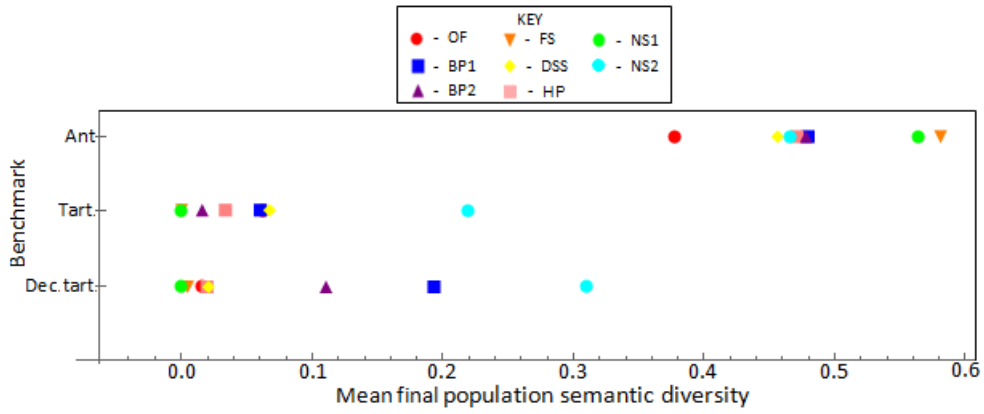


FIGURE 4.33: Path-finding benchmarks: Mean final population semantic diversity

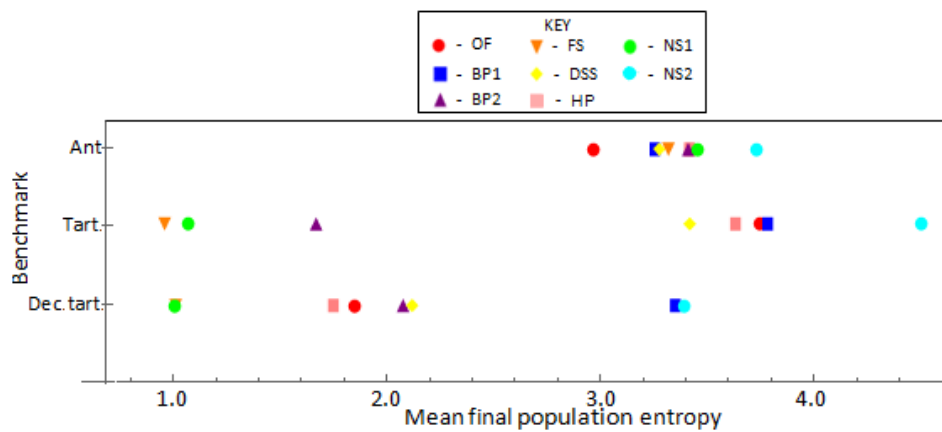


FIGURE 4.34: Path-finding benchmarks: Mean final population entropy

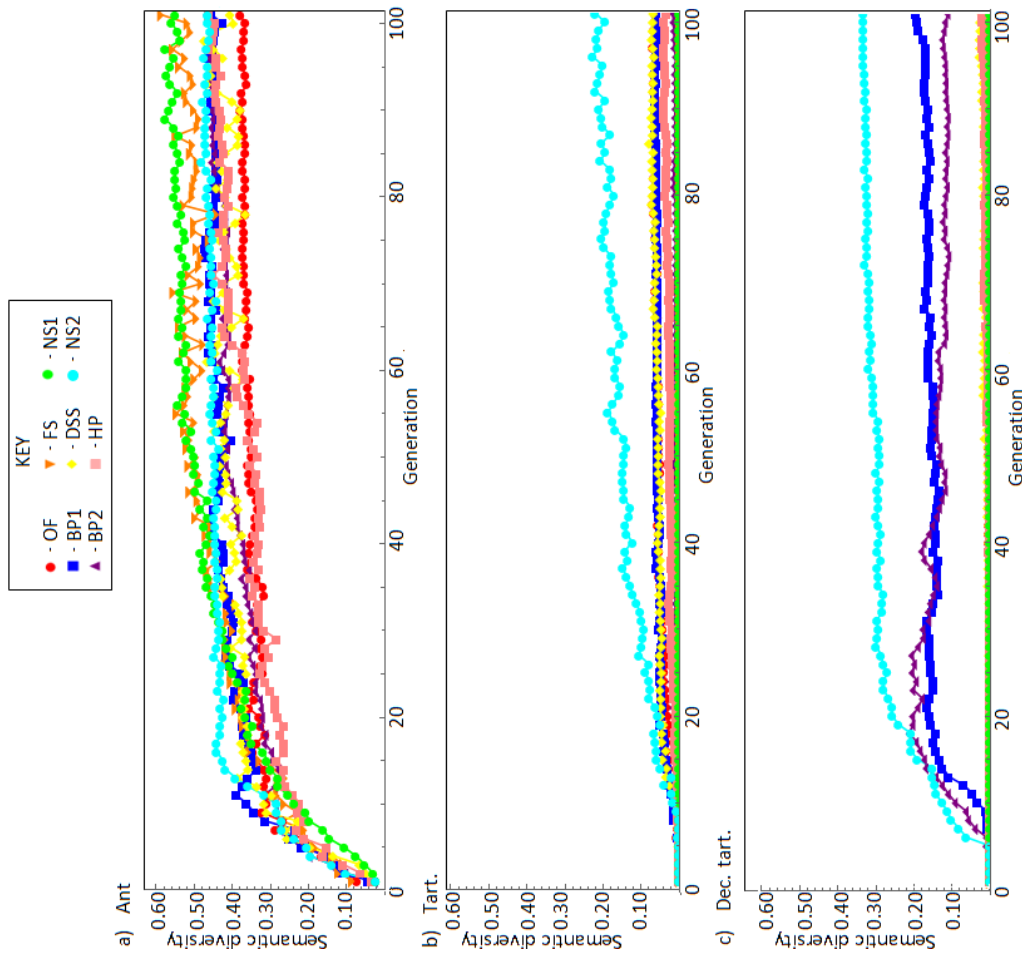


FIGURE 4.35: Path-finding benchmarks: Generational population semantic diversity

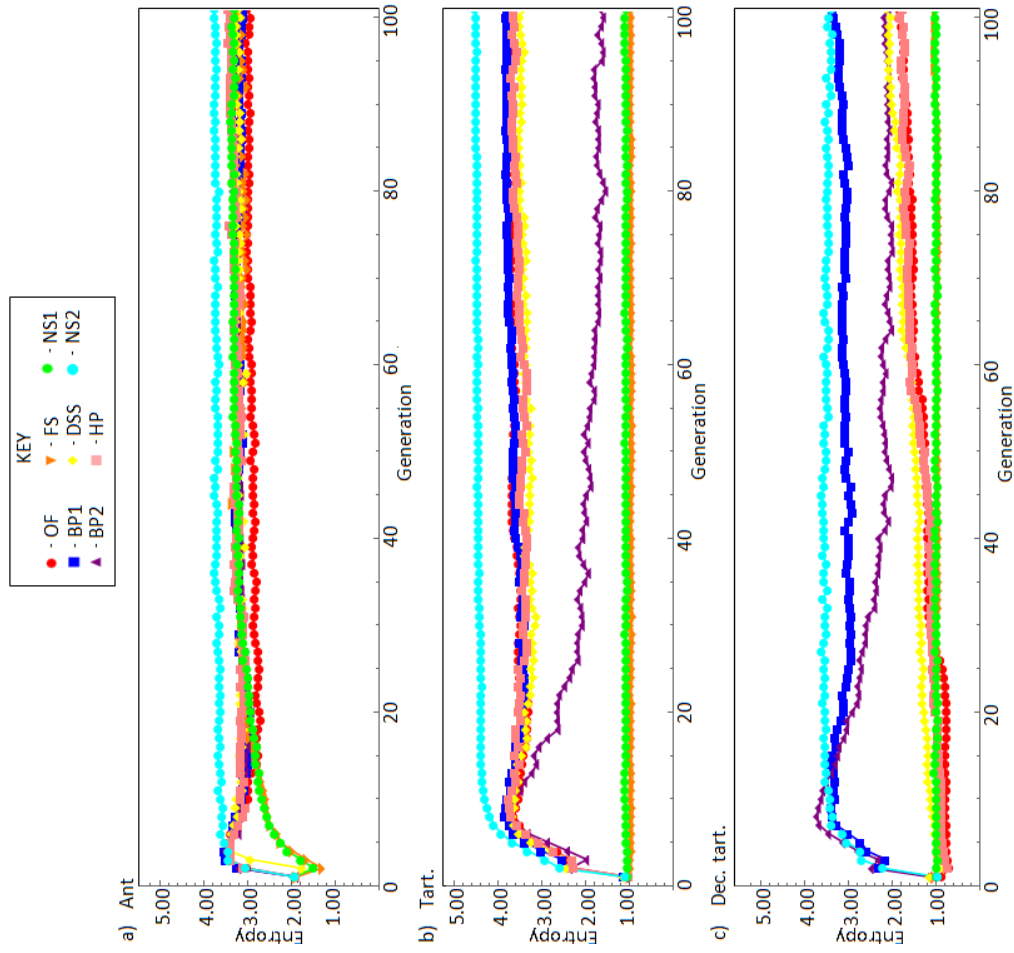


FIGURE 4.36: Path-finding benchmarks: Generational population entropy

Detailed results

The benchmarks are discussed in the following order: A) Artificial ant, B) Tartarus, and C) Deceptive tartarus.

A. Artificial ant

(highest semantic diversity = FS, NS1; lowest semantic diversity = OF; highest entropy = NS2; lowest entropy = OF)

Table 4.225 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.226 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.225: Artificial ant: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	0.53	0.65	0.73	0.71	0.67	0.70	0.69	0.53
μ	0.38	0.48	0.48	0.58	0.46	0.47	0.56	0.47
σ	0.05	0.06	0.07	0.04	0.06	0.06	0.09	0.05

TABLE 4.226: Artificial ant: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	3.45	3.66	3.80	3.95	3.76	3.86	3.93	3.90
μ	2.96	3.26	3.41	3.32	3.27	3.42	3.45	3.73
σ	0.10	0.09	0.09	0.14	0.09	0.11	0.09	0.06

Statistical tests are conducted to verify the significance of the observations in tables 4.225 and 4.226. Tables 4.227 and 4.228 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.227: Artificial ant: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.48						
FS	0.00	0.00	0.00					
DSS	0.00	0.19	0.22	0.00				
HP	0.00	0.39	0.41	0.00	0.32			
NS1	0.00	0.00	0.00	0.20	0.00	0.00		
NS2	0.00	0.22	0.28	0.00	0.34	0.41	0.00	

TABLE 4.228: Artificial ant: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.02						
FS	0.00	0.21	0.13					
DSS	0.00	0.40	0.05	0.31				
HP	0.00	0.01	0.47	0.11	0.04			
NS1	0.00	0.02	0.33	0.09	0.04	0.35		
NS2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that FS and NS1 exhibit significantly higher semantic diversity than the other fitness measures. Also NS2 exhibits the highest entropy. Furthermore, OF exhibits significantly lower semantic diversity and entropy than the other fitness measures.

B. Tartarus

(highest semantic diversity = NS2; lowest semantic diversity = FS, NS1; highest entropy = NS2; lowest entropy = FS)

Table 4.229 lists the best (b), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.230 lists the best (b), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.229: Tartarus: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	0.22	0.24	0.12	0.00	0.26	0.21	0.00	0.41
μ	0.06	0.06	0.02	0.00	0.07	0.03	0.00	0.22
σ	0.05	0.06	0.03	0.00	0.06	0.06	0.00	0.06

TABLE 4.230: Tartarus: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	4.35	4.19	3.68	1.18	4.26	4.15	1.23	4.81
μ	3.75	3.78	1.67	0.96	3.42	3.63	1.07	4.50
σ	0.20	0.10	0.23	0.07	0.19	0.14	0.05	0.08

Statistical tests are conducted to verify the significance of the observations in tables 4.229 and 4.230. Tables 4.231 and 4.232 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables. The results indicate

TABLE 4.231: Tartarus: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.46							
BP2	0.00	0.00						
FS	0.00	0.00	0.10					
DSS	0.38	0.33	0.00	0.00				
HP	0.04	0.03	0.06	0.00	0.02			
NS1	0.00	0.00	0.01	0.50	0.00	0.00		
NS2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.232: Tartarus: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.35							
BP2	0.00	0.00						
FS	0.00	0.00	0.00					
DSS	0.01	0.00	0.00	0.00				
HP	0.16	0.07	0.00	0.00	0.06			
NS1	0.00	0.00	0.00	0.00	0.00	0.00		
NS2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

that NS2 exhibits significantly higher semantic diversity and entropy than the other fitness measures. In turn, FS and NS1 exhibit the lowest semantic diversity, while FS exhibits the lowest entropy.

C. Deceptive tartarus

(highest semantic diversity = NS2; lowest semantic diversity = OF, FS, DSS, HP, NS1; highest entropy = BP1, NS2; lowest entropy = FS, NS1)

Table 4.233 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population semantic diversity. In turn, Table 4.234 lists the best (*b*), mean (μ) and standard deviation (σ) of the final population entropy. The highest and lowest diversity scores are highlighted in the tables.

TABLE 4.233: Deceptive tartarus: Final population semantic diversity

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
<i>b</i>	0.14	0.40	0.33	0.14	0.32	0.12	0.00	0.40
μ	0.02	0.19	0.11	0.00	0.02	0.02	0.00	0.31
σ	0.02	0.06	0.06	0.02	0.09	0.03	0.00	0.02

TABLE 4.234: Deceptive tartarus: Final population entropy

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
<i>b</i>	3.44	4.24	3.68	3.47	3.42	3.27	1.15	4.16
μ	1.85	3.35	2.08	1.01	2.12	1.75	1.00	3.39
σ	0.20	0.22	0.31	0.40	0.42	0.45	0.04	0.21

Statistical tests are conducted to verify the significance of the observations in tables 4.233 and 4.234. Tables 4.235 and 4.236 list the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the tables.

TABLE 4.235: Deceptive tartarus: Statistical tests on final population semantic diversity

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.00						
FS	0.11	0.00	0.00					
DSS	0.35	0.00	0.00	0.10				
HP	0.34	0.00	0.00	0.05	0.47			
NS1	0.10	0.00	0.00	0.16	0.04	0.01		
NS2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE 4.236: Deceptive tartarus: Statistical tests on final population entropy

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.22	0.00						
FS	0.00	0.00	0.00					
DSS	0.17	0.00	0.44	0.00				
HP	0.35	0.00	0.12	0.00	0.08			
NS1	0.00	0.00	0.00	0.47	0.10	0.10		
NS2	0.00	0.32	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS2 exhibits significantly higher semantic diversity than the other fitness measures. In turn, NS2 and BP1 exhibit the highest entropy. Furthermore, OF, FS, DSS, HP and NS1 exhibit on par semantic diversity, whereas FS and NS1 exhibit the lowest entropy.

Summary of the diversity results

The results show that FS, DSS, HP and NS, intended to maintain high levels of diversity in GP largely achieve this. Nevertheless, the fitness measures do not exhibit high diversity on all the problems. The following general observations are made:

- i) FS, DSS, HP, and NS1/NS2 demonstrate the ability to perpetuate the semantic and fitness diversity of GP. Based on the discussion in section 3.4 of chapter 3, FS perpetuates diversity by favoring candidate solution programs that solve unique fitness cases compared to the rest of the population. In turn, DSS and HP continually adapt the fitness case subset used for OF evaluation, discouraging premature convergence to solution programs that solve only “easy” fitness cases. NS favors candidate solution programs that have the highest semantic distance to their k -nearest neighbours. The results show that FS, DSS, HP and NS1/NS2 maintain high levels of semantic diversity and entropy on most of the benchmark problems.
- ii) BP is shown to exhibit low diversity. Section 3.3.2 of 3 established BP favors candidate solution programs that contain useful modules. This results in the spread of the useful modules within the GP population. It follows that as a result of the spread of useful modules, the candidate solution programs in the population become similar with respect to their OF scores as well as the fitness cases solved. The above arguments justify the observation that BP1 and BP2 exhibit low semantic diversity and entropy on most of the benchmark problems.
- iii) The fitness measures do not always influence diversity as described in this section. For example, FS and NS1 exhibit low diversity on the tartarus and deceptive tartarus problems. The fitness cases in the tartarus and deceptive tartarus problems are difficult [248]: random initial population individuals tend not to solve any of the fitness cases. The population individuals solving few (or none) of fitness cases results in low population semantic diversity due to low diversity in the fitness cases solved, as well as low population entropy, due to low diversity in the fitness values achieved. Furthermore, the solution quality results in section 4.3.1 showed that FS and NS1 make little (or no) improvements on the quality of the initial population individuals: thus the fitness measures retain the low initial population semantic diversity and entropy on the problems, as shown in figures 4.35 and 4.36 of section 4.3.3. The indication is that the extent to which a fitness measure influences the population diversity on a given problem depends on its suitability. A suitable fitness measure improves on the initial population, creating more opportunity for the candidate solution programs in subsequent generations to solve diverse fitness cases and achieve diverse fitness scores. Conversely, a fitness measure that fails to improve on the initial population retains the low initial population semantic diversity and entropy.
- iv) The effect of diversity on the quality of the candidate solution programs differs for different problems. For example, the solution quality results in section 4.3.1 showed that BP achieves high quality on modular problems. Hence on these problems, high quality is seen with the low diversity exhibited when using BP. Here, low diversity is associated with the exploitation of useful modules. On the other hand, the solution quality results in section 4.3.1 also showed that FS, DSS, HP and NS achieve high quality on problems prone to local optima. Therefore, on these problems high quality is seen with the high diversity exhibited when using the listed fitness measures. In this case, diversity precludes premature convergence. Overall, the results show that the relationship between diversity and solution quality differs for different problems, whereby different levels of diversity suit different problems.

4.3.4 Structural complexity

This section reports on the structural complexity associated with the different fitness measures. GP solution programs become opaque to human understanding as their size increases [259]. Furthermore, a bloated GP population consumes computational and memory resources [107, 259]. Therefore, in the case of structural complexity, lower values indicate a better result. A presentation of the results from each domain ensues.

Symbolic regression benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.37 shows the mean size of the best solution program found by GP. The mean size of the best solution program is also shown in table 4.237, where the lowest values achieved on each problem are highlighted. In turn, figure 4.38 shows the generational mean population size. In the figures and tables below, lower values indicate better performance of GP. All values are averaged over the 30 GP runs.

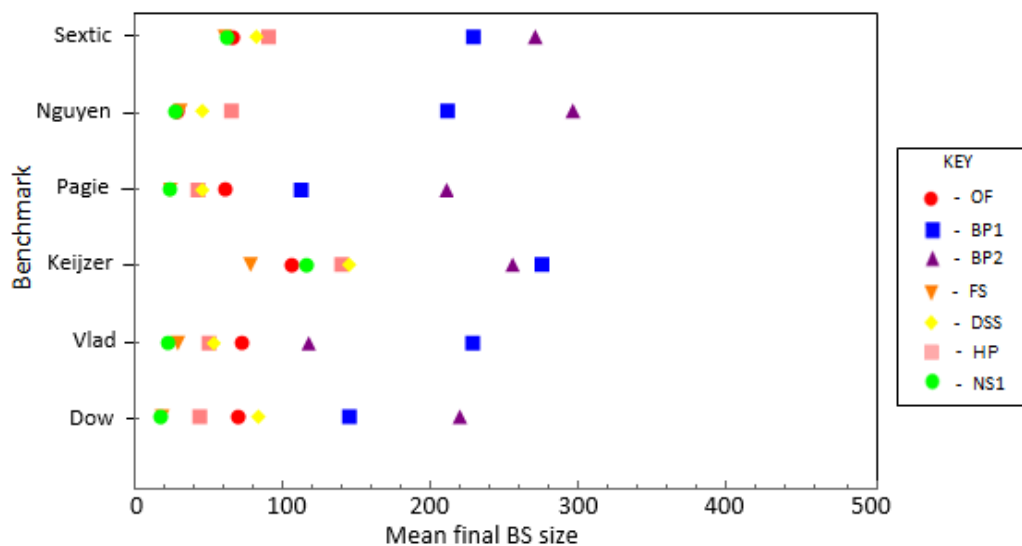


FIGURE 4.37: Symbolic regression benchmarks: Mean final BS size

TABLE 4.237: Symbolic regression benchmarks: Mean final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	66.35	229.52	271.22	61.34	82.86	91.02	63.41
Nguyen	29.36	211.95	296.75	31.37	46.29	65.84	27.92
Pagie	57.46	113.70	211.97	24.69	51.49	48.02	24.28
Keijzer	106.63	275.22	256.05	78.81	145.53	140.54	116.39
Vlad	70.28	227.17	116.02	27.29	48.41	51.67	20.38
Dow	69.00	145.72	220.29	19.06	84.15	44.57	19.52

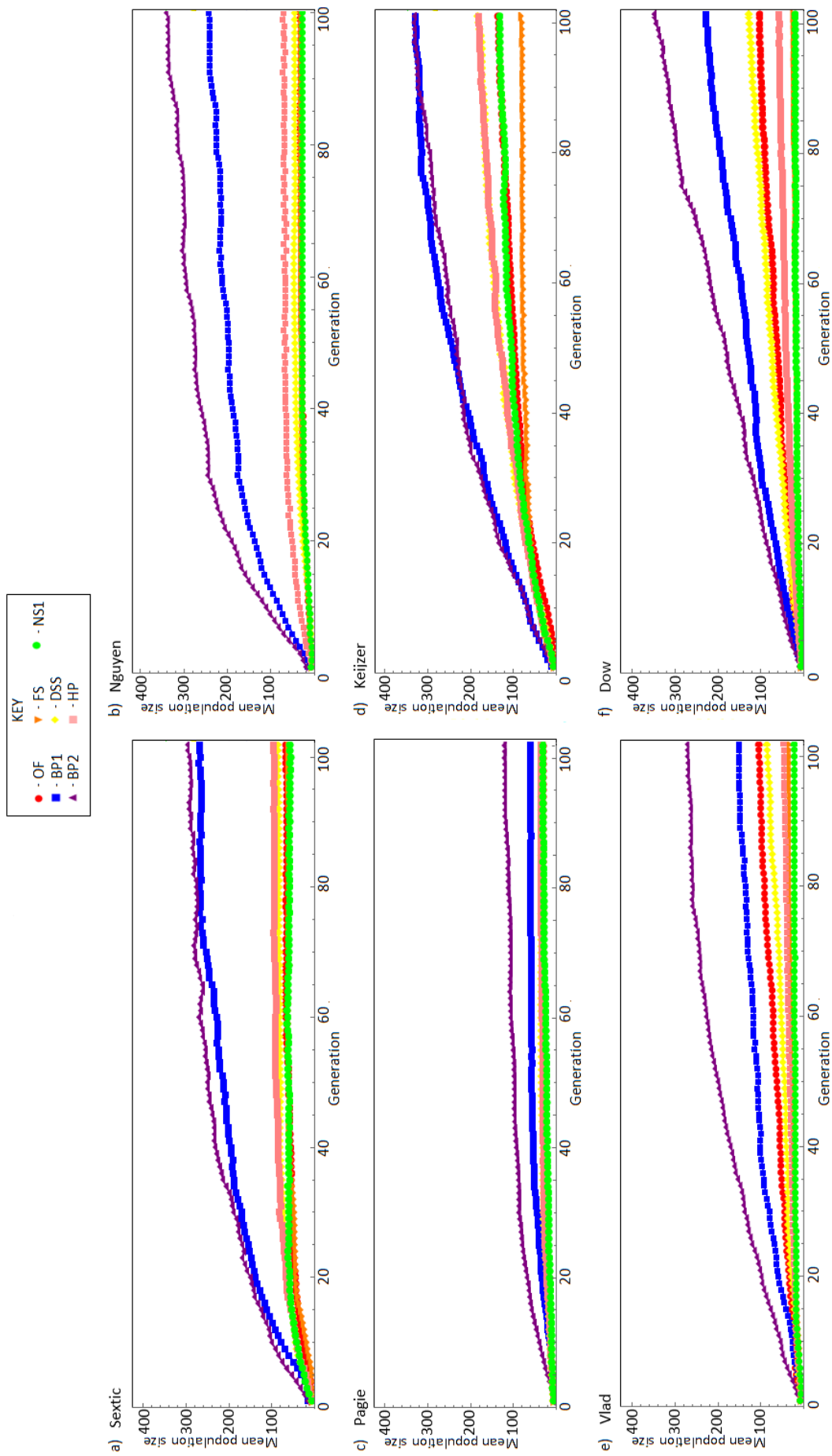


FIGURE 4.38: Symbolic regression benchmarks: Mean generational population size

The results show that FS and NS1 generally discover the smallest solution programs, whereas BP1 and BP2 discover the largest solution programs.

Detailed results

The benchmarks are discussed in the following order: A) Sextic, B) Nguyen, C) Pagie, D) Keijzer, E) Vlad, and F) Dow.

A. Sextic (lowest structural complexity = OF, FS, NS1)

Table 4.238 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions are highlighted in the table.

TABLE 4.238: Sextic: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	95.00	295.00	355.00	77.00	133.00	209.00	78.00
μ	66.35	229.52	271.22	61.34	82.86	91.02	63.41
σ	10.20	24.42	26.01	6.45	15.51	39.31	5.65

Statistical tests are conducted to verify the significance of the observations in tables 4.238. Table 4.239 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.239: Sextic: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.11	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.05		
NS1	0.07	0.00	0.00	0.22	0.00	0.00	

The results indicate that OF, FS and NS1 discover the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

B. Nguyen (lowest structural complexity = OF, FS, NS1)

Table 4.240 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.240: Nguyen: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	96.00	301.00	371.00	91.00	139.00	221.00	90.00
μ	66.35	229.52	271.22	61.34	82.86	91.02	63.41
σ	12.47	26.22	29.21	9.54	18.13	41.14	6.65

Statistical tests are conducted to verify the significance of the observations in tables 4.240. Table 4.241 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that OF, FS and NS1 discover the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

TABLE 4.241: Nguyen: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.17	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.29	0.00	0.00	0.41	0.00	0.00	

C. *Pagie* (lowest structural complexity = FS, NS1)

Table 4.242 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.242: *Pagie*: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	119.00	181.00	269.00	41.00	99.00	126.00	42.00
μ	57.46	113.70	211.97	24.69	51.49	48.02	24.28
σ	22.73	25.22	25.01	7.46	20.33	35.37	7.01

Statistical tests are conducted to verify the significance of the observations in tables 4.242. Table 4.243 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.243: *Pagie*: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.37		
NS1	0.00	0.00	0.00	0.43	0.00	0.00	

The results indicate that FS and NS1 discover the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

D. *Keijzer* (lowest structural complexity = FS)

Table 4.244 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.244: *Keijzer*: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	202.00	381.00	371.00	131.00	230.00	256.00	167.00
μ	106.63	275.22	256.05	78.81	145.53	140.54	116.39
σ	32.34	35.42	35.21	17.63	30.12	45.17	21.09

Statistical tests are conducted to verify the significance of the observations in tables 4.244. Table 4.245 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.245: Keijzer: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.05					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.21		
NS1	0.10	0.00	0.00	0.00	0.00	0.00	

The results indicate that FS discovers the smallest solution programs. Furthermore, BP1 and BP2 discover the largest solution programs.

E. Vlad (lowest structural complexity = FS, NS1)

Table 4.246 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.246: Vlad: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	181.00	341.00	249.00	87.00	138.00	179.00	80.00
μ	70.28	227.17	116.02	27.29	48.41	51.67	20.38
σ	37.12	42.11	49.12	22.32	35.22	39.26	24.29

Statistical tests are conducted to verify the significance of the observations in tables 4.246. Table 4.247 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.247: Vlad: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.04	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.15	0.00	0.00	

The results indicate that FS and NS1 discover the smallest solution programs. Furthermore, BP1 discovers the largest solution programs.

F. Dow (lowest structural complexity = FS, NS1)

Table 4.248 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.248: Dow: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	162.00	265.00	321.00	79.00	172.00	135.00	82.00
μ	69.00	145.72	220.29	19.06	84.15	44.57	19.52
σ	33.12	39.16	40.86	20.15	37.19	32.21	21.27

Statistical tests are conducted to verify the significance of the observations in tables 4.246. Table 4.247 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.249: Dow: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.05	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.40	0.00	0.00	

The results indicate that FS and NS1 discover the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

Supervised classification benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.39 shows the mean size of the best solution program found by GP. The mean size of the best solution program is also shown in table 4.250, whereby the lowest values achieved on each problem are highlighted. In turn, figure 4.40 shows the generational mean population size. In the figures and tables below, lower values indicate better performance of GP. All values are averaged over the 30 GP runs.

The results show that NS1 generally discovers the smallest solution programs, whereas BP1 and BP2 discover the largest solution programs on the problems in the domain.

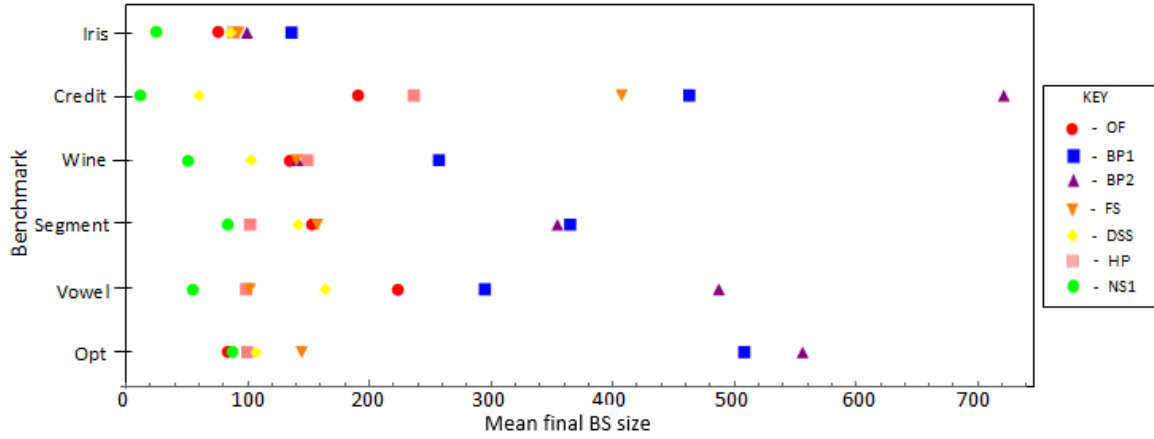


FIGURE 4.39: Supervised classification benchmarks: Mean final BS size

TABLE 4.250: Supervised classification benchmarks: Mean final BS size

	OF	BP1	BP2	FS	DSS	HP	NS
Iris	76.13	136.00	99.40	92.47	86.13	88.33	25.47
Credit	191.13	462.87	721.36	407.27	60.20	236.53	11.67
Wine	134.53	257.27	140.80	140.00	102.80	149.07	51.13
Segment	152.47	365.00	354.53	157.20	141.47	102.00	84.13
Vowel	222.93	294.93	487.17	101.67	163.80	98.67	54.93
Opt	84.20	508.15	555.94	144.33	106.87	99.67	88.13

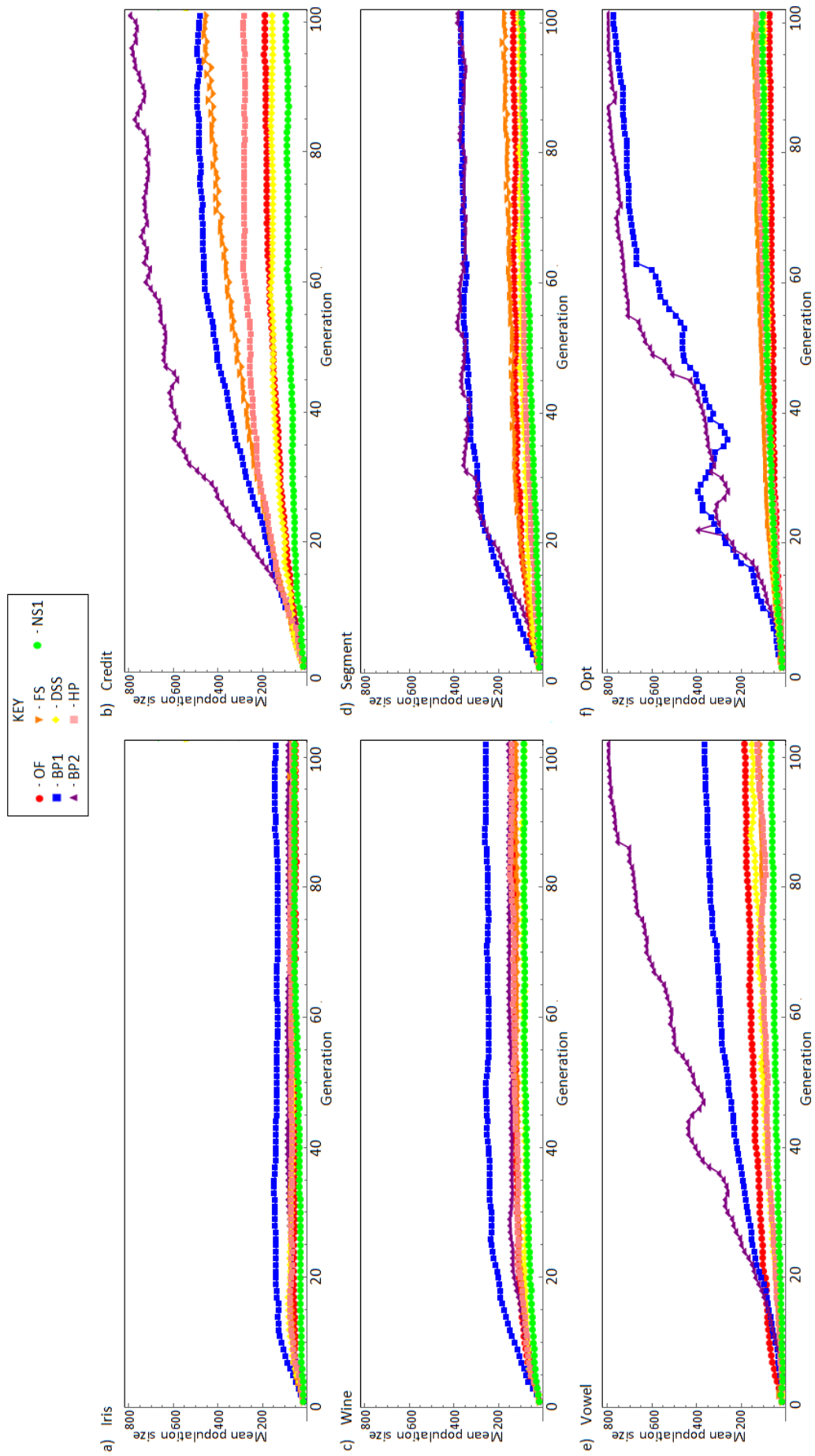


FIGURE 4.40: Supervised classification benchmarks: Mean generational population size

Detailed results

The benchmarks are discussed in the following order: A) Iris, B) Credit, C) Wine, D) Segment, E) Vowel, and F) Opt.

A. Iris (lowest structural complexity = NS1)

Table 4.251 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.251: Iris: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	165.00	255.00	215.00	355.00	179.00	253.00	77.00
μ	76.13	136.00	99.40	92.47	86.13	88.33	25.47
σ	29.20	35.34	35.41	80.24	25.75	50.31	15.55

Statistical tests are conducted to verify the significance of the observations in tables 4.251. Table 4.252 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.252: Iris: Statistical tests on final BS size

One-way ANOVA p-value = 0.05							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.03	0.01					
FS	0.16	0.01	0.34				
DSS	0.17	0.00	0.13	0.35			
HP	0.16	0.00	0.20	0.41	0.43		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP1 discovers the largest solution programs.

B. Credit (lowest structural complexity = NS1)

Table 4.253 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.253: Credit: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	455.00	775.00	1563.00	727.00	189.00	565.00	63.00
μ	191.13	462.87	721.36	407.27	60.20	236.53	11.67
σ	79.13	97.54	250.63	91.15	40.16	97.15	12.76

Statistical tests are conducted to verify the significance of the observations in tables 4.253. Table 4.254 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

TABLE 4.254: Credit: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.00					
FS	0.00	0.12	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.08	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

C. Wine (lowest structural complexity = NS1)

Table 4.255 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.255: Wine: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	281.00	597.00	299.00	373.00	237.00	231.00	175.00
μ	134.53	257.27	140.80	140.00	102.80	149.07	51.13
σ	41.24	99.12	48.15	75.67	41.52	22.72	31.52

Statistical tests are conducted to verify the significance of the observations in tables 4.255. Table 4.256 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.256: Wine: Statistical tests on final BS size

One-way ANOVA p-value = 0.03							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.36	0.00					
FS	0.38	0.00	0.48				
DSS	0.02	0.00	0.01	0.02			
HP	0.16	0.00	0.31	0.30	0.00		
NS	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP1 discovers the largest solution programs.

D. Segment (lowest structural complexity = NS1)

Table 4.257 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.257: Segment: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	277.00	689.00	995.00	383.00	323.00	207.00	149.00
μ	152.47	365.00	354.53	157.20	141.47	102.00	84.13
Min	69.32	113.63	181.14	75.45	63.54	67.45	19.22

Statistical tests are conducted to verify the significance of the observations in tables 4.257. Table 4.258 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.258: Segment: Statistical tests on final BS size

One-way ANOVA p-value = 0.03							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.42					
FS	0.41	0.00	0.00				
DSS	0.25	0.00	0.00	0.24			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.02	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP1 and BP2 discover the largest solution programs.

E. Vowel (lowest structural complexity = NS1)

Table 4.259 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.259: Vowel: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	431.00	587.00	987.00	229.00	363.00	263.00	107.00
μ	222.93	294.93	487.17	101.67	163.80	98.67	54.93
σ	65.42	91.23	155.55	35.24	53.14	39.24	13.16

Statistical tests are conducted to verify the significance of the observations in tables 4.257. Table 4.258 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.260: Vowel: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.02						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.01	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.41	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

E. Opt (lowest structural complexity = OF, NS1)

Table 4.261 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.261: Opt: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS
b	197.00	665.00	711.00	337.00	205.00	321.00	221.00
μ	84.20	508.15	555.94	144.33	106.87	99.67	88.13
σ	28.14	50.24	67.40	66.14	29.63	70.00	33.53

Statistical tests are conducted to verify the significance of the observations in tables 4.257. Table 4.258 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.262: Opt: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS
OF							
BP1	0.00						
BP2	0.00	0.13					
FS	0.00	0.00	0.00				
DSS	0.03	0.00	0.00	0.01			
HP	0.18	0.00	0.00	0.01	0.34		
NS	0.36	0.00	0.00	0.00	0.06	0.25	

The results indicate that OF and NS1 discover the smallest solution programs. Furthermore, BP1 and BP2 discover the largest solution programs.

Boolean function synthesis benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.41 shows the mean size of the best solution program found by GP. The mean size of the best solution program is also shown in table 4.263, whereby the lowest values achieved on each problem are highlighted. In turn, figure 4.42 shows the generational mean population size. In the figures and tables below, lower values indicate better performance of GP. All values are averaged over the 30 GP runs.

The results show that as in the symbolic regression and supervised classification domains, NS1 discovers the smallest solution programs on most of the problems in this domain. HP also discovers the smallest solution programs on the par-7, par-9 and mux-11 problems. Conversely, BP1 and BP2 discover the largest solution programs.

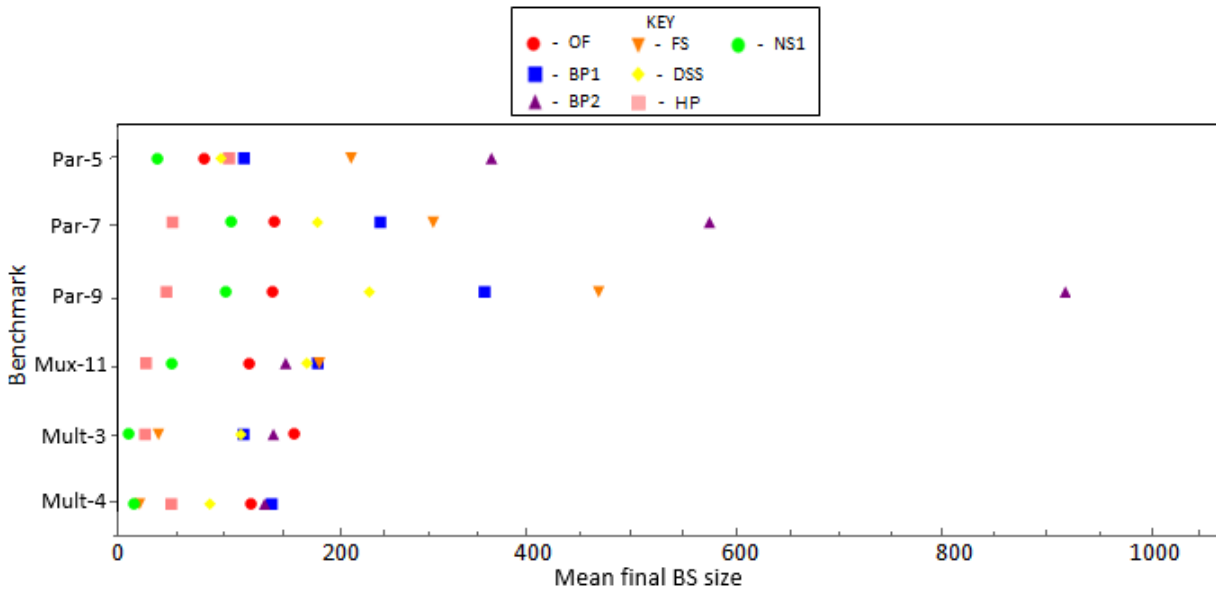


FIGURE 4.41: Boolean function synthesis benchmarks: Mean final BS size

TABLE 4.263: Boolean function synthesis benchmarks: Mean final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	83.27	116.87	375.80	221.73	97.13	104.07	34.40
Par-7	143.07	246.67	587.33	304.27	179.40	46.67	105.20
Par-9	140.93	355.47	924.00	466.47	237.33	41.47	100.53
Mux-11	120.83	179.59	152.27	180.87	170.33	24.00	45.40
Mult-3	159.34	116.34	141.73	34.63	114.11	23.32	9.21
Mult-4	123.38	140.41	134.25	18.63	87.72	45.60	14.48

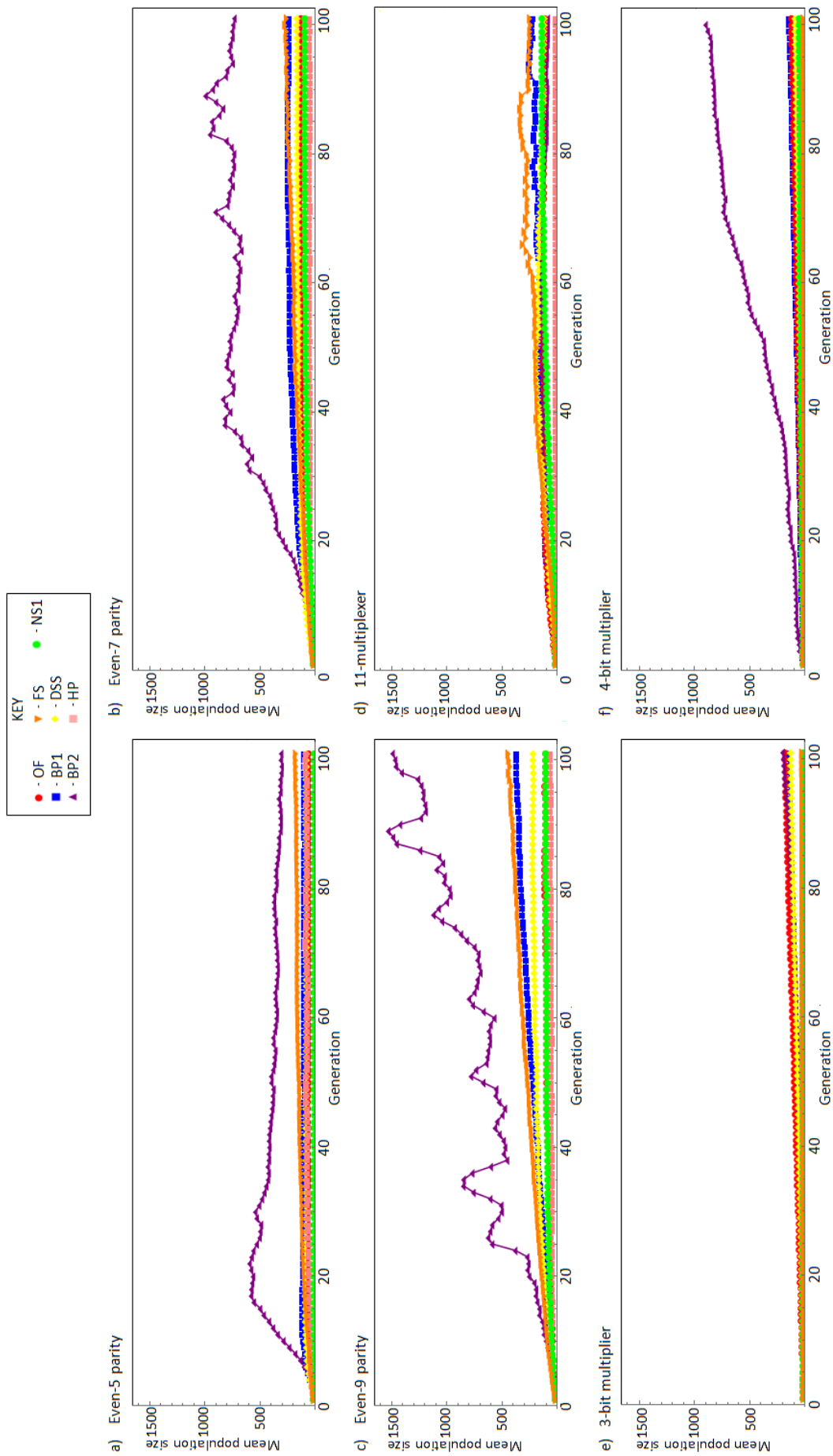


FIGURE 4.42: Boolean function synthesis benchmarks: Mean generational population size

Detailed results

The benchmarks are discussed in the following order: A) Even-5 parity, B) Even-7 parity, C) Even-9 parity, D) 11-multiplexer, E) 3-bit multiplier, and F) 4-bit multiplier.

A. Even-5 parity (lowest structural complexity = NS1)

Table 4.264 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.264: Even-5 parity: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	151.00	313.00	1019.00	827.00	153.00	305.00	67.00
μ	83.27	116.87	375.80	221.73	97.13	104.07	34.40
σ	23.33	60.66	193.42	171.91	18.65	62.13	11.41

Statistical tests are conducted to verify the significance of the observations in tables 4.264. Table 4.265 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.265: Even-5 parity: Statistical tests on final BS size

One-way ANOVA p-value = 0.05							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.01						
BP2	0.00	0.00					
FS	0.00	0.00	0.00				
DSS	0.06	0.06	0.00	0.00			
HP	0.05	0.20	0.00	0.00	0.28		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

B. Even-7 parity (lowest structural complexity = HP)

Table 4.266 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.266: Even-7 parity: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS
Max	329.00	499.00	4947.00	633.00	301.00	135.00	193.00
μ	143.07	246.67	587.33	304.27	179.40	46.67	105.20
Min	60.20	79.92	1150.21	87.02	77.45	33.23	35.32

Statistical tests are conducted to verify the significance of the observations in tables 4.266. Table 4.267 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that HP discovers the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

TABLE 4.267: Even-7 parity: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS
OF							
BP1	0.00						
BP2	0.01	0.02					
FS	0.00	0.04	0.04				
DSS	0.01	0.00	0.01	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS	0.01	0.00	0.00	0.00	0.00	0.00	

C. Even-9 parity (lowest structural complexity = HP)

Table 4.268 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.268: Even-9 parity: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	349.00	881.00	3901.00	1065.00	475.00	111.00	213.00
μ	140.93	355.47	924.00	466.47	237.33	41.47	100.53
σ	59.46	147.74	518.90	197.32	83.33	25.34	31.32

Statistical tests are conducted to verify the significance of the observations in tables 4.268. Table 4.269 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.269: Even-9 parity: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.00	0.01					
FS	0.00	0.01	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that HP discovers the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

D. 11-multiplexer (lowest structural complexity = HP)

Table 4.270 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.270: 11-multiplexer: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	247.00	580.00	360.00	462.00	450.00	86.00	176.00
μ	120.83	179.59	152.27	180.87	170.33	24.00	45.40
σ	41.11	99.46	59.28	64.28	89.47	17.28	38.24

Statistical tests are conducted to verify the significance of the observations in tables 4.270. Table 4.271 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.271: 11-multiplexer: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.05	0.16					
FS	0.00	0.48	0.09				
DSS	0.01	0.37	0.23	0.32			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.01	

The results indicate that HP discovers the smallest solution programs. Furthermore, BP1, BP2, FS and DSS discover the largest solution programs.

E. 3-bit multiplier (lowest structural complexity = NS1)

Table 4.272 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.272: 3-bit multiplier: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	230.00	190.33	329.67	60.00	171.67	42.67	23.67
μ	159.34	116.34	141.73	34.63	114.11	23.32	9.21
σ	29.33	22.67	56.33	9.43	15.67	7.33	3.00

Statistical tests are conducted to verify the significance of the observations in tables 4.272. Table 4.273 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.273: 3-bit multiplier: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.14	0.06					
FS	0.00	0.00	0.00				
DSS	0.00	0.40	0.05	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.00	0.00	0.00	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, OF and BP2 discover the largest solution programs.

E. 4-bit multiplier (lowest structural complexity = FS, NS1)

Table 4.274 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.274: 4-bit multiplier: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1
b	179.50	177.75	208.00	36.00	125.75	80.50	55.25
μ	123.38	140.41	134.25	18.63	87.72	45.60	14.48
σ	14.75	12.20	19.50	5.59	10.25	9.20	9.25

Statistical tests are conducted to verify the significance of the observations in tables 4.274. Table 4.275 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. $p\text{-val} < 0.05$) are highlighted in the table.

TABLE 4.275: 4-bit multiplier: Statistical tests on final BS size

One-way ANOVA p-value = 0.00							
Pairwise one-tail t-tests: p-values:							
	OF	BP1	BP2	FS	DSS	HP	NS1
OF							
BP1	0.00						
BP2	0.08	0.20					
FS	0.00	0.00	0.00				
DSS	0.00	0.00	0.00	0.00			
HP	0.00	0.00	0.00	0.00	0.00		
NS1	0.00	0.00	0.00	0.04	0.00	0.00	

The results indicate that FS and NS1 discover the smallest solution programs. Furthermore, OF, BP1 and BP2 discover the largest solution programs.

Path-finding benchmarks

An overview of the performance in the domain is presented. A detailed breakdown of the results follows.

Performance overview

Figure 4.43 shows the mean size of the best solution program found by GP. The mean size of the best solution program is also shown in table 4.276, whereby the lowest values achieved on each problem are highlighted. In turn, figure 4.44 shows the generational mean population size. In the figures and tables below, lower values indicate better performance of GP. All values are averaged over the 30 GP runs.

The results show that, as in the previous domains, FS and NS1 generally discover the smallest solution programs on the problems in this domain. Furthermore, BP1 and BP2 discover among the largest solution programs on most of the problems. OF and DSS discover the largest solution programs on the ant problem. In turn, BP2 discovers the largest solution programs on the tartarus problem, whereas BP2 and NS2 discover the largest solution programs on the deceptive tartarus problem.

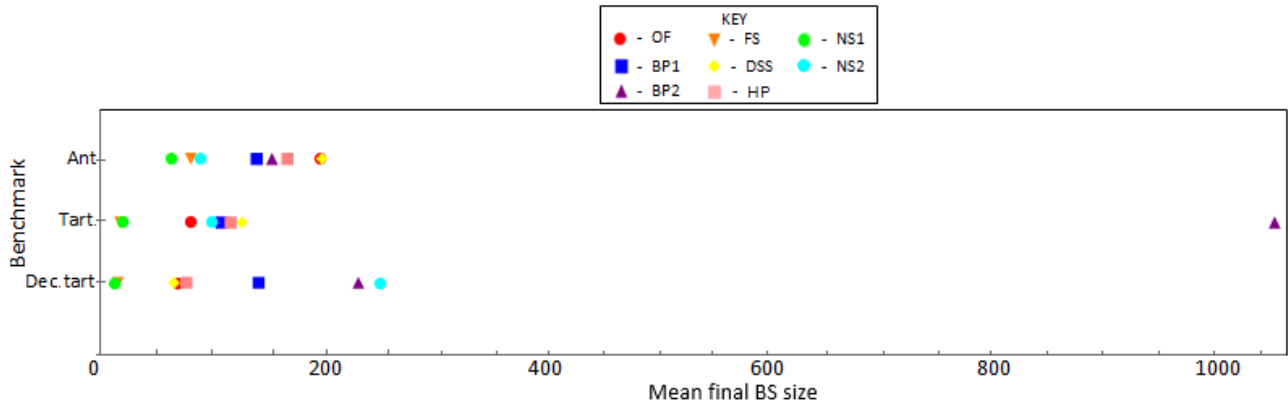


FIGURE 4.43: Path-finding benchmarks: Mean size of best solution

TABLE 4.276: Path-finding benchmarks: Mean size of best solution

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	194.13	136.70	149.20	82.00	195.97	162.10	65.73	90.83
Tart.	82.17	107.53	1056.67	19.57	124.50	115.10	21.73	99.20
Dec. tart.	71.07	138.23	224.73	17.70	68.30	78.37	15.43	243.10

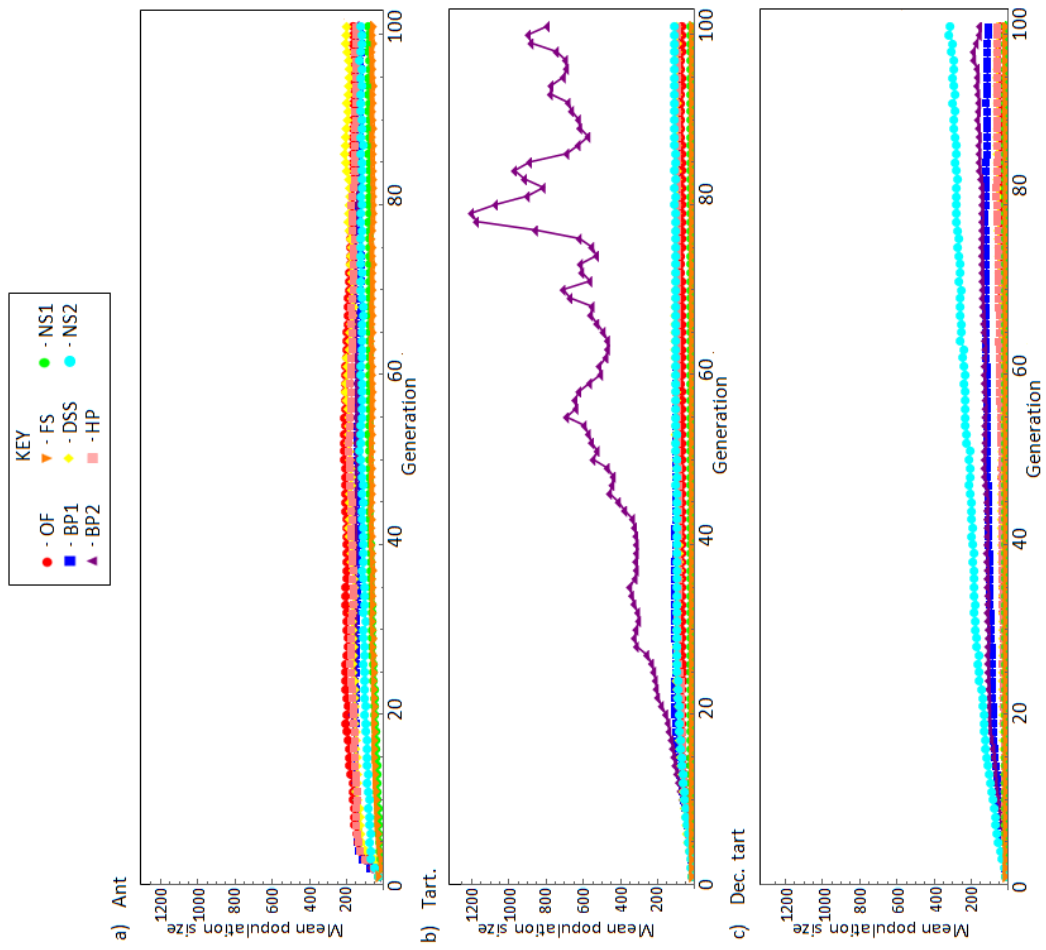


FIGURE 4.44: Path-finding benchmarks: Mean generational population size

Detailed results

The benchmarks are discussed in the following order: A) Artificial Ant, B) Tartarus, and C) Deceptive Tartarus.

A. Artificial Ant (lowest structural complexity = NS1)

Table 4.277 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.277: Artificial Ant: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	683.00	277.00	336.00	160.00	494.00	412.00	162.00	179.00
μ	194.13	136.70	149.20	82.00	195.97	162.10	65.73	90.83
σ	159.29	29.32	59.23	23.23	87.02	74.22	29.03	32.32

Statistical tests are conducted to verify the significance of the observations in tables 4.277. Table 4.278 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.278: Artificial Ant: Statistical tests on final BS size

One-way ANOVA p-value = 0.05								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
OF								
BP1	0.00							
BP2	0.00	0.23						
FS	0.00	0.00	0.00					
DSS	0.48	0.01	0.03	0.00				
HP	0.00	0.12	0.30	0.00	0.00			
NS1	0.00	0.00	0.00	0.05	0.00	0.00		
NS2	0.00	0.00	0.00	0.21	0.00	0.00	0.01	

The results indicate that NS1 discovers the smallest solution programs. Furthermore, OF and DSS discover the largest solution programs.

B. Tartarus (lowest structural complexity = FS, NS1)

Table 4.279 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.279: Tartarus: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	147.00	234.00	1750.00	65.00	223.00	274.00	60.00	193.00
μ	82.17	107.53	1056.67	19.57	124.50	115.10	21.73	99.20
σ	13.20	22.18	68.43	10.92	29.22	49.44	12.23	29.12

Statistical tests are conducted to verify the significance of the observations in tables 4.279. Table 4.280 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

The results indicate that FS and NS1 discover the smallest solution programs. Furthermore, BP2 discovers the largest solution programs.

TABLE 4.280: Tartarus: Statistical tests on final BS size

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS	
OF								
BP1	0.01							
BP2	0.01	0.02						
FS	0.00	0.00	0.01					
DSS	0.00	0.07	0.02	0.00				
HP	0.00	0.26	0.02	0.00	0.20			
NS	0.00	0.00	0.01	0.30	0.00	0.00		
NS2	0.02	0.21	0.01	0.00	0.00	0.06	0.00	

C. Deceptive tartarus (lowest structural complexity = FS, NS1)

Table 4.281 lists the best (b), mean (μ) and standard deviation (σ) of the final BS size. The fitness measures that discover the smallest solutions highlighted in the table.

TABLE 4.281: Deceptive tartarus: Final BS size

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
b	224.00	317.00	1418.00	77.00	218.00	221.00	62.00	593.00
μ	71.07	138.23	224.73	17.70	68.30	78.37	15.43	243.10
σ	42.12	51.44	299.32	19.23	43.22	39.23	12.28	91.22

Statistical tests are conducted to verify the significance of the observations in tables 4.281. Table 4.282 lists the p-values resulting from the statistical tests. The p-values that indicate statistical significance (at a significance level of $\alpha = 0.05$; i.e. p-val < 0.05) are highlighted in the table.

TABLE 4.282: Deceptive tartarus: Statistical tests on final BS size

One-way ANOVA p-value = 0.00								
Pairwise one-tail t-tests: p-values:								
	OF	BP1	BP2	FS	DSS	HP	NS1	
OF								
BP1	0.00							
BP2	0.00	0.04						
FS	0.00	0.00	0.00					
DSS	0.43	0.00	0.00	0.00				
HP	0.33	0.00	0.00	0.00	0.28			
NS1	0.00	0.00	0.00	0.28	0.00	0.00		
NS2	0.00	0.00	0.37	0.00	0.00	0.00	0.00	

The results indicate that FS and NS1 discovers the smallest solution programs. Furthermore, BP2 and NS2 discover the largest solution programs.

Summary of the structural complexity results

The results in this section show that FS and NS1 discover the smallest solution programs on most of the problems. The following general observations are made:

- i) BP consistently incurs high structural complexity on the tackled problems. BP favors solution programs that contain useful modules. The version of BP implemented in this study was prescribed at the origination of the BP paradigm in [160]. This version of BP does not impose any criteria to limit the size of the subtrees associated with useful modules; hence the useful modules may be large (or bloated) subtrees,

or even entire trees. The spread of such modules inevitably leads to increases in the code size. This effect is exacerbated in BP2, where the archive supplied mutation operator recombines the useful modules into the candidate solution programs in the evolving GP population. The indication is that a multi-objective fitness measure incorporating parsimony pressure should be used in BP, in order to mitigate the code growth associated with the fitness measure.

- ii) FS discovers the smallest solution programs on a number of the problems. Section 3.4 of chapter 3 established that FS mitigates bloat: FS penalizes candidate solution programs that solve the same fitness cases, mitigating genetic operations in neutral code regions and corresponding increases in code size. The results indicate that FS mitigates bloat on the tackled problems.
- iii) NS also reliably discovers the smallest solution programs on a number of the problems. Section 3.7 of chapter 3 established that NS mitigates bloat: neutral code inhibits a phenomenon that is rewarded in NS; behavioral change. Therefore, the selective pressure in NS perpetually discourages the growth of neutral code. The results indicate that NS mitigates bloat on the tackled problems.
- iv) Low structural complexity is not always associated with the best outcome. Low structural complexity may result from the failure of GP search to improve on the primitive initial population individuals. Examples of this are seen with HP achieving low quality and exhibiting low structural complexity in the Boolean function synthesis domain, as well as FS and NS1 achieving low quality and exhibiting low structural complexity on the tartarus and deceptive tartarus problems. On the other hand, the relatively high structural complexity incurred by BP1, BP2 and NS2 on the deceptive tartarus problem is due to the ability of these fitness measures to progress the search, while the other fitness measures make little (or no) improvement on the primitive initial population individuals.

4.3.5 Time taken

This section reports on the time taken to execute GP. A presentation of the results from each domain ensues.

Symbolic regression benchmarks

Table 4.283 lists the mean time taken (in seconds) to execute a single GP run; the shortest times incurred are highlighted in the table. The results in table 4.283 are averaged over the 30 GP runs.

TABLE 4.283: Symbolic regression benchmarks: Mean execution time (seconds)

	OF	BP1	BP2	FS	DSS	HP	NS1
Sextic	6.28	1930.82	4930.08	5.42	4.10	5.75	93.01
Nguyen	11.29	2863.12	5007.08	6.06	6.01	8.12	287.76
Pagie	18.85	2950.51	5163.54	13.99	9.45	11.21	506.27
Keijzer	1534.40	6751.90	8340.98	1033.87	428.01	518.34	3692.76
Vlad	1194.22	7838.33	8942.22	1008.79	758.96	900.76	4099.90
Dow	3342.02	17224.25	26563.93	3423.28	2028.23	2100.03	10716.49

The results show that DSS incurs the shortest execution times on the problems. FS incurs a comparable execution time with DSS on the sextic, Nguyen and Pagie problems. In turn, HP incurs a comparable execution with DSS on all the problems. Conversely, BP1 and BP2 are shown to incur the longest execution times on the problems. NS1 also incurs relatively long execution times.

Supervised classification benchmarks

Table 4.284 lists the mean time taken (in seconds) to execute a single GP run; the lowest times incurred are highlighted in the table. The results in table 4.284 are averaged over the 30 GP runs.

TABLE 4.284: Supervised classification benchmarks: Mean execution time (seconds)

	OF	BP1	BP2	FS	DSS	HP	NS1
Iris	10.41	338.22	304.88	11.39	9.69	7.79	17.81
Credit	159.84	11696.10	17205.50	388.27	53.01	119.12	682.45
Wine	21.08	2255.45	2263.54	25.23	22.82	36.58	56.19
Segment	39.37	2301.48	2309.74	39.09	41.95	29.30	56.76
Vowel	83.54	8838.33	18930.20	79.47	58.02	64.24	1089.81
Opt	290.09	327208.00	363563.89	480.53	456.37	192.93	2376.05

The results show that, as in the previous domain, DSS and HP incur short execution times on the problems. Furthermore, BP1 and BP2 incur the longest execution times on all problems, while NS1 also incurs relatively long execution times.

Boolean function synthesis benchmarks

Table 4.285 lists the mean time taken (in seconds) to execute a single GP run; the lowest times incurred are highlighted in the table. The results in table 4.285 are averaged over the 30 GP runs.

TABLE 4.285: Boolean function synthesis benchmarks: Mean execution time (seconds)

	OF	BP1	BP2	FS	DSS	HP	NS1
Par-5	5.71	40.71	317.34	7.45	2.74	5.15	3.36
Par-7	34.81	513.13	2638.61	43.56	34.83	16.17	569.43
Par-9	97.38	2495.73	12230.20	113.37	93.08	50.96	237.87
Mux-11	170.87	2455.20	2146.73	15.19	69.47	40.10	3945.62
Mult-3	34.51	187.63	317.47	34.66	36.79	14.48	818.82
Mult-4	351.02	1860.83	2850.85	873.87	300.28	94.38	3249.55

The results show that DSS incurs the shortest execution time on the par-5 problem. In turn, FS incurs the shortest execution time on the mult-11 problem, whereas HP incurs the shortest execution time on the remaining problems. Conversely, BP1, BP2 and NS1 incur long execution times.

Path-finding benchmarks

Table 4.286 lists the mean time taken (in seconds) to execute a single GP run; the lowest times incurred are highlighted in the table. The results in table 4.286 are averaged over the 30 GP runs.

The results show that BP1 incurs the shortest execution time on the ant problem. In turn, FS and NS1 incur the shortest execution time on the tartarus and deceptive tartarus problems. Conversely, BP2 incurs the longest execution time on the tartarus and deceptive tartarus problems.

TABLE 4.286: Path-finding benchmarks: Mean execution time (seconds)

	OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Ant	26.51	22.28	45.38	34.59	28.60	41.67	29.94	29.05
Tart.	22.06	55.59	189.98	15.83	31.70	33.04	18.60	38.92
Dec. tart	20.80	58.26	78.87	14.13	21.08	30.71	16.36	40.20

Summary of the time taken results

The results in this section show that DSS and HP, anticipated to minimize the GP execution time, largely achieves this. The following general observations are made:

- i) DSS and HP incur the shortest execution times on most of the benchmark problems. Sections 3.5.1 and 3.6.1 of chapter 3 established that DSS and HP evaluate only subsets of the complete fitness case set at a time, which enhances the evolution speed of GP. Hence a shorter time is taken to run GP.
- ii) BP1 and BP2 incur the longest execution times on most of the benchmark problems. Section 3.3.2 of chapter 3 established that in BP, the fitness evaluation involves the additional cost of modelling the program trace data. The time taken to model the program trace data is proportional to $|m| \times s$, where $|m|$ is the number of fitness cases in the training set and s is the number of nodes in the given candidate solution program. The structural complexity results in section 4.3.4 showed that in BP, the size of the candidate solution programs in the population increases considerably with the progress of GP. Therefore, significant time delays are inevitable with the progress of search in BP.
- iii) NS1 also incurs long execution times on a number of the problems. Section 3.7 of chapter 3 established that in NS, the fitness of a candidate solution program, i , is determined by calculating the pairwise distance between i and each of its K -nearest neighbours. Therefore, the time taken to compute i 's fitness is proportional to the value of K . It follows that the time taken to evaluate the fitness of the entire population is quadratic for values of K approaching the population size; that is, given a population of size N , and $k \approx N$, the time taken is proportional to $N \times N$. Hence in NS, there is a trade-off between the execution time, and the value of K used: higher values of K incorporate more information into the search, steering the population towards more novelty; nevertheless, high values of K are associated with costly fitness evaluations.
- iv) The time taken to execute GP also depends on a number factors that vary for the different benchmark problems. For example, achieving high success rates minimizes the time taken to run GP. This is seen with FS achieving high success rates and therefore incurring relatively short execution times on the 11-multiplexer, iris and even-5 parity problems. Also, BP1 achieves the high success rate on the ant problem, and as a result incurs the shortest execution time on the problem.

Bloat is another factor influencing the time taken to run GP. According to the literature [259], larger programs take more execution time than smaller ones. For all of the fitness measures analyzed in the study, the fitness score of a candidate solution program is determined through traversal (see chapter 3), whereby it takes a longer time to traverse a larger program. Hence the fitness evaluations are more expensive in bloated populations. As a result, the fitness measures that mitigate bloat also minimize the time taken to run GP.

4.4 Summary

This chapter compared 6 state of the art fitness measures: OF, BP, FS, DSS, HP and NS. An experiment was conducted to evaluate the effect that the different fitness measures have for different benchmark problems. The study looked at how the different fitness measures address each of the limitations they aim to overcome for different problems.

The results indicate that the fitness measures largely address the intended limitations. BP improves on the quality of the solution programs found by GP on modular problems. However, BP should be implemented with parsimony pressure, in order to minimize the code bloat characteristic of the fitness measure. This will in turn minimize the long execution times also characteristic of BP.

FS and DSS are shown to improve on the quality of the solutions found by GP on problems susceptible to premature convergence. A further observation made is that DSS improves on GP's generalization; this is attributed to the ability to mitigate overfitting. Furthermore, FS implicitly mitigates overfitting as a result the niching strategy used. This is whereby the FS niching strategy leads to solution programs that can solve diverse fitness cases. FS is also seen to mitigate bloat: by penalizing candidate solution programs that solve

the same fitness cases, FS curtails genetic operations in neutral code regions, and corresponding increases in code size.

DSS and HP are shown to minimize the GP execution time. This is due to the fitness measures evaluating only subsets of the complete training set of fitness cases on each generation. Nevertheless, the GP execution time is also influenced by the size of the candidate solution programs being evolved. Hence the fitness measures that mitigate bloat, such as FS, can also minimize the GP execution time.

NS demonstrates the capability improve on the solution quality on difficult problems where there is motivation for exploring the behavior space. Nevertheless NS's reliability is influenced by the choice of the behavior descriptor, as well as the size of the behavior space. The results also indicate that NS mitigates bloat on most problems, whereby bloat is maladaptive to the behavioral change promoted by the fitness measure. A further observation made is that NS incurs relatively long execution times. In NS, search is a trade-off between the number of K -nearest neighbours used to guide search and the time taken to run GP; higher values of K incorporate more information into the search, perpetuating more novelty; nevertheless, high values of K are associated with longer execution times.

The fitness measures do not always influence GP as intended. Rather, the NFL theorems apply: no one fitness measure achieves the best result on all problems with respect to a given criterion. The solution quality achieved depends on the properties of the given problem e.g. modularity, susceptibility to premature convergence, and whether or not there is motivation for exploring the behavior space in the case of NS; these properties differ for the different problems. In turn, generalization is influenced by the relationship between the training and test sets, which differs for the different problems. The extent to which a fitness measure influences the population diversity on a problem is influenced by whether or not the fitness measure suits the given problem: a fitness measure that does not suit the given problem may fail to improve on the quality of the primitive initial population individuals, hence retaining the low initial population semantic diversity and entropy on the problem. Importantly, the relationship between diversity and solution quality differs for the different problems, whereby high diversity is seen with high quality on some of the problems, while low diversity is seen with high quality on other problems. Another observation made is that HP and DSS generally achieve the best result with respect to minimizing the GP execution time. Nevertheless, short execution times are also seen with high success rates; furthermore, the fitness measures that mitigate bloat also minimize the time taken to run GP.

Overall, the problems tackled in the study exhibit different properties; hence different fitness measures suit the different problems. The ensuing chapters employ GA and GP to find the best fitness measure sequences and fitness measure combinations respectively for DFMGP. This is whereby GA and GP are used to approximate the most suitable fitness measures for problems and problem classes.

Chapter 5

Methodology

5.1 Introduction

This chapter sets out the methodology used to achieve the objectives of the thesis, described in chapter 1. The chapter is organized as follows. Section 5.2 presents a critical analysis of related literature. Section 5.3 presents the research methodology to be followed. Section 5.4 details how each objective will be achieved. Section 5.5 specifies the benchmark suite that will be used in the study. Section 5.6 details the hardware and software used to achieve the objectives. Finally section 5.7 summarizes the chapter.

5.2 Critical analysis of related literature

From the literature surveyed in chapter 3, as well as the experiment conducted in chapter 4, it is clear that the concept of a universal “best” fitness measure is a falsehood. Rather, different fitness measures suit different problems. Importantly, the performance of a fitness measure on a given problem depends on the properties exhibited by the tackled problem (e.g. modularity, susceptibility to premature convergence, deception), as well as the capability of the fitness measure in handling the challenges posed by the problem.

The experiment conducted in chapter 4 provided some guidelines that can be used to anticipate the performance of the fitness measures. For example, OF achieves high solution quality on trivial problems. Also, BP achieves high solution quality when useful modules can be found and exploited. In turn, FS is shown to improve on the quality of the solution programs found by OF-GP on problems prone to local optima. As a final example, a well-configured NS improves on the quality of the solution programs found by OF-GP on deceptive problems, such as the deceptive tartarus problem from the path-finding domain. The information summarized above, whereby problem properties are matched to suitable fitness measures, is however only useful to a certain extent. In real world scenarios, GP practitioners are seldom aware of the properties underlying a newly encountered problem. In this case, it is difficult to prescribe suitable fitness measures for such problems a priori.

Importantly, the literature also anticipates that applying different fitness measures at different points of the GP problem solving process should be more effective than applying a single fitness measure individually throughout the algorithm [6, 15]. McKay [6, 15] anticipates this idea, but does not conduct a thorough investigation to this effect. The work done by McKay [6, 15] is motivated by the observation that fitness sharing (FS) measures constantly enforce the maintenance of niches in the GP population, inhibiting widespread convergence of the population in later generations. Nevertheless, widespread population convergence is often necessary in later generations, in order to involve more search points in exploiting the promising solution programs discovered by the search [6, 15]. In this vein, McKay [6, 15] prescribes a ramped approach, whereby FS is applied in the initial 25% of the GP generations, OF is applied in the last 25% of the GP generations, and in the intermediate GP generations, fitness is calculated as a linear ramp between FS and OF [6, 15]. The ramped approach is shown to evolve higher-quality solution programs on different problems when compared to both FS and OF applied individually throughout the course of GP [6, 15]. This result is justified by the argument that

the ramped approach strikes a balance between mitigating premature convergence in the preliminary GP generations, and subsequently allowing widespread convergence and exploitation in later GP generations [6, 15]. McKay's ramped approach [6, 15] may also be applicable to other fitness measures. For example, Mouret [204] argues that NS is incapable of exploitation. NS inhibits population convergence, because the fitness measure maintains the same level of selective pressure in favor of diversification, even when near-optimal candidate solution programs are found [204]. In this scenario, a ramped approach may be useful with respect to facilitating exploitation in later generations, when promising candidate solution programs have been discovered by the NS. A potential drawback of the ramped approach is that the exploration and exploitation phases may span different GP generations for different problems, such that the arbitrary partitioning of generations (first 25% - FS/NS; subsequent 50% - ramped; subsequent 25% - OF) does not achieve consistent results on different problems. Also, based on the results obtained in the previous chapter, it is unlikely that two fitness measures (e.g. FS and OF, or NS and OF) will be suited to all problems in the universe of GP problems. Hence a ramped approach that uses only the listed measures may not achieve consistent results on all problems. Despite the promising results obtained in the work of McKay [6, 15], no further research exists in the literature with respect to applying different fitness measures on the different GP generations.

Based on the above arguments, the main contribution of this research is the proposal of dynamic fitness measure GP (DFMGP). DFMGP involves the use of different fitness measures on the different GP generations, whereby the fitness measures are drawn from a rich database of measures. Furthermore, in the proposed DFMGP approach, rather than arbitrarily applying different fitness measures on the different GP generations, a higher (or meta-) level search algorithm is used to approximate the best fitness measure to use on each generation for the given problem (or set of problems). The term dynamic fitness measure (DFM) refers to the alternation of fitness measures on the DFMGP generations. A high-level search algorithm is needed to search the space of DFMs for DFMGP, because selecting the best fitness measure (or fitness measure combination) to apply on each GP generation is in itself a combinatorial optimization problem, due to a combinatorial explosion of possible fitness measure sequences and combinations. Importantly, the automated selection and/or combination of the fitness measures achieved by the high-level search algorithm will also remove the need for manual configuration of the fitness measures, and in so doing reduces the role of the human expert in the problem solving process. The study also investigates the reusability of the evolved DFMs. DFMs are evolved for different problem classes, whereby the purpose is to evolve problem solvers that can generalize within a problem class. Furthermore, DFMGP is also applied to complex, real-world problems. Here, DFMs evolved by training on the problem classes are tested on real-world problems from the same problem domain. The idea is to train on simpler and less complex problems that will not take as much time to train on, and subsequently employ the evolved DFMs on the more complex problems. Overall, the implication of reusable DFMs is that the total time necessary for the derivations is reduced.

The objectives as stated in chapter 1 for the thesis are: 1) apply GAs for evolving DFMs for DFMGP, 2) apply GP for evolving DFMs for DFMGP, 3) compare the performance of DFMGP with the conventional GP approach, 4) compare the performance of GAs and GP in evolving DFMs, 5) assess the reusability of the evolved DFMs, and 6) analyse the best performing DFMs to identify the fitness measures that are most useful in the different phases of search for different problems.

5.2.1 Justification of the GA approach for evolving DFMs for DFMGP

A GA approach will be used to evolve the DFMs for DFMGP. GAs have proven to be effective for higher (or meta-) level search. For example, in the study conducted by Dioşan and Oltean [17], GAs are used to select which genetic operator to apply at each point in an EA. The GA approach proposed by Dioşan and Oltean [17] evolves the sequence in which crossover, mutation and selection operators are applied an EA, while the candidate EAs attempt to solve the given problem at the lower level [17]. The GA approach in [17] is shown to evolve EAs that perform on par with, and in some cases even better than, manually designed standard EA approaches on a number of problems. GAs have also been used for parameter tuning of EAs [18, 26, 29]. For

example, in the study conducted by Brain and Addicoat [29], a GA approach is used to search the space of quantitative parameters for a lower-level GA. The GA approach proposed in [29] is shown to derive optimal parameters for the lower-level GA on a complex real-world problem.

In the current study, a GA approach will be used to select the best fitness measure to apply on each generation of DFMGP for the given problem (or set of problems). The GA individuals will be represented using a direct value encoding. In section 2.2.2 of chapter 2, it was established that more recent studies on GAs have employed a direct value encoding, rather than a binary encoding, because the former type of encoding offers an intuitive representation of the search space, and a simplified GA implementation, whereby pre-processing and post-processing steps are not required. The direct value encoding used in the current study is described in detail in section 7.3 of chapter 7. The GA approach will also use a generational control model as it will be a good way to establish a basic performance of the approach. It is a well known control model and does provide a level of reliability.

When deriving DFMs for a given problem, the fitness of each GA individual will be measured as the best solution quality achieved by running DFMGP applying the corresponding DFM (i.e. the DFM represented by the individual) on the problem. In chapter 4 of the study, different fitness measures were compared with respect to varied criteria (i.e. the solution quality achieved, generalization capability, population diversity etc.). Nevertheless, the fundamental aim of GP search is to find high quality solutions [1]. Therefore, at the inception of DFMGP in the current study, the aim is to improve on the solution quality. This informs the choice of the solution quality as the sole criterion optimized by the GA. Note that in the course of determining the fitness of a GA individual, it is likely that successive runs of DFMGP applying the corresponding DFM on a given problem should generate completely different solution programs, due to the stochastic nature of GP [1], and hence DFMGP. In this regard, DFMGP applying the corresponding DFM is executed a total of r times, and the fitness of the GA individual is measured as the mean best solution quality achieved over the r runs on the given problem; here, the value of r is set at 5. The value of r is kept low to minimize the computational cost involved in deriving the DFMs; also, in preliminary experiments, increasing the value of r did not yield significant improvement in the derived DFMs.

A further configuration employed by the GA will be tournament selection, rather than fitness proportionate selection, so that the tournament size parameter used in tournament selection can be tuned to mitigate premature convergence [32]. The crossover operator used by the GA will be one-point crossover, because it is found to be effective for other higher (or meta-) level GA approaches in different problem domains [17, 26, 29]. Furthermore, point mutation will be used, because it is also found to be effective for other higher (or meta-) level GA approaches [17, 26, 29]. Importantly, additional genetic operators will be defined during development, which together with the mutation operator will play the role of restoring lost diversity and mitigating premature convergence of the GA. Chapter 7 of the thesis describes the GA approach in detail.

5.2.2 Justification of the GP approach for evolving DFMs for DFMGP

A GP approach will also be used to evolve the DFMs for DFMGP. In the literature, GP has been used at the higher (or meta-) level to construct new EA components from the atomic elements that make up existing components [19, 20]; here, a component is an element that can be easily isolated in the EA structure [19–21]: for instance, the fitness measure used, the selection method used and the crossover and mutation operators used are all examples of EA components. An example is seen in the work of Edmonds [19], where a GP approach is used to evolve new crossover operators for a lower-level GP. The GP approach proposed in [19] constructs new crossover operators from the atomic elements that make up existing crossover operators. The GP approach in [19] is shown to evolve crossover operators that improve on the quality of the solution programs found by a lower-level GP on the odd 4-parity problem from the Boolean function synthesis domain. Edmonds [19] argues that the GP approach proposed in the study is a general technique expected to improve on the performance of a lower-level GP on different problems. As a further example, in the study by Dioşan and Oltean [20], GP is used to evolve new crossover operators for GA. In [20], a training problem is used to evolve the crossover

operators; subsequently GA is executed on unseen problems from the same problem domain using the evolved crossover operators. The results in [20] show that the proposed GP approach evolves new crossover operators that perform better than human-designed crossover operators on unseen problems. Overall, the literature demonstrates that GP has proven to be an effective approach for meta-level search. Furthermore, GP can be used to construct new EA components from the atomic elements that make up existing components.

In the current study, a GP approach will be used to create a new function for the DFMGP fitness measure, whereby the new function is a combination of existing fitness measures, in order to suit the given problem (or set of problems). The GP approach will employ an abstract syntax tree (AST) representation. AST-GP is the canonical GP approach proposed by Koza in [1]; it is a well-known approach and hence will be a good way to establish a basic performance of the GP approach. The aim is for the GP approach to evolve arithmetic and logical combinations of existing fitness measures. Hence the GP individuals being evolved should arithmetically combine existing fitness measures, and also employ logic with respect to selecting the best combination of fitness measures to apply on each DFMGP generation. Arithmetically combined fitness measures are anticipated to be advantageous based on the literature [66, 72]. For example, in [66] (and in this study), the behavioral fitness of a solution program is calculated as an arithmetic combination of the OF score and two terms that quantify the usefulness of the solution's subprograms; the previous chapter showed that the behavioral fitness measure (BP) improves on the performance of OF-GP on modular problems. Another example of arithmetically combined fitness measures is novelty-fitness aggregation, proposed as an NS variant for GP in [72]. In novelty-fitness aggregation, the fitness measure used is a weighted sum of the OF and novelty scores [72]. The purpose of novelty-fitness aggregation is to incorporate the search objective into the novelty search, because complete abandonment of the objective may cause the NS to waste time evaluating candidate solution programs with little or no relevance to the objective [72, 202]. Novelty-fitness aggregation does not achieve performance gain over OF-GP in [72], but this may be due to the NS behavior descriptor used in [72]. A further aspect of the syntax of the GP individuals being evolved is to incorporate logical selection of the arithmetic combinations of fitness measures based on the DFMGP generation. Logical selection of the fitness measures based on the DFMGP generation is anticipated to be advantageous because of the performance of McKay's ramped fitness approach [6, 15], whereby employing different fitness measures on the different generations of GP outperforms the standard GP approach of applying the same fitness measure individually throughout the algorithm. The above discussion indicates that the syntax of the GP individuals will be comprised of arithmetic and logical expressions. For this reason, the GP approach will employ a strongly-typed GP representation, in order to handle the different primitive types required. The strongly-typed representation used is described in detail in section 8.3 of chapter 8.

The GP approach will use a generational control model; it is a commonly used control model for GP. Also, the ramped half-and-half method will be used to produce the GP initial population; the ramped half-and-half method is argued to improve on the diversity of the initial population, due to the ability to generate a wide variety of tree shapes and structures [1]. The fitness of the GP individuals will be determined in the same way as in the GA approach, whereby fitness is measured as the mean best solution quality achieved over 5 runs of DFMGP applying the corresponding DFM on the given problem. A further configuration employed by the GP will be tournament selection, so that the tournament size used can be tuned to mitigate premature convergence. The genetic operators used will be standard GP crossover and mutation. Furthermore, additional genetic operators will be defined during development, which together with the mutation operator will play the role of restoring lost diversity and mitigating premature convergence of the GP. The GP initial population creation routine and genetic operators will be modified so as to facilitate typing. Chapter 8 of the thesis describes the GP approach in detail.

5.3 Research methodology

The objectives of the research will be achieved using the proof by demonstration methodology [260]. This research methodology for computer science requires the development of a single approach which is iterated upon to achieve the stated objective(s). First an initial approach is developed. The initial approach is based on an analysis of the literature. Subsequently, the initial approach is improved upon using iterative refinement, whereby the reasons identified for failure on each iteration form input to the refinement process of the next iteration. For each iteration, the changes to the approach will be based on testing. Each revision of the approach is evaluated in terms of the objectives of the problem to be solved. The formulated approach should be produced to meet the objectives of the research question. Iterative improvement occurs until further improvements are not going to provide significant improvement for the work required.

In order to analyse the developed approaches data will be collected. This will be done by measuring the performance of the approaches on a suite of GP benchmark problems. Statistical tests will be used to establish the significance of the observations made. Finally the data collected and observations made will be presented in a conclusion.

5.4 Objectives

This section describes how the research methodology will be implemented in order to achieve the objectives of this thesis. Objectives one and two are similar and approached in the same manner; this methodology will be discussed in section 5.4.1. Objectives three, four, five and six require the completion of objectives one and two. Objectives three, four and five are discussed in section 5.4.2. Finally, objective six is discussed in section 5.4.3.

5.4.1 Objectives one and two

The first objective set in chapter 1 is to apply GAs for evolving DFMs for DFMGP. The second objective set in chapter 1 is to apply GP for evolving DFMs for DFMGP. A brief discussion of the objectives is given below, after which the details of measurements used for analysis of the objectives will be discussed.

Overview of the objectives

Objectives one and two will be achieved through the proof by demonstration methodology. First an initial approach will be developed for each objective; descriptions of the initial approaches for objectives one and two can be found in sections 5.2.1 and 5.2.2 respectively. The two developed approaches will be tested using the benchmark problems specified in table 5.1 of section 5.5. The developed approaches will be refined until further improvements are not going to provide significant improvement for the work required. The refinements made to both approaches will be similar as they are both evolutionary algorithms as described in chapter 2. Refinements will include looking at the initial population creation, the genetic operators, the selection method, as well as tuning quantitative parameters such as the population size, the genetic operator probabilities (or application rates) and the tournament size used if the selection method employed is tournament selection.

Measurements for analysis of the objectives

Objectives one and two require analysis of the performance of the GA and GP approaches. A key factor that will be used to measure the performance of the GA and GP approaches is the solution quality achieved. Recall that at the inception of DFMGP in the current study, the aim is to improve on the solution quality. The GA and GP approaches will be used to derive DFMs for each of the benchmark problems specified in table 5.1 of section 5.5. Each approach will be executed 30 times for each of the given problems. A total of 30 runs are conducted due to the stochastic nature of GA/GP, whereby each run of the GA/GP approach on a given

problem should generate a different DFM. For each run, the best-of-run individual (i.e. the DFM with the best fitness score found over the course of the run) is retained. The performance of the GA/GP approach on a given problem is measured as the average performance achieved by the best-of-run individuals over the 30 runs; this is the mean fitness score of the best-of-run individuals over the 30 runs: the calculation of the fitness scores is based on the solution quality achieved, and was described in sections 5.2.1 and 5.2.2 for the GA and GP approaches respectively. Overall, for both the GA and GP approaches, the average performance achieved will be measured for all the given problems. The iterative refinements made on the approaches aim to improve on the performance achieved on the problems.

5.4.2 Objectives three, four and five

The third objective of the research is to compare the performance of DFMGP with the conventional GP approach. The fourth objective is to compare the performance of GAs and GP in evolving DFMs. Lastly, the fifth objective is to assess the reusability of the evolved DFMs. Objectives three and four are concerned with performance comparisons. Objective five is also concerned with performance comparisons, whereby a DFM is said to be reusable if DFMGP applying the DFM performs better the conventional GP approach on unseen problems. A brief discussion of the objectives is given below, after which the details of measurements used for analysis of the objectives will be discussed. Lastly, the statistical tests used to determine the significance of the observations made will be discussed.

Overview of the objectives

To achieve objective three, the performance of DFMGP applying the DFMs derived by the GA approach (abbreviated as $DFMGP_{GA}$) is compared to that of standard GP on each of the benchmark problems specified in table 5.1 of section 5.5. The GA approach is executed 30 times on each of the problems. For each problem, the best DFM found over the 30 runs of the GA is retained. The performance of $DFMGP_{GA}$ applying this DFM is compared to that of standard GP over 30 runs of $DFMGP_{GA}$ and standard GP on the same problem used to derive the DFM. Importantly, the total computational cost of using the GA approach to derive a DFM and subsequently running $DFMGP_{GA}$ applying the derived DFM on a given problem is much higher than simply running standard GP on the given problem; hence the total time taken do the former is also compared to the time taken to do the latter for each of the benchmark problems specified in table 5.1. This comparison of the time taken by the approaches demonstrates the high computational cost of the DFM derivations, and hence justifies the investigation into the reusability of the DFMs, addressed in objective five. The above steps are repeated to compare the performance of DFMGP applying the DFMs derived by the GP approach (abbreviated as $DFMGP_{GP}$) to that of standard GP in the same way.

To achieve objective four, the performance of $DFMGP_{GA}$ is compared to that of $DFMGP_{GP}$ on each of the benchmark problems specified in table 5.1 of section 5.5. The DFMs are first derived for each problem, whereby for both the GA and GP approaches, the best DFM found over 30 runs is retained. For each problem, the performance of $DFMGP_{GA}$ and $DFMGP_{GP}$ applying this DFM is compared over 30 runs of $DFMGP_{GA}$ and $DFMGP_{GP}$ on the same problem used to derive the DFM. The total time taken to derive the DFMs and subsequently run DFMGP applying the derived DFM is also compared for the GA and GP approaches.

Two aspects are involved in achieving objective 5. The first aspect is to assess the reusability of the evolved DFMs for problem classes. Table 5.2 of section 5.5 specifies the problem classes used for this purpose. The GA approach is trained on a subset of problem instances from a given problem class, the training set. When training on a problem class, the fitness of a GA individual is measured as the mean best solution quality achieved by $DFMGP_{GA}$ applying the corresponding DFM (i.e. the DFM represented by the individual) on the problems in the training set; here, the best solution quality achieved on a single run of $DFMGP_{GA}$ is measured for each problem, and subsequently the fitness of the GA individual is measured as the average of these values over all the problems in the training set. In the fitness evaluation described here, a single run of

$DFMGP_{GA}$ is conducted for each problem in the training set in order to minimize the total computational cost involved in deriving the DFMs; also, in preliminary experiments, increasing the number of $DFMGP_{GA}$ runs conducted for each of the problems was found not to yield significant improvement in the derived DFMs. Importantly, $DFMGP_{GA}$ is run r times (once for each problem) in the course of fitness determination, which compensates for the stochastic nature of $DFMGP_{GA}$. The GA approach is run 30 times on the problems in the training set. Subsequently, the best DFM found over the 30 runs is retained. This DFM is tested on a set of unseen problem instances from the same problem class, the test set. The performance of $DFMGP_{GA}$ applying the DFM is compared to that of standard GP over 30 runs of the approaches on each of the unseen problems. The above steps are repeated to assess the reusability of the DFMs evolved by the GP approach for problem classes in the same way.

The second aspect of objective 5 is to assess the reusability of the evolved DFMs for complex real-world problems. Table 5.3 of section 5.5 specifies the real-world problems used for this purpose. After the GA approach has been used to derive DFMs for the problem classes, $DFMGP_{GA}$ applying the DFMs is executed on unseen real-world problems, whereby for each problem class, the unseen problems tested come from the same problem domain as the problems in the class. The performance of $DFMGP_{GA}$ is compared to that of standard GP over 30 runs of the approaches on each of the unseen problems. The above steps are repeated to assess the reusability of the DFMs evolved by the GP approach for real-world problems in the same way.

Measurements for analysis of the objectives

This section describes a series of experiments conducted to achieve objectives three, four and five. From the overview of the objectives, it is clear that the steps that will be undertaken to assess the GA approach and $DFMGP_{GA}$ are similar to those that will be used to do the same for the GP approach and $DFMGP_{GP}$. In this regard, two main experiments will be conducted:

1. Experiment 1: This experiment assesses the GA approach and $DFMGP_{GA}$ to determine:
 - (a) The effectiveness of $DFMGP_{GA}$ compared to the conventional GP approach.
 - (b) The reusability of the derived DFMs within problem classes.
 - (c) The reusability of the derived DFMs on real world problems.
2. Experiment 2: This experiment has a similar structure to experiment 1 and assesses the GP approach and $DFMGP_{GP}$ to determine:
 - (a) The effectiveness of $DFMGP_{GP}$ compared to the conventional GP approach. The effectiveness of the GP approach and $DFMGP_{GP}$ is also compared to that of the GA approach and $DFMGP_{GA}$.
 - (b) The reusability of the derived DFMs within problem classes.
 - (c) The reusability of the derived DFMs on real world problems.

Experiments 1 and 2 are detailed below.

Experiment 1

The experiment is described in terms of the assessments made.

a) Testing the effectiveness of $DFMGP_{GA}$

The performance of $DFMGP_{GA}$ applying the DFMs derived by the GA approach is compared to that of standard GP applying the fitness measures individually to solve the same problem on each of the benchmark problems specified in table 5.1 of section 5.5. Furthermore, a control experiment is implemented, whereby randomly generated DFMs (obtained by the GA initial population creation procedure described in section 7.4 of chapter 7) are used for DFMGP: the control, abbreviated as $DFMGP_{RD1}$, is used to establish whether the

GA evolution is needed, or if generating random DFMs for DFMGP is sufficient with respect to obtaining a performance improvement over canonical GP. The number of random DFMs evaluated in the control is the same as the total number of DFMs evaluated by the GA. The above-mentioned approaches are compared on the benchmark problems listed in table 5.1. For each of the problems tackled, $DFMGP_{GA}$, $DFMGP_{RD1}$ and canonical GP will be compared based on the mean best solution quality (that is, the mean best raw OF score) achieved over 30 runs on the given problem.

b) Testing the reusability of the derived DFMs within problem classes

To determine the reusability of the DFMs within problem classes, the GA is trained on a subset of problem instances from a given problem class, the training set; here, the GA calculates the fitness of a candidate DFM as the mean best solution quality yielded over the training instances, as shown in equation 5.1.

$$F(i_{DFM}) = \frac{\sum_{j=1}^m OF(i_{DFM}(best_j))}{m} \quad (5.1)$$

whereby:

1. $F(i_{DFM})$ represents the fitness of i_{DFM} .
2. i_{DFM} is a candidate DFM.
3. $OF(i_{DFM}(best_j))$ represents the quality (i.e. the raw OF score) of the best solution program found on the j^{th} problem in the training set. $OF(i_{DFM}(best_j))$ is averaged over the m problems in the training set.

The DFM derived by the GA during training is tested on a set of unseen problem instances from the same problem class, the test set. Table 5.2 of section 5.5 shows the problem classes tackled in this respect. To assess the reusability of the derived DFMs within a problem class, the performance of $DFMGP_{GA}$ applying the DFMs on each test instance from the class is compared that of canonical GP applying each of the fitness measures individually on the same test instance. The performance of $DFMGP_{RD1}$ on the test instances is also evaluated. For each test instance listed in table 5.2, $DFMGP_{GA}$, $DFMGP_{RD}$ and canonical GP will be compared based on the mean best solution quality achieved over 30 runs on the given problem.

c) Testing the reusability of the derived DFMs on real world problems

To determine the reusability of the DFMs on real world problems, $DFMGP_{GA}$ is applied in real-world scenarios, as summarized in table 5.3 of section 5.5. After the DFMs have been evolved for the problem classes, $DFMGP_{GA}$ applying the DFMs is executed on real-world problems from the same problem domain. In addition, a more general DFM is evolved for each problem domain by training the GA on the combined set of all the training problems from the problem classes in the domain. $DFMGP_{GA}$ applying this DFM is also executed on the real-world problems from the problem domain. For each of the real-world problems listed in table 5.3, $DFMGP_{GA}$, $DFMGP_{RD1}$ and canonical GP will be compared based on the mean best solution quality achieved over 30 runs on the given problem.

Experiment 2

The experiment is described in terms of the assessments made.

a) Testing the effectiveness of $DFMGP_{GP}$

The performance of $DFMGP_{GP}$ applying the DFMs derived by the GP approach is compared to that of standard GP applying the fitness measures individually to solve the same problem on each of the benchmark problems specified in table 5.1 of section 5.5. Furthermore, a control experiment is implemented, whereby randomly generated DFMs (obtained by the GP initial population creation procedure described in section 8.4 of chapter 8) are used for DFMGP: the control, abbreviated as $DFMGP_{RD2}$, is used to establish whether the GP evolution is needed, or if generating random DFMs for DFMGP is sufficient with respect to obtaining a

performance improvement over canonical GP. The number of random DFMs evaluated in the control is the same as the total number of DFMs evaluated by the GP. The above-mentioned approaches are compared on the benchmark problems listed in table 5.1. $DFMGP_{GP}$ is also compared to $DFMGP_{GA}$ on the problems. For each of the problems tackled, $DFMGP_{GP}$, $DFMGP_{GA}$, $DFMGP_{RD2}$ and canonical GP will be compared based on the mean best solution quality (that is, the mean best raw OF score) achieved over 30 runs on the given problem.

b) Testing the reusability of the derived DFMs within problem classes

To determine the reusability of the DFMs within problem classes, the GP is trained on a subset of problem instances from a given problem class, the training set; here, the GP calculates the fitness of a candidate DFM as the mean best solution quality yielded over the training instances in the same way as shown in equation 5.1. The DFM derived by the GP during training is tested on a set of unseen problem instances from the same problem class, the test set. Table 5.2 of section 5.5 shows the problem classes tackled in this respect. Subsequently, the performance of $DFMGP_{GP}$ applying the DFMs on each test instance from the class is compared that of canonical GP applying each of the fitness measures individually on the same test instance. The performance of $DFMGP_{RD2}$ on the test instances is also evaluated. Also, $DFMGP_{GP}$ is also compared to $DFMGP_{GA}$ on the problems. For each test instance listed in table 5.2, $DFMGP_{GP}$, $DFMGP_{GA}$, $DFMGP_{RD2}$ and canonical GP will be compared based on the mean best solution quality achieved over 30 runs on the given problem.

c) Testing the reusability of the derived DFMs on real world problems

To determine the reusability of the DFMs on real world problems, $DFMGP_{GP}$ is applied in real-world scenarios, as summarized in table 5.3 of section 5.5. After the DFMs have been evolved for the problem classes, $DFMGP_{GP}$ applying the DFMs is executed on real-world problems from the same problem domain. In addition, a more general DFM is evolved for each problem domain by training the GP on the combined set of all the training problems from the problem classes in the domain. $DFMGP_{GP}$ applying this DFM is also executed on the real-world problems from the problem domain. For each of the real-world problems listed in table 5.3, $DFMGP_{GP}$, $DFMGP_{GA}$, $DFMGP_{RD2}$ and canonical GP will be compared based on the mean best solution quality achieved over 30 runs on the given problem.

Hypothesis testing

For all the performance comparisons conducted, the mean solution quality achieved by each approach over 30 runs will be the test statistic used for statistical hypothesis testing. A test statistic is a numerical summary of the data collected that reduces the data to a single value used to perform the hypothesis test [256]. A Z-test will be used to determine the statistical significance of results obtained. A Z-test is a statistical test used to determine whether two population means are different [256]. A Z-test assumes a normal distribution of the test statistic. The distribution of the test statistic is approximately normal given a large enough sample size; as a rule of thumb, a sample size of 30 is considered to be large enough to meet this criteria [256]. This argument justifies the calculation of the test statistic over 30 runs, as described in experiments 1 and 2 in the previous section.

Testing for a statistically significant result requires the formulation of a null hypothesis and an alternative hypothesis. The hypotheses test if there is a difference in the test statistics (i.e. the mean solution quality achieved); the null hypothesis is that there is no difference and the alternative is that there is a difference:

$$\begin{aligned} \text{Null hypothesis: } H_0 &: \mu_A = \mu_B \text{ or} \\ &H_0 : \mu_A - \mu_B = 0. \\ \text{Alternative hypothesis: } H_a &: \mu_A > \mu_B. \end{aligned}$$

The Z-tests are used to calculate a Z-score. This Z-score is converted to a p-value by looking up the corresponding p-value in a Z-table [256]. Subsequently, the null hypothesis (H_0) is rejected if the p-value is less than or equal to a small, fixed threshold value, α , referred to as the level of significance. In this thesis, the level of significance will be set to 5% (or 0.05), such that H_0 is rejected when $p_val \leq 0.05$, where p_val is the p-value derived from the observations.

5.4.3 Objective six

The sixth objective of the research is to analyse the best performing DFMs to identify the fitness measures that are most useful in the different phases of search for different problems. A brief discussion of the objective is given below, after which the details of measurements used for analysis of the objective will be discussed.

Overview of the objective

The DFMs derived for the problem classes are analyzed to identify the fitness measures that suit the different phases of search for the problem classes. Also, the more general DFMs derived for the problem domains are analyzed to identify the different fitness measures that suit the different phases of search for the problem domains. Recall that the GA approach is used to derive fitness measure sequences, while the GP approach is used to derive fitness measure combinations. The derived fitness measure sequences and combinations are analyzed separately, and also compared to determine if there are similarities between the two with respect to the fitness measures selected for a given set of problems.

Measurements for analysis of the objective

The DFMs derived by the GA approach will be analyzed. The DFM analyzed for a given problem class is the best DFM found over 30 runs of the GA approach on the training problems from the class. Also, the DFM analyzed for a given problem domain is the best DFM found over 30 runs of the GA approach on the training problems from the domain. The GA chromosomes representing these DFMs will be reported and commentary made on the fitness measures selected in these DFMs.

The DFMs derived by the GP approach will also be analyzed. The DFM analyzed for a given problem class is the best DFM found over 30 runs of the GP approach on the training problems from the class. Also, the DFM analyzed for a given problem domain is the best DFM found over 30 runs of the GP approach on the training problems from the domain. The GP chromosomes representing these DFMs will be reported and commentary made on the fitness measures selected in these DFMs.

5.5 Benchmark suite

The benchmark suite of problems is comprised of the following:

1. Problems used to test the effectiveness of DFMGP.
2. Problem classes used to test the reusability of the DFMs derived by the GA and GP approaches.
3. Real-world problems used to test the reusability of the DFMs derived by the GA and GP approaches.

The different problems that make up the benchmark suite are discussed in section 5.5.1. Subsequently, section 5.5.2 discusses the function and terminal sets used for the tackled problems. Finally, section 5.5.3 lists the fitness cases used for the problems.

5.5.1 Benchmark problems

The ensuing sections discuss the problems that make up the benchmark suite.

Problems used to test the effectiveness of DFMGP

The problems are listed in table 5.1. The problems in table 5.1 were selected from the benchmark suite specified in section 4.2.3 of chapter 4. These problems were selected on the basis of two considerations. The first consideration is to ensure representation from the different problem domains. Problems were selected from the symbolic regression, Boolean function synthesis and path-finding domains. The supervised classification domain is excluded, because the concept of a problem class (i.e. different instances of a given problem) is not applicable to the real-world datasets defined in chapter 4; i.e. the datasets do not contain problem instances, neither can one manipulate or transform the datasets to obtain problem instances without tampering with the real-world nature of the data. Conversely, different instances can be specified for the problems listed in table 5.1, as detailed in the ensuing discussion on the problem classes analyzed. The second consideration is that the GA and GP approaches used to derive the DFMs are computationally expensive approaches that runs several instances of DFMGP at the lower level; hence problems with a low computational cost (i.e. few fitness cases) were selected, so as not to exacerbate the overall cost of finding and executing the optimal DFMs.

TABLE 5.1: Benchmark problems tackled

Problem domain	Problem
Symbolic regression	Sextic polynomial (<i>abbrev. Sextic</i>)
Symbolic regression	Keijzer-6 (<i>abbrev. Keijzer</i>)
Boolean function synthesis	Even-7 Parity (<i>abbrev. Par-7</i>)
Boolean function synthesis	3-bit multiplier (<i>abbrev. Mult-3</i>)
Path-finding	Tartarus (<i>abbrev. Tart</i>)
Path-finding	Deceptive tartarus (<i>abbrev. Dec-tart</i>)

Problem classes used to test the reusability of the DFMs evolved by the GA and GP approaches

The problems are listed in table 5.2. In the symbolic regression domain, a problem class is defined in terms of a coefficient to a given regression function: here, given a regression function, $f(x)$, all problems of the form $a.f(x)$ (whereby $a \in \mathbb{Z}$) belong to the same problem class. In the Boolean and path-finding problems, a problem class is defined based on different instances (i.e. different levels of complexity and the use of different input parameters respectively) of a given problem.

Real-world problems used to test the reusability of the DFMs evolved by the GA and GP approaches

The problems are listed in table 5.3. The abalone [245] and Dow Chemical parser (abbrev. Dow) [232] datasets are selected as the real-world problems from the symbolic regression domain. The Dow dataset is a challenging real-world problem that was the subject of the symbolic regression EvoCompetitions event of the 2010 EvoStar conference [232]; this problem has been described in detail in section 4.2.3 of chapter 4. In turn, the abalone dataset is a well-known real-world problem to which GP has been applied in previous research [261, 262]: in the problem, a regression function is evolved to predict the age of abalone shellfish from 8 attributes; the fitness case set is comprised of 4177 instances [261, 262].

The flame detection circuit [263] (abbrev. sensor) and binary-coded-decimal-to-seven-segment decoder (abbrev. decoder) [264] problems are selected as the real-world problems from the Boolean function synthesis domain. The sensor problem involves the design of a sensor-input circuit to detect the presence or absence of a flame in a toxic waste incinerator [263]: in the current study, 6 sensors (each with a value of true="flame present" or false = "flame absent") supply input to the circuit, and the task is to produce 2 outputs: 1) an indication that a flame is present when at least 4 of the sensors detect a flame, and 2) an indication that the sensors should be repaired when at least 2 of the sensors differ from the other sensors [263]. The decoder problem involves converting a decimal number input (encoded as a 4-bit binary digit) into a seven segment light emitting diode (LED) display, whereby specific segments of the LED are turned on to display the seven

TABLE 5.2: Benchmark training/test problems tackled

Problem class	Training problems	Test problems
Sextic polynomial (<i>abbrev. Sextic</i>) (Symbolic regression domain)	10 random problems drawn from the set $a.(x^6 - 2x^4 - x^2)$. Note that care is taken to exclude the problems in the test set.	1) $2.(x^6 - 2x^4 + x^2)$ 2) $5.(x^6 - 2x^4 + x^2)$ 3) $10.(x^6 - 2x^4 + x^2)$ 4) $20.(x^6 - 2x^4 + x^2)$ 5) $50.(x^6 - 2x^4 + x^2)$ 6) $100.(x^6 - 2x^4 + x^2)$ 7) $200.(x^6 - 2x^4 + x^2)$ 8) $250.(x^6 - 2x^4 + x^2)$ 9) $500.(x^6 - 2x^4 + x^2)$ 10) $1000.(x^6 - 2x^4 - x^2)$
Keijzer-6 (<i>abbrev. Keijzer</i>) (Symbolic regression domain)	10 random problems drawn from the set $a.(\sum_{j=1}^x \frac{1}{j})$. Note that care is taken to exclude the problems in the test set.	1) $2.(\sum_{j=1}^x \frac{1}{j})$ 2) $5.(\sum_{j=1}^x \frac{1}{j})$ 3) $10.(\sum_{j=1}^x \frac{1}{j})$ 4) $20.(\sum_{j=1}^x \frac{1}{j})$ 5) $50.(\sum_{j=1}^x \frac{1}{j})$ 6) $100.(\sum_{j=1}^x \frac{1}{j})$ 7) $200.(\sum_{j=1}^x \frac{1}{j})$ 8) $250.(\sum_{j=1}^x \frac{1}{j})$ 9) $500.(\sum_{j=1}^x \frac{1}{j})$ 10) $1000.(\sum_{j=1}^x \frac{1}{j})$
Even-N parity (<i>abbrev Par-N</i>) (Boolean function synthesis domain)	The set Even-N parity, whereby $N \in \{3, 4, 5, 6, 7, 8\}$.	1) Even-9 parity 2) Even-10 parity 3) Even-11 parity 4) Even-12 parity 5) Even-13 parity
N-bit multiplier (<i>abbrev. Mult-N</i>) (Boolean function synthesis domain)	The set N-bit multiplier, whereby $N \in \{2, 3, 4\}$.	1) 5-bit multiplier 2) 6-bit multiplier 3) 7-bit multiplier 4) 8-bit multiplier 5) 9-bit multiplier
Tartarus (<i>abbrev Tart.</i>) (Path finding domain) KEY: gl = grid length; nb = number of blocks	10 random problems drawn from the set: Tartarus (gl = m; nb = n), whereby $m \in [4, 12]$ and $n \in [4, 12]$. Note that care is taken to exclude the problems in the test set.	1) Tartarus (gl = 8; nb = 6) 2) Tartarus (gl = 10; nb = 6) 3) Tartarus (gl = 6; nb = 7) 4) Tartarus (gl = 8; nb = 8) 5) Tartarus (gl = 6; nb = 8) 6) Tartarus (gl = 9; nb = 7) 7) Tartarus (gl = 10; nb = 7) 8) Tartarus (gl = 9; nb = 8) 9) Tartarus (gl = 10; nb = 8) 10) Tartarus (gl = 10; nb = 10)
Deceptive tartarus (<i>abbrev. Dec-tart.</i>) (Path finding domain) KEY: gl = grid length; nb = number of blocks	10 random problems drawn from the set: Deceptive tartarus (gl = m; nb = n), whereby $m \in [4, 12]$ and $n \in [4, 12]$. Note that care is taken to exclude the problems in the test set	1) Deceptive tartarus (gl = 8; nb = 6) 2) Deceptive tartarus (gl = 10; nb = 6) 3) Deceptive tartarus (gl = 6; nb = 7) 4) Deceptive tartarus (gl = 8; nb = 8) 5) Deceptive tartarus (gl = 6; nb = 8) 6) Deceptive tartarus (gl = 9; nb = 7) 7) Deceptive tartarus (gl = 10; nb = 7) 8) Deceptive tartarus (gl = 9; nb = 8) 9) Deceptive tartarus (gl = 10; nb = 8) 10) Deceptive tartarus (gl = 10; nb = 10)

segment representation of the number [264]. Decoder circuits are used widely in electronic devices that display numerical information, including basic calculators and digital clocks [264].

5.5.2 Function and terminal sets

This section lists the function and terminal sets used for the problems tackled in experiments 1 and 2. For the problems introduced in chapter 4, the information listed below is the same as provided in section 4.2.3 of

TABLE 5.3: Real-world problems tackled

Problem domain	Training problems	Test problems
Symbolic Regression	<i>Sextic</i> training problems	1) Abalone (<i>abbrev. Abalone</i>) 2) Dow Chemical parser (<i>abbrev. Dow</i>)
Symbolic Regression	<i>Keijzer</i> training problems	
Symbolic Regression	<i>Sextic + Keijzer</i> training problems	
Boolean Function Synthesis	<i>Par-N</i> training problems	1) Flame detection circuit (<i>abbrev. Sensor</i>) 2) Binary Coded Decimal to Seven Segment Decoder (<i>abbrev. Decoder</i>)
Boolean Function Synthesis	<i>Mu It-N</i> training problems	
Boolean Function Synthesis	<i>Par-N + Mult-N</i> training problems	

chapter 4. Additional information is incorporated below in relation to the new real-world problems included in experiments 1 and 2:

1. The function set used in the symbolic regression (*sextic*, *Keijzer-6*, *Dow* and *abalone*) problems is comprised of common mathematical $\{+, -, \times, \%, \sin, \cos, \log, \exp\}$ functions; however, the division ($\%$) and logarithm (\log) functions used are protected, such that the result of division is 1 whenever the denominator is 0, and the argument of the logarithm is always converted to its absolute value [1]. The terminal set used in the *sextic* and *Keijzer-6* problems is comprised of a single variable, $\{x\}$ [1]. In the *Dow* and *abalone* problems, the terminal sets used are comprised of alphabet characters that represent each of the distinctive attributes defined for the respective dataset.
2. The function set used in the even- N parity, *sensor* and *decoder* problems is comprised of the Boolean operators $\{AND, OR, NAND, NOR\}$ [1]. The even- N parity problems employ a terminal set $\{d_0, d_1, \dots, d_{N-1}\}$, comprised of a total of N variables, where the value of N is dependent on the particular even- N parity instance tackled [1]. In the *sensor* problem, input is received from 6 sensors, hence the terminal set used is $\{d_0, d_1, \dots, d_5\}$. In turn, in the *decoder* problem, a 4-bit input is received, hence the terminal set used is $\{d_0, d_1, \dots, d_3\}$.
3. The function set used in the N -bit multiplier problems is comprised of the Boolean operators $\{AND, ANDI, OR, XOR\}$ [234]. In turn, the terminal set $\{d_0, d_1, \dots, d_{2N-1}\}$, is comprised of a total of $2N$ variables, where the value of N is dependent on the particular N -bit multiplier instance tackled [234].
4. The function set used in the path-finding (*tartarus* and *deceptive tartarus*) problems, taken from [248], is comprised of a connective function (*PROGN2*), problem-specific logical functions (i.e. *IF-POSITION-IS-STATE* where $POSITION \in \{UM, UR, MR, LR, LM, LL, ML, UL\}$, and $STATE \in \{E, B, W\}$), simple logical functions (*NOT, EQ, IF*) and read/write functions (i.e. *READi, WRITEi* where $i \in \{1, 2, 3\}$). The terminal set is comprised of constants (*Zero, One, Two*) and actions (*TurnLeft, TurnRight, Move-Forward*).

5.5.3 Fitness cases

This section lists the fitness cases used for the tackled problems.

1. *Sextic* regression: 50 x -axis points, evenly spaced with an interval of 0.04 in the range $[-1, 1]$ [1].
2. *Keijzer* regression: 50 x -axis points, evenly spaced with an interval of 1.00 in the range $[1, 50]$ [232].
3. *Dow* regression: the set of 747 fitness cases defined in [232].
4. *Abalone* regression: the set of 4177 fitness cases defined in [245].
5. Even- N parity: 2^N Boolean numbers ranging from 0 to $2^N - 1$: the value of N is dependent on the particular even- N parity instance tackled [1].

6. N-bit multiplier: 2^{2N} Boolean numbers ranging from 0 to $2^{2N} - 1$: the value of N is dependent on the particular N -bit multiplier instance tackled [1].
7. Sensor: 2^6 Boolean numbers ranging from 0 to $2^6 - 1$ [263].
8. Decoder: 2^4 Boolean numbers ranging from 0 to $2^4 - 1$ [264].
9. Path finding (tartarus, deceptive tartarus): 40 randomly generated path-finding grid-worlds [248].

5.6 Technical specifications

The technical specifications of the experiment are identical to those listed in section 4.2.5 of chapter 4, and are restated here. The algorithms tested in the study were developed using Java SE (Oracle, version 8; [253]). The programs were developed on a computer with the following specifications: Intel(R) Celeron(R) N2840 @ 2.16GHz, 2.00 GB RAM, Windows 8 Enterprise OS. Simulations (trial and final) were run on the Center for High Performance Computing, South Africa¹. The analysis of the results was performed using Microsoft Excel 2010 [254] and Wolfram Mathematica 10.0 [255].

5.7 Summary

This chapter presented the methodology used for achieving the objectives outlined in chapter 1. The measurements used to analyse the achievement of the objectives were discussed. The benchmark problems that will be tackled to achieve the objectives of the research were also presented. In cases where there is need to show the significance of a result, a Z-test will be used to demonstrate statistical significance. Finally, the technical specifications for the development and testing of the developed approaches (the GA and GP approaches, as well as DFMGP) were provided.

¹See <https://www.chpc.ac.za/index.php/resources/lengau-cluster> for cluster specifications.

Chapter 6

The DFMGP Algorithm

6.1 Introduction

This chapter describes the DFMGP algorithm in detail. The chapter is organized as follows. Section 6.2 describes the DFMGP algorithm. Subsequently, sections 6.3, 6.4, 6.5, 6.6 and 6.7 describe the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used within DFMGP respectively. Next, section 6.8 discusses the parameters used for DFMGP. Finally, section 6.9 summarizes the chapter.

6.2 The DFMGP algorithm

DFMGP resembles Koza’s canonical GP [1], shown in listing 2.2 of chapter 2. The only difference between DFMGP and canonical GP is that in the former, different fitness measures are applied on the different GP generations; here, the fitness measure, f_i , applied on DFMGP generation i can be an existing fitness measure, or an arithmetic combination of existing fitness measures. Figure 6.1 illustrates a DFMGP algorithm spanning a total of N generations.

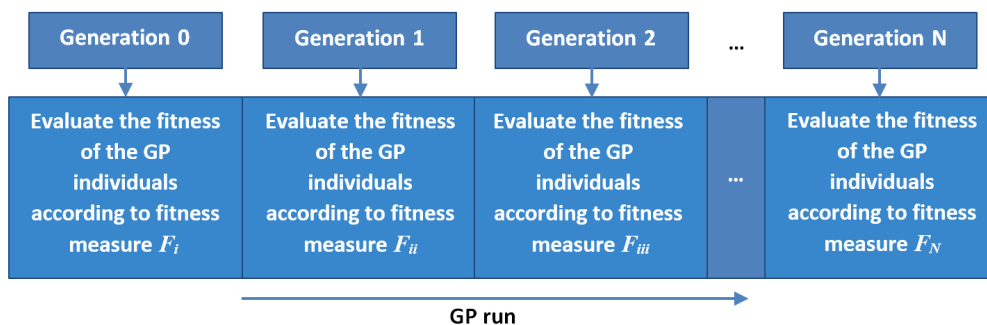


FIGURE 6.1: Schematic of DFMGP. The symbols $F_i, F_{ii}, F_{iii} \dots F_N$ represent different fitness measures.

Like canonical GP, DFMGP uses a generational control model [1]. DFMGP evolves a population of N_{DFMGP} candidate solutions for g_{DFMGP} generations; here, N_{DFMGP} and g_{DFMGP} are control parameters. As in the case with canonical GP, DFMGP terminates when a global optimum solution is found, or when the maximum number of generations have elapsed [1].

6.3 Representation scheme

The DFMGP candidate solutions are standard AST-GP parse trees, as proposed for canonical GP in [1]. The function and terminal sets used depend on the specific problem being tackled; these have been described in detail for each of the problems tackled in the study in section 5.5.2 of chapter 5.

6.4 Initial population creation

The DFMGP initial populations are generated using the ramped half-and-half method, with the parse trees ranging from depths of 2 to 6, as proposed for canonical GP in [1]. The ramped half-and-half method is argued to generate a more diverse initial population compared to the full and grow methods discussed in section 2.3.3 of chapter 2, because of the ability of the former to generate a wide variety of tree shapes and structures [1].

6.5 Fitness evaluation

The different fitness measures that will be used in DFMGP, as depicted in figure 6.1 of section 6.2, are comprised of fitness measures selected from a pre-defined subset; these are the same fitness measures that were analyzed in chapter 4:

1. Objective fitness (OF)
2. Behavioral programming (BP1)
3. Behavioral programming with archive supplied mutation (BP2)
4. Fitness sharing (FS)
5. Dynamic subset selection (DSS)
6. Host-parasite coevolution (HP)
7. Novelty search (NS1)
8. Novelty search with problem-specific path-finding behavior descriptor (NS2)

The GA and GP approaches used to evolve DFMs for DFMGP will select and combine fitness measures from the measures listed above. Chapter 4 established that the listed measures suit different problems. Furthermore some of the fitness measures have been argued to suit specific phases of search: for example, FS and NS are more suited to the preliminary generations of GP, rather than the later generations, due to poor exploitation capability [6, 15, 204]. The goal in providing a diverse subset of fitness measures for the DFMs is for the GA and GP approaches to be able to produce optimal DFMs for varied problems and problem classes.

6.6 Selection method

The DFMGP selection methods used are tournament and fitness-proportionate selection [1]. Different selection methods are used with the different fitness measures within DFMGP, as described in detail in section 6.8.

6.7 Genetic operators

The DFMGP genetic operators used are standard GP crossover, mutation and reproduction, as proposed for canonical GP in [1]; also, the archive-supplied mutation operator is applied with BP2 [5]. Different genetic operator application rates are used with the different fitness measures within DFMGP, as described in detail in section 6.8.

6.8 Parameter tuning

The SMAC-tuned fitness measure specific parameters employed in the experimental treatments in chapter 4 are used. These parameters are used for the fitness measure when applied in DFMGP. Hence, a different selection method and different genetic operator application rates are used for each fitness measure within DFMGP. This parameter tuning approach is adopted as an approximation of the optimal configuration for each of the fitness measures in DFMGP; parameter tuning is not applied directly to DFMGP, due to the combinatorial explosion of possible DFMGPs. The list of SMAC-tuned parameters for each of the fitness measures is the same as was provided in table 4.2 of chapter 4.

Apart from the SMAC-tuned parameters, the DFMGP population size and maximum number of DFMGP generations are tuned in the context of the GA/GP approach being used to evolve DFMs for DFMGP. These parameters are discussed in section 7.8 of chapter 7 for the GA approach and section 8.8 of chapter 8 for the GP approach.

6.9 Summary

This chapter presented the DFMGP algorithm used to apply different fitness measures on the different generations of GP. An overview of the algorithm was presented. Subsequently, detail was provided of the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used within DFMGP. The parametric configuration of DFMGP was also discussed. The ensuing chapters will detail the GA and GP approaches employed at the higher (or meta-) level to evolve DFMs for DFMGP.

Chapter 7

A GA Approach for Deriving DFMs for DFMGP

7.1 Introduction

This chapter describes the GA approach implemented to search the space of DFMs for DFMGP. The implemented GA operates at the higher (or meta-) level, and aims to discover the optimal sequence in which fitness measures should be applied in DFMGP for a given problem (or problem class); at the lower-level, DFMGP attempts to solve the problem.

This chapter is organized as follows. Section 7.2 provides an overview of the GA approach. Subsequently, sections 7.3, 7.4, 7.5, 7.6 and 7.7 describe the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used respectively. Next, section 7.8 discusses the parameters used by the GA approach. Finally, section 7.9 summarizes the chapter.

7.2 Genetic algorithm for dynamic fitness measure GP

The GA approach evolves a population of candidate fitness measure sequences for DFMGP. The GA takes one or more GP problems as input. Subsequently, the GA aims to discover an optimal fitness measure sequence for the input problem(s). Figure 7.1 illustrates the interaction between the GA, DFMGP and the solution program space.

The GA evolves a population of N_{GA} candidate fitness measure sequences for g_{GA} generations; N_{GA} and g_{GA} are control parameters. The fitness of a candidate fitness measure sequence is the mean best solution quality achieved by DFMGP applying the sequence. Here, DFMGP applying the sequence is run r times in order to obtain an accurate estimate for the best solution quality, given the stochastic nature of DFMGP; section 5.2.1 of chapter 5 established that when evolving DFMs for a given problem, r is set at 5; otherwise, when evolving DFMs for a given problem class, DFMGP applying the corresponding DFM is run once for each of the problems in the training set, such that the value of r is equal to the total number of problems in the training set. The GA approach is run 30 times on the given problem(s), such that the DFM derived by the approach is the best DFM found over the 30 runs. Overall, the GA exerts considerable computational effort. However, the GA's automation is better than a cumbersome manual search of the space of fitness measure sequences; the latter approach is infeasible, given the combinatorial explosion of possible sequences. Also, the GA's computational effort is considered worthwhile if the derived fitness measure sequences are shown to be reusable on unseen problems, reducing the total time necessary for the derivations.

7.3 Representation

Each GA chromosome represents a sequence of fitness measures. The previous chapter established that the fitness measures used in DFMGP will be selected from a pre-defined subset: 1) OF, 2) BP1, 3) BP2, 4) FS, 5)

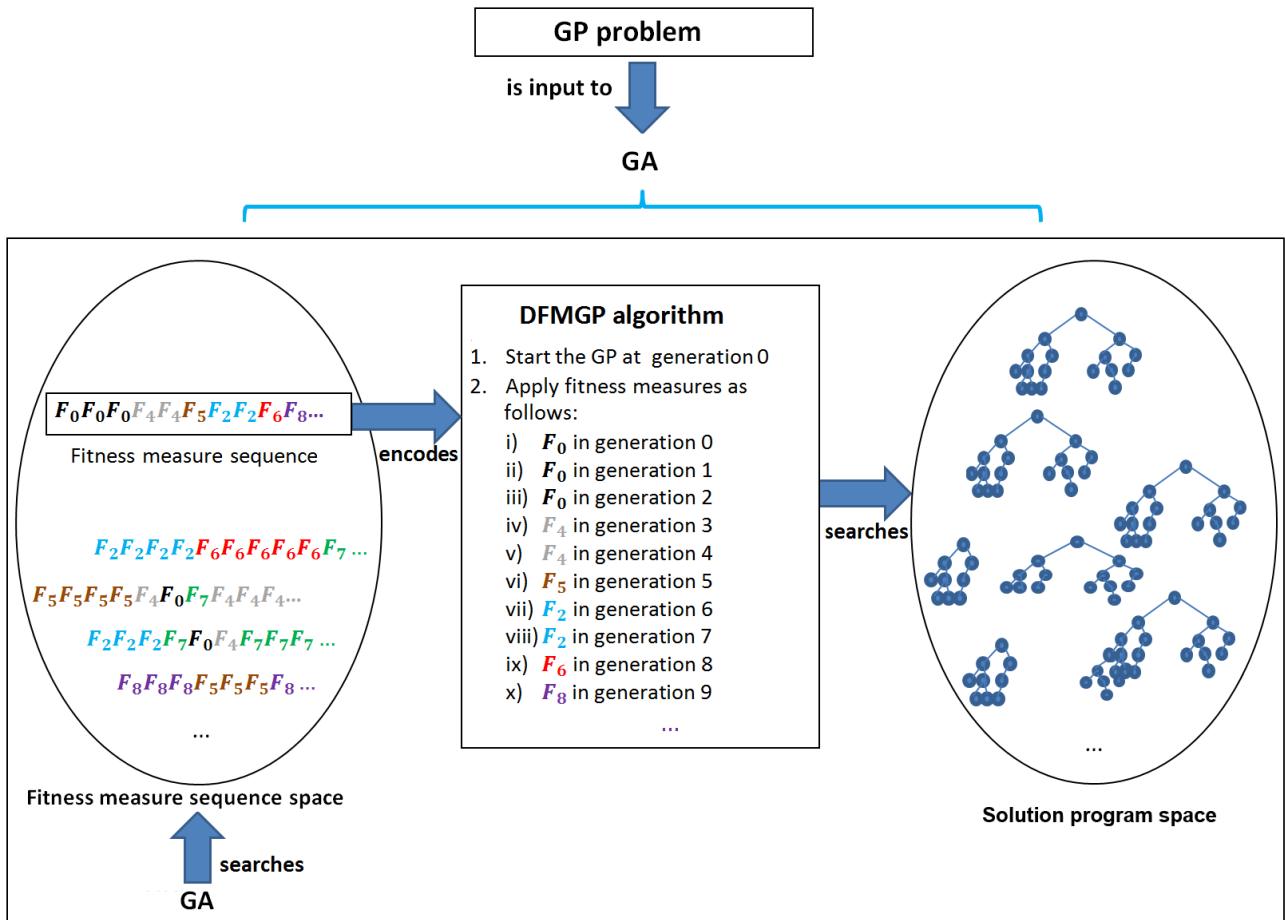


FIGURE 7.1: Overall GA approach

DSS, 6) HP, 7) NS1, and 8) NS2. The following characters are used to encode the fitness measures:

- i. *A* encodes Objective fitness (OF).
- ii. *B* encodes Behavioral programming (BP1).
- iii. *C* encodes Behavioral programming with archive supplied mutation (BP2).
- iv. *D* encodes Fitness sharing (FS).
- v. *E* encodes Dynamic subset selection (DSS).
- vi. *F* encodes Host-parasite coevolution (HP).
- vii. *G* encodes Novelty search (NS1).

An additional character, *H*, is appended to this set when tackling path-finding problems. *H* encodes novelty search with a problem-specific path-finding behavior descriptor (NS2). In chapter 4 of the study, problem-specific path-finding descriptors were shown to enhance the potential of NS in the deceptive path-finding domain.

The GA chromosomes used are of length g_{GP} , the number of DFMGP generations, such that the character at position p within a chromosome corresponds to the fitness measure applied on the generation with the same index within the DFMGP. An example GA chromosome is shown below:

Example:

```
GGGGGGGGGGGGGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
AAAAAAAAAAAAAAAAAAAAADDDDDDDDDDDDDDDDDDDDDDDDC
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
CCCCCCCCCCCCCCCCC
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

The above chromosome represents a sequence spanning 101 DFMGP generations: 13 generations of novelty search (NS1 - encoded as *G*), followed by 50 generations of objective fitness (OF - encoded as *A*), followed by 20 generations of fitness sharing (FS - encoded as *D*), and lastly, 18 generations of behavioral programming with archive supplied mutation (BP2 - encoded as *C*).

7.4 Initial population creation

The initial population is generated randomly, but in such a way that the fitness measures span contiguous DFMGP generations: e.g. *AAAAAAACCCCCDDDD*, as opposed to *ADCBGHEFAFEFACDC*. This is done in order to ensure that the DFMGPs are not noisy in the sense of continually switching between the fitness measures. Rather, each fitness measure has an opportunity to progress the search in its own respect for a number of generations. For this reason, each initial population chromosome is created by randomly selecting k fitness measures, where $k = 1, 2, 3, 4$; here, each fitness measure is applied for m consecutive DFMGP generations with $m \approx g_{GP}/k$. The initial population contains an equal number of chromosomes for each value of k . The maximum value for k is set at 4, because higher values of k are seen not to yield improvements in the fitness of the initial population.

For example, if $g_{GP} = 101$ and $k = 2$, an example of a resulting chromosome is:

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

In the above example, OF (encoded as *A*) is selected as the first fitness measure spanning the first 51 DFMGP generations. BP1 (encoded as *B*) is selected as the second fitness measure spanning the remaining 50 DFMGP generations.

7.5 Fitness evaluation

The fitness of a candidate fitness measure sequence is the mean quality of the best solution program produced by DFMGP applying the sequence. Here, the mean quality is obtained by conducting a total of r runs of the DFMGP, and averaging the best quality achieved over the r runs. The solution quality is measured in terms of the raw objective fitness (OF) score (see equation 4.1 in chapter 4). Equation 7.1 below summarizes the fitness evaluation of a candidate fitness measure sequence.

$$F(i_{DFM}) = \frac{\sum_{j=1}^r OF(i_{DFM}(best_j))}{r} \quad (7.1)$$

whereby:

1. $F(i_{DFM})$ represents the fitness of i_{DFM} .
2. i_{DFM} is a candidate fitness measure sequence.
3. $OF(i_{DFM}(best_j))$ represents the raw OF score of the best solution program found on the j^{th} run of DFMGP applying i_{DFM} . $OF(i_{DFM}(best_j))$ is averaged over the r runs of the DFMGP applying i_{DFM} .

7.6 Selection method

Tournament selection is used, whereby t elements of the population are selected at random, and only the element with the higher fitness is retained [1]. The tournament size, t , is configured by empirical tuning, whereby multiple runs are conducted, and the tournament size that gives the best result is selected.

7.7 Genetic operators

This section describes the genetic operators used by the GA. These operators include those traditionally used in GAs, namely, one-point crossover and mutation [3]. In addition, newly defined disruptive genetic operators are applied, which were seen to delay convergence during empirical testing of the GA on different problems, namely, the operators reverse, chop and reorder the fitness measure sequences. The new operators delay convergence with the aim of mitigating premature convergence of the GA. The genetic operators used are described below:

1. *Standard one-point crossover* - Two fitness measure sequences are crossed over at a randomly selected position.

Before crossover:

Sequence 1 = *AAAAAAAAAAAAAAAA...AAAAAAAAAAAAAAAAAAAA*

Sequence 2 = *BBBBBBBBBBBBBB...BBBBBBBBBBBBBBBBBBBB*

After crossover:

Offspring 1 = *AAAABBBBBBBB...BBBBBBBBBBBBBBBBBBBB*

Offspring 2 = *BBBBBAAAAAAAAA...AAAAAAAAAAAAAAAAAAAA*

In the above example the two fitness measure sequences are crossed over at position 6.

2. *Standard one-point mutation* - The fitness measure at a randomly selected position within the given sequence is substituted with another fitness measure randomly drawn from the alphabet of fitness measures.

Example:

Before mutation: *AAAAAAAAAAAAAAAAAAAAAAAAAAAAA...*

After mutation: *ABAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...*

In the above example mutation is performed at position 2 in the sequence.

3. *Replace* - All occurrences of a randomly selected fitness measure within the given fitness measure sequence are substituted with another fitness measure randomly selected from the characters representing the fitness measures.

Example:

Before replace: *AAAAAACCCCCAAAABBAACCCCAA...*

After replace: *AAAAAADDDDDAAAABBAADDDDDAA...*

In the above example all occurrences of the fitness measure C are replaced with the fitness measure D.

4. *Shift* - The given fitness measure sequence is shifted right by a random number. The shift operation wraps around, such that the last fitness measure in the sequence is shifted to the beginning of the sequence.

Example:

Before shift: *DDDDDDDDDDAAAAAAAAAAAAA...BBBBBB*

After shift: *BBBDDDDDDDDDDAAAAAAAAAAAAA...BB*

In the above example the fitness measure sequence is shifted right by 4 steps.

5. *Reverse* - The given fitness measure sequence is reversed.

Example:

Before reverse: *AAAAAAAAAAAAA...BBBBBBBBBBBBBBBB*

After reverse: *BBBBBBBBBBBBBBBB...AAAAAAAAAAAAA*

6. *Random sequence* - A random subsequence of fitness measures replaces a subsequence of the same length at a random position within the given fitness measure sequence.

Example: random subsequence = *HHHDDAAAGGCH*

Before replacement: *AAAAAAAAAAAAAAAAAAAAAAAAAAAAA...*

After replacement: *AAHHHDDAAAGGCHAAAAAAAAAAAAA...*

In the above example the random subsequence is substituted into position 3 of the given fitness measure sequence.

7.8 Parameter tuning

This section discusses the parameters decided during development. There are two sets of parameters, namely, the GA parameters and the parameters used by the lower-level DFMGP. All the parameters discussed below were tuned empirically, whereby multiple runs are conducted, and the parameters that produce the best results with respect to both solution quality and time considerations, are selected. The parameter tuning was done to explore a number of possible configurations.

The following GA parameters were tuned: the initial population creation routine, the genetic operators, the selection method, the genetic operator probabilities, the population size and the number of generations. The initial population routine described in section 7.4 was shown to produce the best results over the iterative refinements of the GA approach. Also, the genetic operators described in section 7.7 were shown to produce the best results. The following genetic operator probabilities were decided on: standard GA crossover is employed with 50% probability; the remaining genetic operators described in section 7.7 are each applied with a probability of 10%. The selection method that produced the best results was tournament selection with a tournament size of 7. The following were decided on for the GA population size and termination criteria: the GA evolves a population of 50 DFMs and terminates when after a total of 51 GA generations have elapsed.

The following DFMGP parameters were tuned: the population size and the number of DFMGP generations. The following configuration for the lower-level GPs produced the best result on the different problems tackled: for each candidate DFM within the GA, the lower-level DFMGPs applying the DFM evolve a population of 500 candidate solution programs for 101 generations (i.e. an initial random generation 0 plus 100 additional generations); 101 generations are selected in order to allow DFMGP to improve on solution quality over an extended period of time (the convention used in the literature [1, 252] is 51 GP generations).

7.9 Summary

This chapter described a GA approach to derive a sequence of fitness measures for DFMGP. The GA operates at the high (or meta-) level, and searches the space of fitness measure sequences with the aim of discovering an optimal sequence for DFMGP; at the lower-level, DFMGP attempts to solve the given problem. An overview of the GA approach was presented. Subsequently, detail was provided of the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used by the approach. Finally, the parameters chosen for the GA approach were presented.

Chapter 8

A GP Approach for Deriving DFMs for DFMGP

8.1 Introduction

This chapter describes the GP approach implemented to search the space of DFMs for DFMGP. The implemented GP operates at the higher (or meta-) level, and aims to discover optimal fitness measure combinations for DFMGP run on a given problem instance (or problem class); at the lower-level, DFMGP attempts to solve the problem.

This chapter is organized as follows. Section 8.2 provides an overview of the GP approach. Subsequently, sections 8.3, 8.4, 8.5, 8.6 and 8.7 describe the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used respectively. Next, section 8.8 discusses the parameters used by the GP approach. Finally, section 8.9 summarizes the chapter.

8.2 Genetic programming for dynamic fitness measure GP

The GP approach evolves a population of candidate fitness measure combinations for DFMGP. The GP takes one or more problems as input, and aims to discover an optimal fitness measure combination for the input problem(s). Figure 8.1 illustrates the relationship between the GP, DFMGP and the solution program space.

A candidate DFM in the GP is interpreted as follows: on each DFMGP generation, the parse tree encoding the DFM is traversed to obtain the fitness measure combination to use on the DFMGP generation; here, the obtained fitness measure combination is used to evaluate the fitness of all candidate solution programs in the current DFMGP generation; this process is repeated for all the DFMGP generations.

The GP evolves a population of N_{GP} DFMs for g_{GP} generations; N_{GP} and g_{GP} are control parameters. The fitness of a candidate DFM is the mean best solution quality achieved by DFMGP applying the DFM. Here, DFMGP applying the DFM is run r times in order to obtain an accurate estimate for the best solution quality, given the stochastic nature of DFMGP; section 5.2.2 of chapter 5 established that when evolving DFMs for a given problem, r is set at 5; otherwise, when evolving DFMs for a given problem class, DFMGP applying the corresponding DFM is run once for each of the problems in the training set, such that the value of r is equal to the total number of problems in the training set. The GP approach is run 30 times on the given problem(s), such that the DFM derived by the approach is the best DFM found over the 30 runs. Overall, the GP exerts considerable computational effort. However, the GP's automation is better than a cumbersome manual search of the space of fitness measure combinations; the latter approach is infeasible, given the combinatorial explosion of possible combinations. Also, as in the case with the GA approach, the GP's computational effort is considered worthwhile if the derived fitness measure combinations are shown to be reusable on unseen problems, reducing the total time necessary for the derivations.

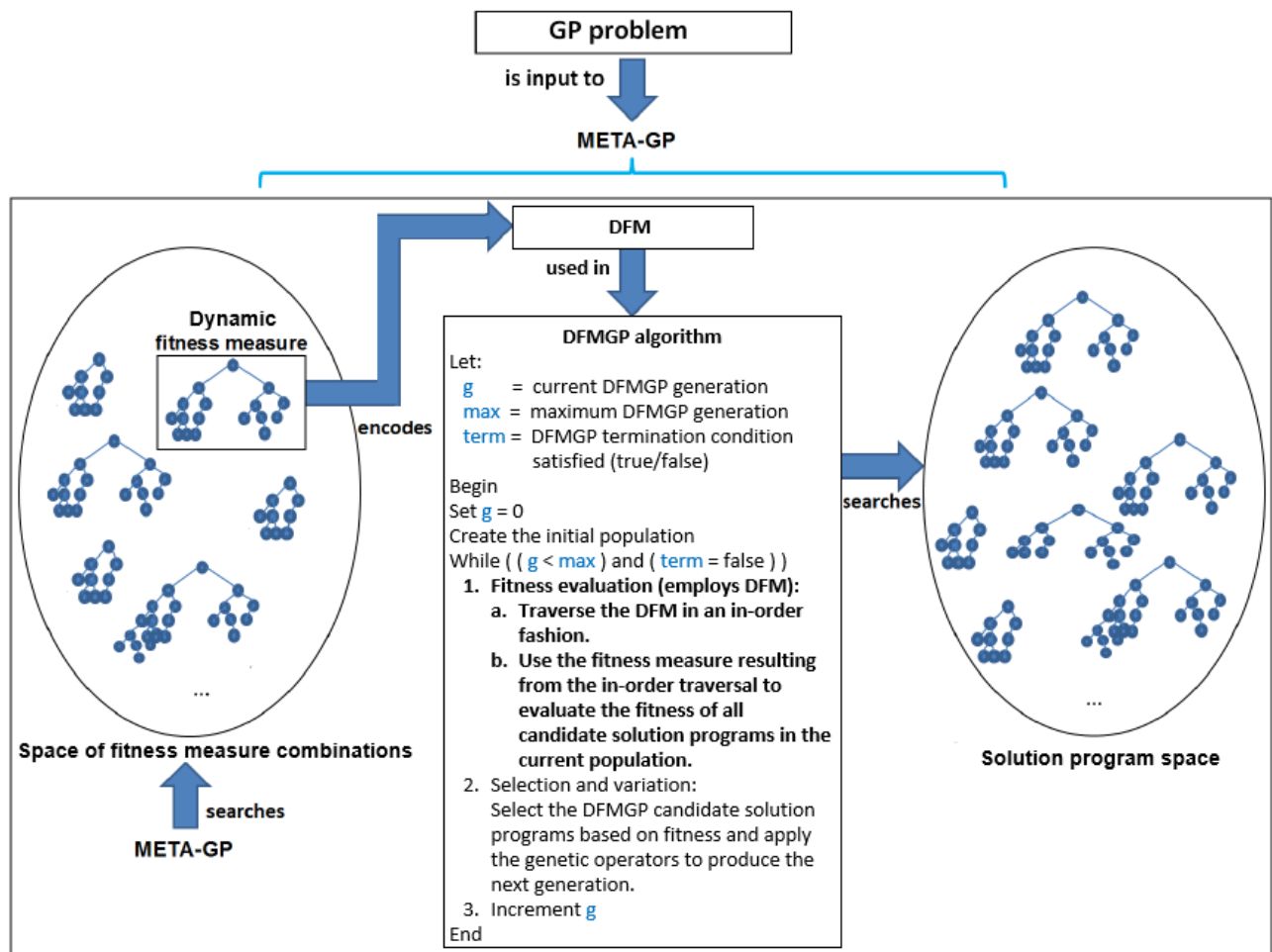


FIGURE 8.1: Overall approach

8.3 Representation

The GP uses a strongly-typed parse-tree representation comprised of the following types: 1) integer (abbrev. \mathcal{I}), and 2) fitness measure (abbrev. \mathcal{F}). The following constraints are applied:

1. A candidate DFM in the GP should always return a variable of type \mathcal{F} : thus the root of a parse tree encoding a DFM should be a node of type \mathcal{F} . Ultimately, \mathcal{F} is the type of interest. Nodes of type \mathcal{I} are strictly used to configure an integer parameter that influences the returned fitness measure (see section 8.3.2).
2. As in all strongly-typed GP systems, the genetic operators used in the GP are constrained such that nodes can only replace identical types during crossover and mutation events. The initial population creation routine described in section 8.4 is also modified such that random but valid arguments are supplied to all functions, such that only legal trees are created.

Further information on the listed types is provided in a discussion of the terminal and function sets in sections 8.3.1 and 8.3.2 respectively.

8.3.1 Terminal Set

The terminal set is comprised of the following:

1. The set $\{A, B, C, D, E, F, G\}$ of type \mathcal{F} primitives encoding the fitness measures, whereby:
 - i. A encodes Objective fitness (OF).

- ii. B encodes Behavioral programming (BP1).
- iii. C encodes Behavioral programming with archive supplied mutation (BP2).
- iv. D encodes Fitness sharing (FS).
- v. E encodes Dynamic subset selection (DSS).
- vi. F encodes Host-parasite coevolution (HP).
- vii. G encodes Novelty search (NS1).

An additional character, H , is appended to this set when tackling path-finding problems. H encodes novelty search with a problem-specific path-finding behavior descriptor (NS2). In chapter 4 of the study, problem-specific path-finding descriptors were shown to enhance the potential of NS in the deceptive path-finding domain.

2. \mathcal{R} , a symbol representing an ephemeral random constant (ERC): an ERC is a random integer generated during the construction of an initial population individual, which is assigned to a terminal node and remains fixed for the duration of the GP run [1]. Each time the symbol \mathcal{R} is chosen in the construction of an initial tree, a different random integer is generated from the interval $[0, g_{DFMGP}]$, where g_{DFMGP} is the maximum number of DFMGP generations.

Table 8.1 summarizes the terminal set used.

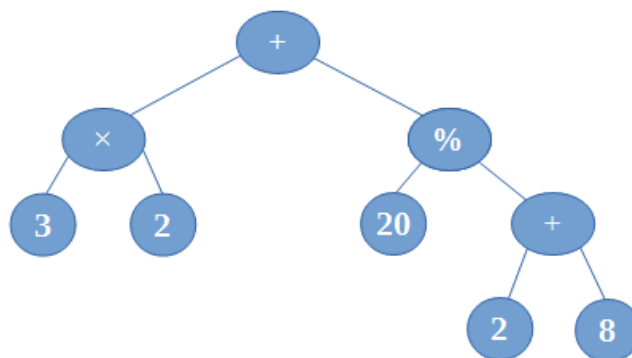
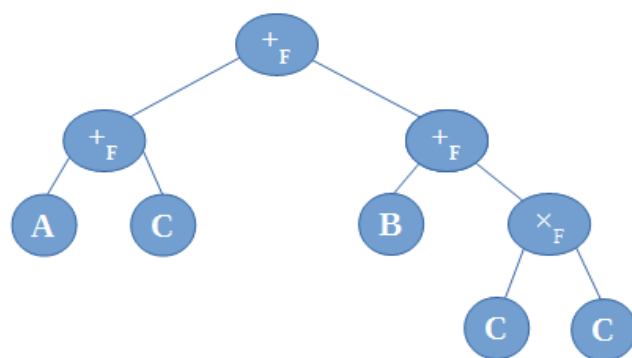
TABLE 8.1: Terminal set

Terminal	Type
A	\mathcal{F}
B	\mathcal{F}
C	\mathcal{F}
D	\mathcal{F}
E	\mathcal{F}
F	\mathcal{F}
G	\mathcal{F}
H	\mathcal{F}
\mathcal{R}	\mathcal{I}

8.3.2 Function Set

The function set is comprised of the following:

1. The set of arithmetic operators $\{+, -, \times, \%\}$: these are the addition, subtraction, multiplication and protected division operators respectively. The operators are used to perform operations on the terminal set ERCs. Each arithmetic operator takes two variables of type \mathcal{I} as input, and outputs a variable of type \mathcal{I} . Figure 8.2 shows an example subtree rooted at this type of operator. The subtree in the figure evaluates to $(3 \times 2) + (20\%(2 + 8)) = 8$.
2. The set of arithmetic operators $\{+_F, -_F, \times_F, \%_F\}$: here, the subscript, F , indicates operands of type \mathcal{F} , distinguishing these operators from the arithmetic operators described above. The operators behave exactly like arithmetic operators, except that they are used to add, subtract, multiply and divide the arithmetic output returned by the fitness measures. Fig 8.3 shows an example subtree rooted at this type of operator. The subtree in the figure evaluates to $(A + C) + (B + (C \times C))$: here, the fitness score is calculated individually according to each listed fitness measure (i.e. A , B and C). Subsequently, the arithmetic outputs from A , B and C are combined according to the listed formula, to produce a final arithmetic output of type \mathcal{F} .

FIGURE 8.2: Example subtree rooted at arithmetic operator of type I FIGURE 8.3: Example subtree rooted at arithmetic operator of type F

- Two conditional operators, $IFGLT$ and $IFGGT$. The operators are used to select fitness measures conditionally, based on the current DFMGP generation. Both $IFGLT$ and $IFGGT$ take three input parameters: one of type I (I_1), and two of type F (F_2 and F_3). $IFGLT$ and $IFGGT$ both return an output of type F . $IFGLT$ is interpreted as follows: if the current DFMGP generation, g , is less than I_1 , apply fitness measure F_2 , otherwise apply fitness measure F_3 . $IFGGT$ is interpreted similarly, except that the condition evaluated is g greater than I_1 . To ensure that I_1 specifies a valid DFMGP generation, the parameter is adjusted to yield an integer in the interval $[0, g_{DFMGP}]$, whereby g_{DFMGP} is the maximum DFMGP generation: this is achieved by applying the transformation $I_1 = (\lceil I_1 \rceil) \bmod (g_{DFMGP})$ to the given value of I_1 , where $\lceil \cdot \rceil$ and \bmod are the ceiling and modulus operators respectively.

Figure 8.4 shows two example subtrees rooted at $IFGLT$ and $IFGGT$. The first tree is interpreted as follows: if the current DFMGP generation, g , is less than 5, apply fitness measure A , otherwise apply fitness measure B . The second tree is interpreted as follows: if the current DFMGP generation, g , is greater than the value of $((\lceil 3 \times 34 \rceil) \bmod (g_{DFMGP}))$, apply fitness measure E , otherwise the fitness measure is $(A + (C \times C))$, an arithmetic combination of the individual outputs from A and C .

Table 8.2 summarizes the function set used.

8.4 Initial population creation

As in the case with the DFMGPs running at the lower level, the GP initial population is generated using the ramped half-and-half method, with the parse trees ranging from depths of 2 to 6 [1].

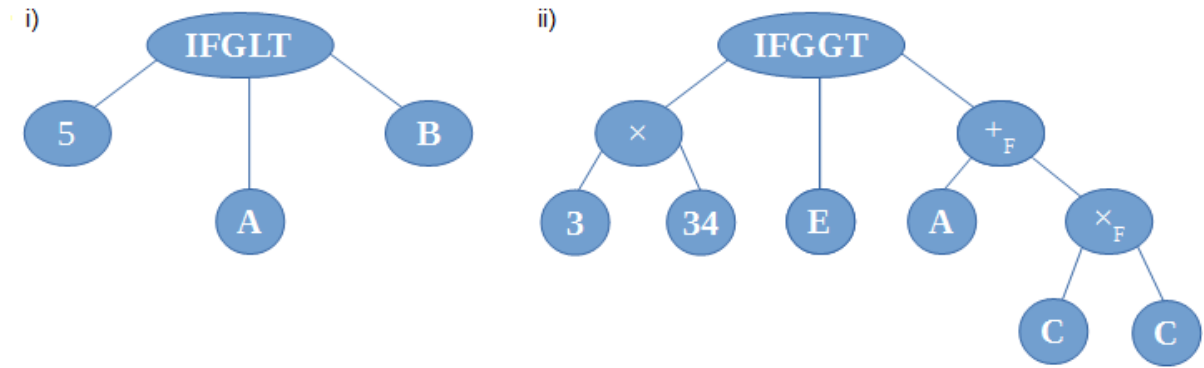
FIGURE 8.4: Example subtrees rooted at *IFGGT* and *IFGLT*

TABLE 8.2: Function set

Function	Input Types	Output Type
+	2 inputs: type \mathcal{I}	\mathcal{I}
−	2 inputs: type \mathcal{I}	\mathcal{I}
×	2 inputs: type \mathcal{I}	\mathcal{I}
%	2 inputs: type \mathcal{I}	\mathcal{I}
$+_F$	2 inputs: type \mathcal{F}	\mathcal{F}
$-_F$	2 inputs: type \mathcal{F}	\mathcal{F}
\times_F	2 inputs: type \mathcal{F}	\mathcal{F}
$\%_F$	2 inputs: type \mathcal{F}	\mathcal{F}
<i>IFGLT</i>	3 inputs: 1 type \mathcal{I} , 2 type \mathcal{F}	\mathcal{F}
<i>IFGGT</i>	3 inputs: 1 type \mathcal{I} , 2 type \mathcal{F}	\mathcal{F}

8.5 Fitness evaluation

As in the case with the GA approach, the fitness of a candidate DFM in the GP is calculated as the mean quality of the best solution program produced by the DFMGP applying the DFM: the mean quality is obtained by conducting a total of r runs of the DFMGP, and averaging the best quality achieved over the r runs. Equation 8.1 below summarizes the fitness evaluation of a candidate DFM.

$$F(i_{DFM}) = \frac{\sum_{j=1}^r OF(i_{DFM}(best_j))}{r} \quad (8.1)$$

whereby:

1. $F(i_{DFM})$ represents the fitness of i_{DFM} .
2. i_{DFM} is a candidate DFM.
3. $OF(i_{DFM}(best_j))$ represents the raw OF score of the best solution program found on the j^{th} run of DFMGP applying i_{DFM} . $OF(i_{DFM}(best_j))$ is averaged over the r runs of the DFMGP applying i_{DFM} .

8.6 Selection method

Tournament selection is used. The tournament size is configured by empirical tuning, whereby multiple runs are conducted, and the tournament size that gives the best result is selected.

8.7 Genetic operators

Standard GP crossover and mutation are used [1]. A creation operator, which was observed to mitigate against premature convergence during empirical tuning, is also used: the creation operator replaces an existing parse

tree with a random tree, whereby the latter is created using the initial population creation routine described in section 8.4. The probabilities for the mentioned genetic operators are configured by empirical tuning.

8.8 Parameter tuning

This section discusses the parameters decided during development. As in the case with the GA approach, There are two sets of parameters, namely, the GP parameters and the parameters used by the lower-level DFMGP. All the parameters discussed below were tuned empirically, whereby multiple runs are conducted, and the parameters that produce the best results with respect to both solution quality and time considerations, are selected. The parameter tuning was done to explore a number of possible configurations.

The following GP parameters were tuned: the maximum tree depth for the GP parse trees, the genetic operators, the selection method, the genetic operator probabilities, the population size and the number of generations. The maximum tree depth decided on for the GP parse trees is 17; this value for the maximum tree depth was shown to produce the best results over the iterative refinements of the GP approach. Also, the genetic operators described in section 8.7 were shown to produce the best results. The following genetic operator application rates were decided on: standard crossover, mutation and creation are employed with probabilities of 80%, 10% and 10% respectively. The selection method that produced the best results was tournament selection with a tournament size of 7. The following were decided on for the GP population size and termination criteria: the GP evolves a population of 50 DFMs and terminates when after a total of 51 GP generations have elapsed. This configuration of the population size and termination criteria are the same as were decided for the GA approach, so that the total number of DFM fitness evaluations conducted by the GA and GP approaches are comparable.

The lower-level DFMGP parameters used by the GP approach are the same as were used by the GA approach. For each candidate DFM within the GP, the lower-level DFMGPs applying the DFM evolve a population of 500 candidate solution programs for 101 generations (i.e. an initial random generation 0 plus 100 additional generations).

8.9 Summary

This chapter described a GP approach to derive a sequence of fitness measures for DFMGP. The GP operates at the high (or meta-) level, and searches the space of fitness measure combinations with the aim of discovering an optimal combination for DFMGP; at the lower-level, DFMGP attempts to solve the given problem. An overview of the GP approach was presented. Subsequently, detail was provided of the representation scheme, initial population creation, fitness evaluation, selection method and genetic operators used by the approach. Finally, the parameters chosen for the GP approach were presented.

Chapter 9

Results and Discussion

9.1 Introduction

This chapter details the results of the two approaches developed to achieve the objectives outlined in chapter 5. Section 9.2 presents the results of the GA approach for deriving DFMs for DFMGP, and DFMGP applying these DFMs ($DFMGP_{GA}$). Section 9.3 presents the results of the GP approach for deriving DFMs for DFMGP, and DFMGP applying these DFMs ($DFMGP_{GP}$). Section 9.4 presents a comparison of the two approaches results. Finally, section 9.5 draws conclusions on the study based on the results presented.

9.2 Results of the GA approach and $DFMGP_{GA}$

This section presents the results obtained by the approach described in chapter 7, namely a GA approach for deriving DFMs for DFMGP. Section 9.2.1 discusses the results obtained from testing the effectiveness of the GA approach and $DFMGP_{GA}$. Subsequently, section 9.2.2 presents the results obtained from testing the reusability of the derived DFMs within problem classes. Next, section 9.2.3 presents the results obtained from testing the reusability of the derived DFMs on real world problems. Finally, section 9.2.4 presents an analysis of the derived DFMs to identify the fitness measures that suit the different phases of search for different problems.

9.2.1 Testing the effectiveness of $DFMGP_{GA}$

This section presents the results obtained from comparing the performance of $DFMGP_{GA}$ with that of standard GP applying each of the fitness measures individually on the benchmark problems listed in table 5.1 of chapter 5. The performance of $DFMGP_{GA}$ is also compared to that of DFMGP applying randomly generated DFMs (obtained by the GA initial population creation procedure described in section 7.4 of chapter 7); the latter approach is abbreviated as $DFMGP_{RD1}$.

Table 9.1 shows the results obtained by running $DFMGP_{GA}$, $DFMGP_{RD1}$ and standard GP on the tackled problems. The table shows the best solution quality, b , achieved over 30 runs of GP/DFMGP; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. As in the experiments conducted in chapter 4, in the Boolean and path-finding domains, the solution quality is measured as an increasing function spanning the interval $[0, 1]$: in the former case, this is the proportion of the defined fitness cases solved; in the latter case, this is the proportion of the path-finding task solved. In turn, in the symbolic regression domain, the solution quality is measured as the sum of the absolute error over the fitness cases defined for the given problem: the quality scores span the interval $[0, \infty]$, whereby lower scores indicate better quality. As in chapter 4, for the sake of uniformity with the Boolean and path-finding domains, the transformation function $1 - (x/(x + 1))$ is applied to each symbolic regression quality score. Therefore, all quality scores listed in table 9.1 span the interval $[0, 1]$, with higher scores indicating better quality.

In table 9.1, the results for NS2 are greyed out for problems other than the path-finding problems, because the behavior descriptors used in NS2 are specific to the path-finding problems (see chapter 4). Table 9.1 highlights the best performing GP approach on each problem. $DFMGP_{GA}$ is seen to achieve near-optimal

TABLE 9.1: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP							
			OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Sextic	$b = 1.00$ $\mu = 0.95$ $\sigma = 0.03$	$b = 0.96$ $\mu = 0.85$ $\sigma = 0.05$	$b = 1.00$ $\mu = 0.86$ $\sigma = 0.03$	$b = 1.00$ $\mu = 0.84$ $\sigma = 0.04$	$b = 1.00$ $\mu = 0.91$ $\sigma = 0.04$	$b = 0.91$ $\mu = 0.80$ $\sigma = 0.04$	$b = 0.93$ $\mu = 0.85$ $\sigma = 0.06$	$b = 0.90$ $\mu = 0.86$ $\sigma = 0.06$	$b = 0.91$ $\mu = 0.80$ $\sigma = 0.06$	
Keijzer	$b = 0.87$ $\mu = 0.77$ $\sigma = 0.08$	$b = 0.71$ $\mu = 0.59$ $\sigma = 0.11$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.08$	$b = 0.71$ $\mu = 0.59$ $\sigma = 0.09$	$b = 0.76$ $\mu = 0.66$ $\sigma = 0.09$	$b = 0.76$ $\mu = 0.67$ $\sigma = 0.09$	$b = 0.75$ $\mu = 0.67$ $\sigma = 0.09$	$b = 0.73$ $\mu = 0.65$ $\sigma = 0.07$	$b = 0.61$ $\mu = 0.49$ $\sigma = 0.11$	
Par-7	$b = 0.84$ $\mu = 0.73$ $\sigma = 0.05$	$b = 0.71$ $\mu = 0.59$ $\sigma = 0.07$	$b = 0.73$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.62$ $\mu = 0.58$ $\sigma = 0.02$	$b = 0.75$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.59$ $\mu = 0.56$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.66$ $\sigma = 0.03$	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.58$ $\sigma = 0.01$	
Mult-3	$b = 0.99$ $\mu = 0.97$ $\sigma = 0.01$	$b = 0.95$ $\mu = 0.89$ $\sigma = 0.11$	$b = 0.93$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.93$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.92$ $\sigma = 0.01$	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.01$	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.01$	$b = 0.92$ $\mu = 0.88$ $\sigma = 0.02$	$b = 0.74$ $\mu = 0.71$ $\sigma = 0.02$	
Tart	$b = 0.74$ $\mu = 0.62$ $\sigma = 0.04$	$b = 0.60$ $\mu = 0.41$ $\sigma = 0.04$	$b = 0.67$ $\mu = 0.51$ $\sigma = 0.06$	$b = 0.64$ $\mu = 0.52$ $\sigma = 0.09$	$b = 0.66$ $\mu = 0.51$ $\sigma = 0.06$	$b = 0.30$ $\mu = 0.14$ $\sigma = 0.02$	$b = 0.74$ $\mu = 0.57$ $\sigma = 0.06$	$b = 0.65$ $\mu = 0.48$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.14$ $\sigma = 0.02$	$b = 0.69$ $\mu = 0.59$ $\sigma = 0.04$
Dec-tart	$b = 0.78$ $\mu = 0.69$ $\sigma = 0.02$	$b = 0.50$ $\mu = 0.43$ $\sigma = 0.03$	$b = 0.59$ $\mu = 0.49$ $\sigma = 0.04$	$b = 0.76$ $\mu = 0.66$ $\sigma = 0.03$	$b = 0.78$ $\mu = 0.67$ $\sigma = 0.03$	$b = 0.52$ $\mu = 0.46$ $\sigma = 0.02$	$b = 0.65$ $\mu = 0.50$ $\sigma = 0.03$	$b = 0.61$ $\mu = 0.48$ $\sigma = 0.04$	$b = 0.52$ $\mu = 0.46$ $\sigma = 0.02$	$b = 0.68$ $\mu = 0.56$ $\sigma = 0.03$

performance on the sextic and mult-3 problems. Importantly, $DFMGP_{GA}$ achieves the best result on all the tackled problems. Statistical tests are conducted to ascertain the performance advantage of $DFMGP_{GA}$ over $DFMGP_{RD1}$ and standard GP: the result obtained by running $DFMGP_{GA}$, (μ_0, σ_0) , is compared to that obtained by running $DFMGP_{RD1}$ /standard GP on the same problem (μ_1, σ_1) : here, a pairwise z-test, specified as follows $H_0 : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Table 9.2 shows the resulting p-values; here, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted.

TABLE 9.2: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1	$DFM-GP_{GA}$ vs. NS2
Sextic	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	
Keijzer	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Par-7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Mult-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Tart	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.03
Dec-tart	0.00	0.00	0.03	0.05	0.00	0.00	0.00	0.00	0.00

Tables 9.1 and 9.2 indicate that $DFMGP_{GA}$ largely achieves better quality than both $DFMGP_{RD1}$ and standard GP at the 5% level of significance. Standard GP with BP2 offers competitive performance on the deceptive tartarus problem, while standard GP with DSS and standard GP with NS2 offer competitive performance on the tartarus problem; from the observations in section 4.3.1 of chapter 4, recall that BP2 achieves the highest solution quality on the deceptive tartarus problem, while DSS and NS2 achieve the highest solution quality on the tartarus problem. Nevertheless, $DFMGP_{GA}$ consistently achieves significantly better performance than standard GP on most of the problems. Hence $DFMGP_{GA}$ is observed to be more effective than standard GP on the problems. Furthermore, the GA evolution is shown to be an important component of deriving the DFMs, as $DFMGP_{GA}$ consistently outperforms $DFMGP_{RD1}$. The GA is however observed to be a power-hungry approach: the GA executes a colossal number of $DFMGP$ runs in the quest to discover an optimal DFM. Table 9.3 shows the total time taken to train the GA and subsequently run $DFMGP_{GA}$ applying the evolved DFM (this time is abbreviated as GA + $DFMGP_{GA}$); the time is compared to the time taken to run standard GP.

TABLE 9.3: GA + $DFMGP_{GA}$ vs. Standard GP: Execution Time (milliseconds)

	GA + $DFMGP_{GA}$	Standard GP							
		OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Sextic	2.30×10^7	9.08×10^1	1.93×10^3	7.93×10^3	9.09×10^1	8.71×10^1	9.55×10^1	9.30×10^1	
Keijzer	4.25×10^7	1.53×10^3	6.75×10^3	8.34×10^3	1.03×10^3	4.28×10^2	5.18×10^2	3.69×10^3	
Par-7	1.90×10^8	4.07×10^4	6.68×10^4	9.95×10^4	4.19×10^4	3.58×10^3	4.04×10^3	5.78×10^4	
Mult-3	1.69×10^8	2.67×10^4	3.54×10^4	4.01×10^4	2.73×10^4	8.81×10^3	9.21×10^3	3.04×10^4	
Tart	1.41×10^8	3.74×10^4	7.74×10^4	7.94×10^4	3.81×10^4	8.91×10^3	9.02×10^3	6.18×10^4	6.20×10^4
Dec-tart	1.17×10^8	3.66×10^4	7.01×10^4	7.22×10^4	4.01×10^4	9.02×10^3	9.13×10^3	6.84×10^4	6.11×10^4

The key observation in table 9.3 is that the total time taken to train the GA and subsequently execute $DFMGP_{GA}$ is markedly higher than the time taken to run standard GP. This is expected because of the additional computational effort incurred by the GA. The GA's computational expense can however prove worthwhile if the evolved DFMs generalize to unseen problem instances; in this case, given a problem class, the GA need only be executed *once*, in order to evolve a general problem solver for the class. In this vein, the ensuing sections use training and test sets to verify the reusability of the evolved DFMs.

9.2.2 Testing the reusability of the derived DFMs within problem classes

This section presents the results obtained from comparing the performance of $DFMGP_{GA}$, $DFMGP_{RD1}$ and standard GP on the problem classes listed in table 5.2 of chapter 5.

Table 9.4 shows the results obtained by running $DFMGP_{GA}$ on unseen instances of the sextic problem class; here the GA is trained on the sextic training set defined in table 5.2 of chapter 5; subsequently, $DFMGP_{GA}$ applying the derived DFM is run on the test set; tests 1 to 10 represent test set problems defined in table 5.2. Table 9.4 shows the best solution quality, b , achieved over 30 runs of GP/DFMGP; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. Table 9.4 highlights the GP approaches that achieve the best mean quality scores on each problem.

Table 9.4 shows that $DFMGP_{GA}$ achieves among the best results on all problems. Standard GP with BP2 performs just as well as $DFMGP_{GA}$ for the second test problem producing the same best and average fitness over the 30 runs. Similarly, standard GP with BP2 produces the same average fitness over the 30 runs for test 10, however $DFMGP_{GA}$ produces a better best fitness. Similar to the solution quality results obtained for the sextic problem in section 4.3.1 of chapter 4, BP2's high performance on the sextic problem class is attributed to these problems being inherently modular; the sextic problems are modular, because useful lower-order functions, such as x^2 , are reused to construct the sextic function. Thus BP is expected to do well in the sextic problem class, because BP exploits modularity [5]. Nevertheless, $DFMGP_{GA}$ generally outperforms BP2 by producing the best results for all the test problems. The justification for $DFMGP_{GA}$'s performance advantage is discussed in section 9.2.4.

Statistical tests, identical to those reported in table 9.2, are conducted to ascertain the performance advantage of $DFMGP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the sextic class. Table 9.5 shows the resulting p-values; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. The results in table 9.5 ascertain that $DFMGP_{GA}$ largely outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance. Also, standard GP with BP2 achieves on par performance with $DFMGP_{GA}$ on two of the problems.

Table 9.6 shows the results obtained by running DFMGP on unseen instances of the Keijzer problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of chapter 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.6 shows that $DFMGP_{GA}$ achieves among the best results on all problems. Conversely, standard GP achieves varied results with no particular fitness measure demonstrating a consistent performance advantage.

TABLE 9.4: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Sextic Class - Test Set Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.98$ $\mu = 0.94$ $\sigma = 0.03$	$b = 0.89$ $\mu = 0.71$ $\sigma = 0.18$	$b = 0.90$ $\mu = 0.85$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.68$ $\sigma = 0.11$	$b = 0.93$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.89$ $\mu = 0.85$ $\sigma = 0.03$	$b = 0.89$ $\mu = 0.82$ $\sigma = 0.06$	$b = 0.92$ $\mu = 0.86$ $\sigma = 0.04$	$b = 0.90$ $\mu = 0.82$ $\sigma = 0.06$
Test 2	$b = 0.80$ $\mu = 0.68$ $\sigma = 0.09$	$b = 0.74$ $\mu = 0.48$ $\sigma = 0.22$	$b = 0.74$ $\mu = 0.65$ $\sigma = 0.09$	$b = 0.67$ $\mu = 0.56$ $\sigma = 0.07$	$b = 0.80$ $\mu = 0.68$ $\sigma = 0.11$	$b = 0.36$ $\mu = 0.25$ $\sigma = 0.10$	$b = 0.74$ $\mu = 0.63$ $\sigma = 0.13$	$b = 0.72$ $\mu = 0.62$ $\sigma = 0.09$	$b = 0.31$ $\mu = 0.22$ $\sigma = 0.06$
Test 3	$b = 0.99$ $\mu = 0.90$ $\sigma = 0.08$	$b = 0.81$ $\mu = 0.42$ $\sigma = 0.24$	$b = 0.85$ $\mu = 0.81$ $\sigma = 0.06$	$b = 0.84$ $\mu = 0.78$ $\sigma = 0.06$	$b = 0.89$ $\mu = 0.85$ $\sigma = 0.05$	$b = 0.43$ $\mu = 0.30$ $\sigma = 0.07$	$b = 0.89$ $\mu = 0.74$ $\sigma = 0.13$	$b = 0.81$ $\mu = 0.70$ $\sigma = 0.09$	$b = 0.29$ $\mu = 0.20$ $\sigma = 0.05$
Test 4	$b = 0.69$ $\mu = 0.52$ $\sigma = 0.09$	$b = 0.67$ $\mu = 0.29$ $\sigma = 0.18$	$b = 0.60$ $\mu = 0.37$ $\sigma = 0.09$	$b = 0.62$ $\mu = 0.38$ $\sigma = 0.07$	$b = 0.67$ $\mu = 0.42$ $\sigma = 0.08$	$b = 0.29$ $\mu = 0.10$ $\sigma = 0.04$	$b = 0.50$ $\mu = 0.33$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.30$ $\sigma = 0.09$	$b = 0.17$ $\mu = 0.13$ $\sigma = 0.04$
Test 5	$b = 0.72$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.65$ $\mu = 0.25$ $\sigma = 0.22$	$b = 0.52$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.43$ $\mu = 0.27$ $\sigma = 0.06$	$b = 0.52$ $\mu = 0.40$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.10$ $\sigma = 0.07$	$b = 0.40$ $\mu = 0.22$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.20$ $\sigma = 0.07$	$b = 0.09$ $\mu = 0.05$ $\sigma = 0.02$
Test 6	$b = 0.70$ $\mu = 0.46$ $\sigma = 0.09$	$b = 0.59$ $\mu = 0.20$ $\sigma = 0.19$	$b = 0.20$ $\mu = 0.13$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.06$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.07$	$b = 0.14$ $\mu = 0.08$ $\sigma = 0.04$	$b = 0.33$ $\mu = 0.17$ $\sigma = 0.05$	$b = 0.30$ $\mu = 0.17$ $\sigma = 0.05$	$b = 0.04$ $\mu = 0.02$ $\sigma = 0.02$
Test 7	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.09$	$b = 0.59$ $\mu = 0.35$ $\sigma = 0.18$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.06$	$b = 0.54$ $\mu = 0.46$ $\sigma = 0.05$	$b = 0.61$ $\mu = 0.50$ $\sigma = 0.06$	$b = 0.21$ $\mu = 0.11$ $\sigma = 0.07$	$b = 0.59$ $\mu = 0.44$ $\sigma = 0.13$	$b = 0.53$ $\mu = 0.42$ $\sigma = 0.09$	$b = 0.19$ $\mu = 0.11$ $\sigma = 0.05$
Test 8	$b = 0.39$ $\mu = 0.25$ $\sigma = 0.08$	$b = 0.33$ $\mu = 0.15$ $\sigma = 0.21$	$b = 0.24$ $\mu = 0.16$ $\sigma = 0.09$	$b = 0.29$ $\mu = 0.19$ $\sigma = 0.07$	$b = 0.33$ $\mu = 0.20$ $\sigma = 0.09$	$b = 0.20$ $\mu = 0.09$ $\sigma = 0.05$	$b = 0.24$ $\mu = 0.18$ $\sigma = 0.09$	$b = 0.19$ $\mu = 0.11$ $\sigma = 0.09$	$b = 0.13$ $\mu = 0.07$ $\sigma = 0.05$
Test 9	$b = 0.69$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.60$ $\mu = 0.21$ $\sigma = 0.19$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.10$	$b = 0.27$ $\mu = 0.20$ $\sigma = 0.07$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.08$	$b = 0.20$ $\mu = 0.15$ $\sigma = 0.07$	$b = 0.26$ $\mu = 0.13$ $\sigma = 0.09$	$b = 0.26$ $\mu = 0.16$ $\sigma = 0.07$	$b = 0.17$ $\mu = 0.11$ $\sigma = 0.06$
Test 10	$b = 0.90$ $\mu = 0.77$ $\sigma = 0.09$	$b = 0.81$ $\mu = 0.40$ $\sigma = 0.19$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.10$	$b = 0.81$ $\mu = 0.72$ $\sigma = 0.07$	$b = 0.87$ $\mu = 0.77$ $\sigma = 0.08$	$b = 0.55$ $\mu = 0.43$ $\sigma = 0.11$	$b = 0.80$ $\mu = 0.69$ $\sigma = 0.09$	$b = 0.81$ $\mu = 0.69$ $\sigma = 0.09$	$b = 0.41$ $\mu = 0.33$ $\sigma = 0.06$

TABLE 9.5: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Sextic Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.03	0.00	0.29	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.01	0.27	0.00	0.00	0.00	0.00

Table 9.7 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the Keijzer class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.7 indicates that while standard GP performs on par with $DFMGP_{GA}$ in a number of instances, $DFMGP_{GA}$ outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance on most of the problems. On the problem instances where standard GP performs on par with $DFMGP_{GA}$, one may argue that it is easier to simply try standard GP with the different fitness measures on the given problem, and select the best result; this would also be less computationally expensive

TABLE 9.6: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Keijzer-6 Class - Test Set Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.62$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.55$ $\mu = 0.28$ $\sigma = 0.21$	$b = 0.55$ $\mu = 0.43$ $\sigma = 0.10$	$b = 0.54$ $\mu = 0.43$ $\sigma = 0.09$	$b = 0.57$ $\mu = 0.44$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.32$ $\sigma = 0.20$	$b = 0.60$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.52$ $\mu = 0.41$ $\sigma = 0.08$	$b = 0.28$ $\mu = 0.18$ $\sigma = 0.06$
Test 2	$b = 0.89$ $\mu = 0.76$ $\sigma = 0.08$	$b = 0.90$ $\mu = 0.31$ $\sigma = 0.25$	$b = 0.84$ $\mu = 0.73$ $\sigma = 0.09$	$b = 0.89$ $\mu = 0.76$ $\sigma = 0.10$	$b = 0.82$ $\mu = 0.73$ $\sigma = 0.08$	$b = 0.64$ $\mu = 0.56$ $\sigma = 0.11$	$b = 0.90$ $\mu = 0.75$ $\sigma = 0.07$	$b = 0.86$ $\mu = 0.75$ $\sigma = 0.10$	$b = 0.53$ $\mu = 0.39$ $\sigma = 0.11$
Test 3	$b = 0.81$ $\mu = 0.69$ $\sigma = 0.11$	$b = 0.75$ $\mu = 0.30$ $\sigma = 0.19$	$b = 0.75$ $\mu = 0.63$ $\sigma = 0.07$	$b = 0.72$ $\mu = 0.61$ $\sigma = 0.06$	$b = 0.75$ $\mu = 0.64$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.34$ $\sigma = 0.11$	$b = 0.72$ $\mu = 0.63$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.64$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.17$ $\sigma = 0.05$
Test 4	$b = 0.79$ $\mu = 0.67$ $\sigma = 0.09$	$b = 0.70$ $\mu = 0.42$ $\sigma = 0.30$	$b = 0.71$ $\mu = 0.65$ $\sigma = 0.06$	$b = 0.70$ $\mu = 0.62$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.44$ $\mu = 0.32$ $\sigma = 0.09$	$b = 0.78$ $\mu = 0.67$ $\sigma = 0.05$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.07$	$b = 0.33$ $\mu = 0.20$ $\sigma = 0.10$
Test 5	$b = 0.62$ $\mu = 0.51$ $\sigma = 0.07$	$b = 0.50$ $\mu = 0.22$ $\sigma = 0.35$	$b = 0.42$ $\mu = 0.31$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.40$ $\sigma = 0.08$	$b = 0.58$ $\mu = 0.44$ $\sigma = 0.13$	$b = 0.20$ $\mu = 0.14$ $\sigma = 0.04$	$b = 0.49$ $\mu = 0.36$ $\sigma = 0.10$	$b = 0.43$ $\mu = 0.31$ $\sigma = 0.11$	$b = 0.29$ $\mu = 0.07$ $\sigma = 0.05$
Test 6	$b = 0.59$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.50$ $\mu = 0.22$ $\sigma = 0.21$	$b = 0.40$ $\mu = 0.29$ $\sigma = 0.09$	$b = 0.41$ $\mu = 0.34$ $\sigma = 0.08$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.12$ $\mu = 0.09$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.27$ $\sigma = 0.07$	$b = 0.20$ $\mu = 0.10$ $\sigma = 0.11$	$b = 0.07$ $\mu = 0.05$ $\sigma = 0.01$
Test 7	$b = 0.41$ $\mu = 0.26$ $\sigma = 0.09$	$b = 0.21$ $\mu = 0.10$ $\sigma = 0.22$	$b = 0.14$ $\mu = 0.12$ $\sigma = 0.01$	$b = 0.29$ $\mu = 0.18$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.20$ $\sigma = 0.06$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.02$	$b = 0.25$ $\mu = 0.18$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.05$	$b = 0.08$ $\mu = 0.05$ $\sigma = 0.02$
Test 8	$b = 0.27$ $\mu = 0.18$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.10$ $\sigma = 0.19$	$b = 0.11$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.11$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.16$ $\mu = 0.10$ $\sigma = 0.05$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.18$ $\mu = 0.12$ $\sigma = 0.05$	$b = 0.15$ $\mu = 0.12$ $\sigma = 0.03$	$b = 0.08$ $\mu = 0.04$ $\sigma = 0.02$
Test 9	$b = 0.22$ $\mu = 0.14$ $\sigma = 0.06$	$b = 0.18$ $\mu = 0.09$ $\sigma = 0.15$	$b = 0.16$ $\mu = 0.09$ $\sigma = 0.04$	$b = 0.16$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.18$ $\mu = 0.11$ $\sigma = 0.05$	$b = 0.12$ $\mu = 0.08$ $\sigma = 0.02$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.08$ $\mu = 0.03$ $\sigma = 0.02$
Test 10	$b = 0.20$ $\mu = 0.12$ $\sigma = 0.09$	$b = 0.18$ $\mu = 0.08$ $\sigma = 0.10$	$b = 0.14$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.14$ $\mu = 0.10$ $\sigma = 0.02$	$b = 0.18$ $\mu = 0.10$ $\sigma = 0.05$	$b = 0.10$ $\mu = 0.07$ $\sigma = 0.02$	$b = 0.14$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.17$ $\mu = 0.12$ $\sigma = 0.03$	$b = 0.05$ $\mu = 0.03$ $\sigma = 0.01$

than training the GA on the problem class, and subsequently running $DFMGP_{GA}$ applying the derived DFM. Nevertheless, $DFMGP_{GA}$ produces a *better* result than standard GP on most of the problems.

TABLE 9.7: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Keijzer-6 Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.36	0.00	0.00	0.32	0.20	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.03	0.00	0.04	0.00	0.27	0.04	0.00
Test 5	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.01	0.03	0.00	0.03	0.39	0.00

Table 9.8 shows the results obtained by running DFMGP on unseen instances of the even-N parity problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.8 shows that $DFMGP_{GA}$ achieves the best result on all problems. Standard GP with BP2 also performs competitively in the class: this is attributed to BP exploiting the inherent modularity of the even-N parity problems [1]. Nevertheless, $DFMGP_{GA}$ is seen to consistently outperform BP2.

TABLE 9.8: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Even-N Parity Class - Test Set Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.65$ $\mu = 0.59$ $\sigma = 0.04$	$b = 0.59$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.02$	$b = 0.59$ $\mu = 0.55$ $\sigma = 0.04$	$b = 0.51$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.55$ $\mu = 0.54$ $\sigma = 0.01$	$b = 0.51$ $\mu = 0.50$ $\sigma = 0.00$
Test 2	$b = 0.59$ $\mu = 0.54$ $\sigma = 0.02$	$b = 0.55$ $\mu = 0.51$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.53$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 3	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.03$	$b = 0.56$ $\mu = 0.50$ $\sigma = 0.04$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 4	$b = 0.55$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.53$ $\mu = 0.50$ $\sigma = 0.03$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.53$ $\mu = 0.51$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 5	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.52$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$

Table 9.9 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the even-N parity class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. The results in table 9.9 show that $DFMGP_{GA}$ largely outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance.

TABLE 9.9: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Even-N Parity Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1
Test 1	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 9.10 shows the results obtained by running DFMGP on unseen instances of the mult-N problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approach is highlighted in the table. Table 9.10 shows that $DFMGP_{GA}$ achieves the best result on all problems. Conversely, standard GP achieves varied results with no particular fitness measure demonstrating a consistent performance advantage.

Table 9.11 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the mult-N class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.11 shows that $DFMGP_{GA}$ outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance on all problems.

TABLE 9.10: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: N-bit Multiplier Class - Test Set Quality scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.82$ $\mu = 0.79$ $\sigma = 0.01$	$b = 0.78$ $\mu = 0.60$ $\sigma = 0.09$	$b = 0.75$ $\mu = 0.74$ $\sigma = 0.01$	$b = 0.77$ $\mu = 0.75$ $\sigma = 0.01$	$b = 0.78$ $\mu = 0.75$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.71$ $\sigma = 0.01$	$b = 0.74$ $\mu = 0.73$ $\sigma = 0.01$	$b = 0.74$ $\mu = 0.72$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.57$ $\sigma = 0.01$
Test 2	$b = 0.79$ $\mu = 0.77$ $\sigma = 0.01$	$b = 0.70$ $\mu = 0.55$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.70$ $\sigma = 0.02$	$b = 0.72$ $\mu = 0.70$ $\sigma = 0.01$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.02$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.02$	$b = 0.60$ $\mu = 0.54$ $\sigma = 0.06$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.01$
Test 3	$b = 0.76$ $\mu = 0.73$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.68$ $\mu = 0.66$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.68$ $\mu = 0.66$ $\sigma = 0.01$	$b = 0.73$ $\mu = 0.70$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.52$ $\sigma = 0.04$	$b = 0.62$ $\mu = 0.60$ $\sigma = 0.01$
Test 4	$b = 0.59$ $\mu = 0.54$ $\sigma = 0.01$	$b = 0.54$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.52$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 5	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$

TABLE 9.11: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: N-bit Multiplier Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 9.12 shows the results obtained by running DFMGP on unseen instances of the tartarus problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approach is highlighted in the table. Table 9.12 shows that $DFMGP_{GA}$ achieves among the best results on all problems. Standard GP with NS2 is seen to perform competitively in the class; the performance advantage is attributed to NS2's task-specific behavior descriptors, which mitigate against deception [4] on the deceptive tasks. Standard GP with NS2 was also shown to perform competitively on a problem from the tartarus class in section 4.3.1 of chapter 4. Also, similarly to the results obtained for the tartarus problem in chapter 4, table 9.12 shows that standard GP with DSS performs competitively in the tartarus class. Nevertheless, the results in table 9.12 show that $DFMGP_{GA}$ largely outperforms standard GP by consistently producing the best result on all the test instances.

Table 9.13 shows the result of statistical tests conducted to ascertain the performance advantage of $DFM-GP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the tartarus class; in the table, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. The p-values in table 9.13 show that standard GP with DSS performs on par with $DFMGP_{GA}$ on test 7, while standard GP with NS2 performs on par with $DFMGP_{GA}$ on test 8. Nevertheless, $DFMGP_{GA}$ largely outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance on the tackled problems.

Table 9.14 shows the results obtained by running DFMGP on unseen instances of the deceptive tartarus problem class: test 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.14 shows that $DFMGP_{GA}$ achieves among the best results on all problems. Standard GP with BP2 and NS2 are also seen to perform competitively in

TABLE 9.12: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Tartarus Class - Test Set Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP							
			OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Test 1	$b = 0.53$ $\mu = 0.51$ $\sigma = 0.01$	$b = 0.47$ $\mu = 0.32$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.37$ $\sigma = 0.05$	$b = 0.44$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.36$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.23$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.36$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.06$	$b = 0.24$ $\mu = 0.23$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.02$
Test 2	$b = 0.45$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.25$ $\sigma = 0.05$	$b = 0.36$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.36$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.19$ $\mu = 0.18$ $\sigma = 0.02$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.04$	$b = 0.36$ $\mu = 0.27$ $\sigma = 0.04$	$b = 0.19$ $\mu = 0.18$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.36$ $\sigma = 0.01$
Test 3	$b = 0.73$ $\mu = 0.69$ $\sigma = 0.05$	$b = 0.69$ $\mu = 0.29$ $\sigma = 0.05$	$b = 0.67$ $\mu = 0.52$ $\sigma = 0.11$	$b = 0.63$ $\mu = 0.52$ $\sigma = 0.08$	$b = 0.60$ $\mu = 0.52$ $\sigma = 0.07$	$b = 0.09$ $\mu = 0.06$ $\sigma = 0.02$	$b = 0.64$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.65$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.11$ $\mu = 0.07$ $\sigma = 0.02$	$b = 0.63$ $\mu = 0.57$ $\sigma = 0.05$
Test 4	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.03$	$b = 0.51$ $\mu = 0.32$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.35$ $\sigma = 0.08$	$b = 0.44$ $\mu = 0.37$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.25$ $\sigma = 0.02$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.37$ $\sigma = 0.04$	$b = 0.26$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.54$ $\mu = 0.50$ $\sigma = 0.02$
Test 5	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.06$	$b = 0.60$ $\mu = 0.35$ $\sigma = 0.05$	$b = 0.53$ $\mu = 0.46$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.50$ $\mu = 0.44$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.29$ $\sigma = 0.04$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.04$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.28$ $\sigma = 0.03$	$b = 0.59$ $\mu = 0.50$ $\sigma = 0.05$
Test 6	$b = 0.62$ $\mu = 0.50$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.32$ $\sigma = 0.08$	$b = 0.44$ $\mu = 0.35$ $\sigma = 0.06$	$b = 0.44$ $\mu = 0.35$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.25$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.04$	$b = 0.37$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.29$ $\mu = 0.29$ $\sigma = 0.00$	$b = 0.38$ $\mu = 0.35$ $\sigma = 0.05$
Test 7	$b = 0.48$ $\mu = 0.38$ $\sigma = 0.06$	$b = 0.44$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.06$	$b = 0.18$ $\mu = 0.17$ $\sigma = 0.01$	$b = 0.44$ $\mu = 0.38$ $\sigma = 0.04$	$b = 0.37$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.21$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.35$ $\sigma = 0.05$
Test 8	$b = 0.52$ $\mu = 0.44$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.25$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.43$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.26$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.39$ $\sigma = 0.04$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.44$ $\sigma = 0.05$
Test 9	$b = 0.54$ $\mu = 0.45$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.22$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.37$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.22$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.04$	$b = 0.40$ $\mu = 0.35$ $\sigma = 0.05$	$b = 0.26$ $\mu = 0.25$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.05$
Test 10	$b = 0.54$ $\mu = 0.48$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.43$ $\mu = 0.29$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.50$ $\mu = 0.42$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.32$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.49$ $\mu = 0.44$ $\sigma = 0.06$

TABLE 9.13: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Tartarus Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-GP_{GA}$ vs. OF	$DFM-GP_{GA}$ vs. BP1	$DFM-GP_{GA}$ vs. BP2	$DFM-GP_{GA}$ vs. FS	$DFM-GP_{GA}$ vs. DSS	$DFM-GP_{GA}$ vs. HP	$DFM-GP_{GA}$ vs. NS1	$DFM-GP_{GA}$ vs. NS2
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.35	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.39
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

the class. As in section 4.3.1 of chapter 4, BP2's performance advantage is attributed to the following: the problems in the deceptive tartarus class are highly deceptive due to penalization of some of the necessary intermediate steps towards achieving the goal behavior [212]; BP favors the retention of subtrees that achieve useful intermediate results: thus the fitness measure mitigates against the loss of useful subtrees, which would otherwise be discarded by OF measures. NS2's performance advantage shown in table 9.14 is attributed to the task-specific behavior descriptors mitigating against deception on the deceptive tasks. Nevertheless, table 9.14 shows that $DFMGP_{GA}$ largely outperforms standard GP by consistently producing the best result on all the

test instances.

Table 9.15 shows the result of statistical tests conducted to ascertain the performance advantage of $DFM-GP_{GA}$ over $DFMGP_{RD1}$ and standard GP in the deceptive tartarus class; in the table, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. The p-values shown in table 9.15 show that standard GP with BP2 performs on par with $DFMGP_{GA}$ on test 8. However, $DFMGP_{GA}$ largely outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance on the tackled problems.

TABLE 9.14: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Deceptive Tartarus Class - Test Set Quality Scores

	$DFM-GP_{GA}$	$DFM-GP_{RD1}$	Standard GP							
			OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Test 1	$b = 0.50$ $\mu = 0.48$ $\sigma = 0.01$	$b = 0.47$ $\mu = 0.38$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.41$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.43$ $\sigma = 0.02$	$b = 0.41$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.41$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.37$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.46$ $\mu = 0.43$ $\sigma = 0.02$
Test 2	$b = 0.49$ $\mu = 0.46$ $\sigma = 0.01$	$b = 0.49$ $\mu = 0.33$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.47$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.35$ $\mu = 0.34$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.37$ $\mu = 0.36$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.42$ $\sigma = 0.01$
Test 3	$b = 0.75$ $\mu = 0.72$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.53$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.65$ $\mu = 0.63$ $\sigma = 0.02$	$b = 0.71$ $\mu = 0.68$ $\sigma = 0.03$	$b = 0.43$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.52$ $\mu = 0.49$ $\sigma = 0.02$	$b = 0.50$ $\mu = 0.44$ $\sigma = 0.02$	$b = 0.45$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.69$ $\mu = 0.65$ $\sigma = 0.02$
Test 4	$b = 0.46$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.34$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.38$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.35$ $\sigma = 0.02$	$b = 0.47$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.33$ $\mu = 0.32$ $\sigma = 0.01$	$b = 0.40$ $\mu = 0.38$ $\sigma = 0.02$	$b = 0.40$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.31$ $\mu = 0.31$ $\sigma = 0.01$	$b = 0.44$ $\mu = 0.40$ $\sigma = 0.01$
Test 5	$b = 0.73$ $\mu = 0.69$ $\sigma = 0.02$	$b = 0.71$ $\mu = 0.40$ $\sigma = 0.01$	$b = 0.51$ $\mu = 0.47$ $\sigma = 0.03$	$b = 0.68$ $\mu = 0.61$ $\sigma = 0.05$	$b = 0.70$ $\mu = 0.64$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.54$ $\mu = 0.49$ $\sigma = 0.04$	$b = 0.50$ $\mu = 0.45$ $\sigma = 0.04$	$b = 0.43$ $\mu = 0.43$ $\sigma = 0.01$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.02$
Test 6	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.50$ $\mu = 0.38$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.38$ $\sigma = 0.03$	$b = 0.38$ $\mu = 0.37$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.44$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.35$ $\sigma = 0.00$	$b = 0.45$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.44$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.35$ $\sigma = 0.00$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.02$
Test 7	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.50$ $\mu = 0.38$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.38$ $\sigma = 0.03$	$b = 0.38$ $\mu = 0.37$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.44$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.35$ $\sigma = 0.00$	$b = 0.45$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.44$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.35$ $\sigma = 0.00$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.02$
Test 8	$b = 0.51$ $\mu = 0.45$ $\sigma = 0.03$	$b = 0.47$ $\mu = 0.36$ $\sigma = 0.01$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.02$	$b = 0.48$ $\mu = 0.45$ $\sigma = 0.02$	$b = 0.34$ $\mu = 0.34$ $\sigma = 0.00$	$b = 0.43$ $\mu = 0.40$ $\sigma = 0.02$	$b = 0.44$ $\mu = 0.40$ $\sigma = 0.03$	$b = 0.39$ $\mu = 0.41$ $\sigma = 0.01$	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.03$
Test 9	$b = 0.49$ $\mu = 0.45$ $\sigma = 0.03$	$b = 0.46$ $\mu = 0.35$ $\sigma = 0.01$	$b = 0.37$ $\mu = 0.36$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.35$ $\sigma = 0.03$	$b = 0.49$ $\mu = 0.42$ $\sigma = 0.01$	$b = 0.33$ $\mu = 0.32$ $\sigma = 0.00$	$b = 0.42$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.39$ $\mu = 0.37$ $\sigma = 0.01$	$b = 0.33$ $\mu = 0.32$ $\sigma = 0.00$	$b = 0.46$ $\mu = 0.41$ $\sigma = 0.02$
Test 10	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.04$	$b = 0.46$ $\mu = 0.35$ $\sigma = 0.01$	$b = 0.36$ $\mu = 0.33$ $\sigma = 0.03$	$b = 0.39$ $\mu = 0.35$ $\sigma = 0.03$	$b = 0.42$ $\mu = 0.38$ $\sigma = 0.04$	$b = 0.30$ $\mu = 0.29$ $\sigma = 0.00$	$b = 0.39$ $\mu = 0.37$ $\sigma = 0.02$	$b = 0.38$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.30$ $\mu = 0.29$ $\sigma = 0.00$	$b = 0.40$ $\mu = 0.37$ $\sigma = 0.02$

The results in this section show that the GA-evolved DFMs are reusable within the problem classes: the DFMs solve unseen problem instances to optimality. Furthermore, the importance of using the GA to evolve the DFMs is ascertained by the fact that $DFMGP$ applying randomly generated DFMs does not fair as well as $DFMGP$ applying the evolved DFMs on the problem classes.

TABLE 9.15: $DFMGP_{GA}$ vs. $DFMGP_{RD1}$ /Standard GP: Deceptive Tartarus Class - Statistical Tests

	$DFMGP_{GA}$ vs. $DFMGP_{RD1}$	$DFM-$ GP_{GA} vs. OF	$DFM-$ GP_{GA} vs. BP1	$DFM-$ GP_{GA} vs. BP2	$DFM-$ GP_{GA} vs. FS	$DFM-$ GP_{GA} vs. DSS	$DFM-$ GP_{GA} vs. HP	$DFM-$ GP_{GA} vs. NS1	$DFM-$ GP_{GA} vs. NS2
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.36	0.00	0.00	0.00	0.00	0.00
Test 9	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

9.2.3 Testing the reusability of the derived DFMs on real world problems

This section presents the results obtained from comparing the performance of $DFMGP_{GA}$, $DFMGP_{RD1}$ and standard GP on the real-world problems listed in table 5.3 of chapter 5.

Table 9.16 shows the results obtained on the real-world problems. For the regression problems in the table, $DFMGP_{P1}$ represents $DFMGP_{GA}$ employing the DFM trained on the sextic training set, $DFMGP_{P2}$ represents $DFMGP_{GA}$ employing the DFM trained on the Keijzer training set, and $DFMGP_{UN}$ represents $DFMGP_{GA}$ employing the DFM trained on the combined sextic and Keijzer training sets. For the Boolean problems in the table, $DFMGP_{P1}$ represents $DFMGP_{GA}$ employing the DFM trained on the even-N parity training set, $DFMGP_{P2}$ represents $DFMGP_{GA}$ employing the DFM trained on the mult-N training set, and $DFMGP_{UN}$ represents $DFMGP_{GA}$ employing the DFM trained on the combined even-N and mult-N training sets. In the ensuing discussion, the term DFMGP approach refers to $DFMGP_{GA}$ together with the data it is trained on. Table 9.16 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs. The best performing GP approach is highlighted in the table.

TABLE 9.16: DFMGP vs Standard GP: Real-World Problem Quality Scores

	$DFM-$ GP_{P1}	$DFM-$ GP_{P2}	$DFM-$ GP_{UN}	Standard GP						
				OF	BP1	BP2	FS	DSS	HP	NS1
Dow	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.03$	$b = 0.43$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.37$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.37$ $\sigma = 0.02$	$b = 0.41$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.41$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.36$ $\mu = 0.34$ $\sigma = 0.01$
Abalone	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.02$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.17$ $\mu = 0.15$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.14$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.14$ $\sigma = 0.01$	$b = 0.13$ $\mu = 0.12$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.12$ $\mu = 0.12$ $\sigma = 0.01$
Sensor	$b = 0.95$ $\mu = 0.90$ $\sigma = 0.03$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.95$ $\mu = 0.90$ $\sigma = 0.03$	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.03$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.99$ $\mu = 0.93$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.95$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.84$ $\mu = 0.81$ $\sigma = 0.01$
Decoder	$b = 0.95$ $\mu = 0.92$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.95$ $\mu = 0.91$ $\sigma = 0.02$	$b = 0.92$ $\mu = 0.88$ $\sigma = 0.02$	$b = 0.94$ $\mu = 0.89$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.90$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.95$ $\sigma = 0.01$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.80$ $\mu = 0.76$ $\sigma = 0.02$

Table 9.16 shows that in each case, one of the DFMGP approaches achieves the best result. $DFMGP_{UN}$ achieves the best result on the Dow and abalone problems, while $DFMGP_{P2}$ achieves the best result on the sensor and decoder problems. Statistical tests are conducted to ascertain the significance of the differences observed. For each DFMGP approach, the result obtained on each problem, (μ_0, σ_0) , is compared to the same for the other DFMGP approaches or standard GP, (μ_1, σ_1) : whereby, a pairwise z-test, specified as follows $H_O : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Tables 9.17, 9.18 and 9.19 show the resulting p-values for $DFMGP_{P1}$,

$DFMGP_{P2}$ and $DFMGP_{UN}$ respectively; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted.

TABLE 9.17: $DFMGP_{P1}$ vs. $DFMGP_{P2}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{P1}$ vs. $DFMGP_{P2}$	$DFMGP_{P1}$ vs. $DFMGP_{UN}$	$DFM-$ GP_{P1} vs. OF	$DFM-$ GP_{P1} vs. BP1	$DFM-$ GP_{P1} vs. BP2	$DFM-$ GP_{P1} vs. FS	$DFM-$ GP_{P1} vs. DSS	$DFM-$ GP_{P1} vs. HP	$DFM-$ GP_{P1} vs. NS1
Dow	0.32	0.00	0.14	0.39	0.19	0.00	0.16	0.28	0.00
Abalone	0.35	0.00	0.22	0.14	0.10	0.00	0.10	0.11	0.00
Sensor	0.00	0.32	0.25	0.29	0.00	0.00	0.36	0.27	0.00
Decoder	0.00	0.36	0.00	0.02	0.15	0.00	0.22	0.21	0.00

TABLE 9.18: $DFMGP_{P2}$ vs. $DFMGP_{P1}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{P2}$ vs. $DFMGP_{P1}$	$DFMGP_{P2}$ vs. $DFMGP_{UN}$	$DFM-$ GP_{P2} vs. OF	$DFM-$ GP_{P2} vs. BP1	$DFM-$ GP_{P2} vs. BP2	$DFM-$ GP_{P2} vs. FS	$DFM-$ GP_{P2} vs. DSS	$DFM-$ GP_{P2} vs. HP	$DFM-$ GP_{P2} vs. NS1
Dow	0.32	0.00	0.36	0.41	0.34	0.00	0.33	0.13	0.00
Abalone	0.35	0.00	0.13	0.06	0.05	0.00	0.08	0.10	0.00
Sensor	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Decoder	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLE 9.19: $DFMGP_{UN}$ vs. $DFMGP_{P1}/DFMGP_{P2}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{UN}$ vs. $DFMGP_{P1}$	$DFMGP_{UN}$ vs. $DFMGP_{P2}$	$DFM-$ GP_{UN} vs. OF	$DFM-$ GP_{UN} vs. BP1	$DFM-$ GP_{UN} vs. BP2	$DFM-$ GP_{UN} vs. FS	$DFM-$ GP_{UN} vs. DSS	$DFM-$ GP_{UN} vs. HP	$DFM-$ GP_{UN} vs. NS1
Dow	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Abalone	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor	0.32	0.00	0.34	0.27	0.00	0.00	0.21	0.24	0.00
Decoder	0.36	0.00	0.00	0.04	0.15	0.00	0.24	0.26	0.17

Tables 9.17, 9.18 and 9.19 show that $DFMGP_{P2}$ consistently outperforms standard GP and the other DFMGP approaches on the sensor and decoder problems; on the other hand, $DFMGP_{P1}$ and $DFMGP_{UN}$ do not perform reliably on the problems. The tables also show that $DFMGP_{UN}$ consistently outperforms standard GP and the other DFMGP approaches on the Dow and abalone problems, whereas $DFMGP_{P1}$ and $DFMGP_{P2}$ do not perform reliably on the same problems. Overall, the inference gained from the data is that DFMGP has the capacity to outperform standard GP. However, it is difficult to a priori ascertain the particular DFMGP approach that will achieve superior performance. The performance of DFMGP on unseen problems has to do with how similar the unseen problems are compared to the training problems with respect to problem properties addressed by the different fitness measures e.g. local optima, deceptiveness, bloat. Therefore, definitive training sets for DFMGP can only be obtained if prior knowledge exists of shared problem properties. Ultimately, it would be useful if simple heuristics exist that can be used to a priori detect the properties of the real-world problems. Subsequently, training sets with the same properties can be used to deduce suitable DFMs for DFMGP. Preliminary research already exists with respect to detecting problem properties. For example, section 3.3.1 of chapter 3 discussed the heuristic proposed by Krawiek and Wieloch [158] to detect modularity in Boolean function synthesis GP; the discussion established that the heuristic proposed in [158] is not applicable to problems with continuous (i.e. non-discrete) outputs. Overall, more work is required with respect to detecting the properties of different problems.

In turn, exploitation is a de facto local search that consists of probing a limited - but promising - region of the search space: exploitation is used to refine promising solutions in later GP generations, when good points in the search space have been discovered [22, 266]. FS and NS are ideal choices for the initial GP generations because the fitness measures support exploration. Section 3.4 of chapter 3 established that FS maintains the diversity of the GP population, permitting GP to explore new niches (or optima). In turn, section 3.7 of chapter 3 established that NS favors exploration by explicitly rewarding diversification from prior behaviors. Both FS and NS fall short when it comes to exploitation. FS enforces the maintenance of niches on all GP generations: this can inhibit necessary exploitation and convergence in later GP generations. In this vein, McKay [6, 15] empirically shows that FS works better when confined to the initial GP generations, and proceeded by a more exploitative fitness measure in later generations. NS inhibits exploitation, because the fitness measure maintains the same level of selective pressure in favor of diversification, even when near-optimal solutions are found [204]. In a study applying NS to an evolutionary algorithm (EA), Mouret [204] postulates that NS can work better when confined to the initial generations of the EA, proceeded by a more exploitative fitness measure in later generations. This hypothesis is empirically proved in [204]: Mouret [204] implements a multi-objective approach that automatically switches between NS and a more exploitative OF measure. In switching between the fitness measures, Mouret’s approach [204] achieves better exploitation capability in later generations compared to NS applied individually throughout the EA. In table 9.20, FS and NS dominate and are mostly confined to the initial generations of the DFMs.

In a similar vein, section 3.3.2 of chapter 3 motivates that BP performs well when useful modules (or sub-programs) can be detected. Whereas useful modules can exist in the randomly generated initial population in trivial problems, more complex problems may require that further exploration is conducted in the preliminary GP generations, as a precursor to BP: thus, useful modules can be made available for BP to exploit in later GP generations. In table 9.20, BP is selected for the middle and later DFMGP generations in the problem classes tackled.

Table 9.21 shows the DFM evolved from the combined training set of the sextic and Keijzer classes, and the DFM evolved from the combined training set of the par-N and mult-N classes. The DFMs in table 9.21 are similar to those shown in table 9.20, whereby the explorative fitness measures FS and NS (encoded as D and G respectively) dominate the preliminary DFMGP generations. Furthermore, the more exploitative BP1 and BP2 (encoded as B and C respectively) are selected for the middle and later DFMGP generations.

In summary, tables 9.20 and 9.21 show that the GA approach used to evolve the DFMs for DFMGP has the ability to select the fitness measures that suit the particular on-going phase of GP. For the problem classes tackled, explorative fitness measures are selected for and confined to the initial DFMGP generations, while the more exploitative fitness measures are selected in later DFMGP generations. Different DFMs are also evolved with the different training sets. Therefore, the expectation is that DFMs will generalize to unseen problems that share the specific properties of the problems used in the training sets.

TABLE 9.21: Evolved Fitness Measure Sequences - Combined Training Sets

Class	Evolved DFM
Sextic + Keijzer	GGGGDDDDDDDDGGGGGGGGDDDDBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB 0 1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829303132333435363738394041424344454647484950 BBBBCCCCCCCCCCCCCCCCBBBBBBBBBBBBBBBBBBBBBBBBCCCCC 51525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100
Par-N + Mult-N	GGGGDDDDDDGGGGDGGGGDGGCCDCCCCCGGCCCCCCCCCCBBBBBBBBB 0 1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829303132333435363738394041424344454647484950 BBBBCCCCCCCCCCCCCCCCCECCAAAAAACCCCCCCCCC 51525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100

9.3 Results of the GP approach and DFMGP_{GP}

This section presents the results obtained by the approach described in chapter 8, namely a GP approach for deriving DFMs for DFMGP. Section 9.3.1 discusses the results obtained from testing the effectiveness of

the GP approach and $DFMGP_{GP}$. Subsequently, section 9.3.2 presents the results obtained from testing the reusability of the derived DFMs within problem classes. Next, section 9.3.3 presents the results obtained from testing the reusability of the derived DFMs on real world problems. Finally, section 9.3.4 presents an analysis of the derived DFMs to identify the fitness measures that suit the different phases of search for different problems.

9.3.1 Testing the effectiveness of $DFMGP_{GP}$

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$ with that of standard GP on the benchmark problems listed in table 5.1 of chapter 5. The performance of $DFMGP_{GP}$ is also compared to that of DFMGP applying randomly generated DFMs (obtained by the GP initial population creation procedure described in section 8.4 of chapter 8); the latter approach is abbreviated as $DFMGP_{RD2}$.

Table 9.22 shows the results obtained by running $DFMGP_{GP}$, $DFMGP_{RD2}$ and standard GP on the tackled problems. The table shows the best solution quality, b , achieved over 30 runs of GP/DFMGP; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. As in the results presented for the GA approach in section 9.2, the transformation function $1 - (x/(x + 1))$ is applied to all symbolic regression quality scores listed in table 9.22, such that all quality scores in the table span the interval $[0, 1]$, with higher scores indicating better quality.

TABLE 9.22: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Quality Scores

	$DFM-GP_{GP}$	$DFM-GP_{RD2}$	Standard GP							
			OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Sextic	$b = 0.99$ $\mu = 0.94$ $\sigma = 0.03$	$b = 0.97$ $\mu = 0.85$ $\sigma = 0.05$	$b = 1.00$ $\mu = 0.86$ $\sigma = 0.03$	$b = 1.00$ $\mu = 0.84$ $\sigma = 0.04$	$b = 1.00$ $\mu = 0.91$ $\sigma = 0.04$	$b = 0.91$ $\mu = 0.80$ $\sigma = 0.04$	$b = 0.93$ $\mu = 0.85$ $\sigma = 0.06$	$b = 0.90$ $\mu = 0.86$ $\sigma = 0.06$	$b = 0.91$ $\mu = 0.80$ $\sigma = 0.06$	
Keijzer	$b = 0.92$ $\mu = 0.83$ $\sigma = 0.08$	$b = 0.70$ $\mu = 0.58$ $\sigma = 0.11$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.08$	$b = 0.71$ $\mu = 0.59$ $\sigma = 0.09$	$b = 0.76$ $\mu = 0.66$ $\sigma = 0.09$	$b = 0.76$ $\mu = 0.67$ $\sigma = 0.09$	$b = 0.75$ $\mu = 0.67$ $\sigma = 0.09$	$b = 0.73$ $\mu = 0.65$ $\sigma = 0.07$	$b = 0.61$ $\mu = 0.49$ $\sigma = 0.11$	
Par-7	$b = 0.90$ $\mu = 0.78$ $\sigma = 0.06$	$b = 0.73$ $\mu = 0.57$ $\sigma = 0.07$	$b = 0.73$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.62$ $\mu = 0.58$ $\sigma = 0.02$	$b = 0.75$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.59$ $\mu = 0.56$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.66$ $\sigma = 0.03$	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.58$ $\sigma = 0.01$	
Mult-3	$b = 0.98$ $\mu = 0.97$ $\sigma = 0.01$	$b = 0.96$ $\mu = 0.91$ $\sigma = 0.11$	$b = 0.93$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.93$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.92$ $\sigma = 0.01$	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.01$	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.01$	$b = 0.92$ $\mu = 0.88$ $\sigma = 0.02$	$b = 0.74$ $\mu = 0.71$ $\sigma = 0.02$	
Tart	$b = 0.74$ $\mu = 0.67$ $\sigma = 0.03$	$b = 0.60$ $\mu = 0.41$ $\sigma = 0.04$	$b = 0.67$ $\mu = 0.51$ $\sigma = 0.06$	$b = 0.64$ $\mu = 0.52$ $\sigma = 0.09$	$b = 0.66$ $\mu = 0.51$ $\sigma = 0.06$	$b = 0.30$ $\mu = 0.14$ $\sigma = 0.02$	$b = 0.74$ $\mu = 0.57$ $\sigma = 0.06$	$b = 0.65$ $\mu = 0.48$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.14$ $\sigma = 0.02$	$b = 0.69$ $\mu = 0.59$ $\sigma = 0.04$
Dec-tart	$b = 0.79$ $\mu = 0.73$ $\sigma = 0.03$	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.02$	$b = 0.59$ $\mu = 0.49$ $\sigma = 0.04$	$b = 0.76$ $\mu = 0.66$ $\sigma = 0.03$	$b = 0.78$ $\mu = 0.67$ $\sigma = 0.03$	$b = 0.52$ $\mu = 0.46$ $\sigma = 0.02$	$b = 0.65$ $\mu = 0.50$ $\sigma = 0.03$	$b = 0.61$ $\mu = 0.48$ $\sigma = 0.04$	$b = 0.52$ $\mu = 0.46$ $\sigma = 0.02$	$b = 0.68$ $\mu = 0.56$ $\sigma = 0.03$

Table 9.22 highlights the best performing GP approach on each problem. $DFMGP_{GP}$ is seen to achieve near-optimal performance on the sextic, Keijzer, par-7 and mult-3 problems. Importantly, $DFMGP_{GP}$ achieves the best result on all the tackled problems. Statistical tests are conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{RD2}$ and standard GP: the result obtained by running $DFMGP_{GP}$, (μ_0, σ_0) , is compared to that obtained by running $DFMGP_{RD2}$ /standard GP on the same problem (μ_1, σ_1) : here, a pairwise z-test, specified as follows $H_O : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Table 9.23 shows the resulting p-values; here, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted.

Tables 9.22 and 9.23 indicate that $DFMGP_{GP}$ achieves better quality than both $DFMGP_{RD2}$ and standard GP at the 5% level of significance on all problems. Hence $DFMGP_{GP}$ is observed to be more effective than standard GP on all the tackled problems. Furthermore, the GP evolution is shown to be an important component of deriving the DFMs, as $DFMGP_{GP}$ consistently outperforms $DFMGP_{RD2}$. Nevertheless, as in the case with the GA approach, the GP is observed to be a power-hungry approach, executing a colossal number

TABLE 9.23: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-$ GP_{GP} vs. OF	$DFM-$ GP_{GP} vs. BP1	$DFM-$ GP_{GP} vs. BP2	$DFM-$ GP_{GP} vs. FS	$DFM-$ GP_{GP} vs. DSS	$DFM-$ GP_{GP} vs. HP	$DFM-$ GP_{GP} vs. NS1	$DFM-$ GP_{GP} vs. NS2
Sextic	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Keijzer	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Par-7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Mult-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Tart	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Dec-tart	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

of $DFMGP$ runs in the quest to discover an optimal DFM. In this regard, table 9.24 shows the total time taken to train the GP and subsequently run $DFMGP_{GP}$ applying the evolved DFM (this time is abbreviated as GP + $DFMGP_{GP}$); the time is compared to the time taken to run standard GP.

TABLE 9.24: GP + $DFMGP_{GP}$ vs. Standard GP: Execution Time (milliseconds)

	GP + $DFMGP_{GP}$	Standard GP							
		OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Sextic	3.00×10^7	9.98×10^1	1.99×10^3	8.41×10^3	9.78×10^1	9.01×10^1	9.91×10^1	9.40×10^1	
Keijzer	4.90×10^7	1.53×10^3	6.75×10^3	8.34×10^3	1.03×10^3	4.28×10^2	5.18×10^2	3.69×10^3	
Par-7	2.05×10^8	4.07×10^4	6.68×10^4	9.95×10^4	4.19×10^4	3.58×10^3	4.04×10^3	5.78×10^4	
Mult-3	1.98×10^8	2.67×10^4	3.54×10^4	4.01×10^4	2.73×10^4	8.81×10^3	9.21×10^3	3.04×10^4	
Tart	1.62×10^8	3.74×10^4	7.74×10^4	7.94×10^4	3.81×10^4	8.91×10^3	9.02×10^3	6.18×10^4	6.20×10^4
Dec-tart	1.54×10^8	3.66×10^4	7.01×10^4	7.22×10^4	4.01×10^4	9.02×10^3	9.13×10^3	6.84×10^4	6.11×10^4

Similar to the results obtained for the GA approach and $DFMGP_{GA}$ in table 9.3 of section 9.3.1, the key observation in table 9.24 is that the total time taken to train the GP and subsequently execute $DFMGP_{GP}$ is markedly higher than the time taken to run standard GP. This is expected because of the additional computational effort incurred by the GP. As in the case with the GA approach, the GP's computational expense can prove worthwhile if the evolved DFMs generalize to unseen problem instances; in this case, given a problem class, the GP need only be executed *once*, in order to evolve a general problem solver for the class. In this vein, the ensuing sections use training and test sets to verify the reusability of the evolved DFMs.

9.3.2 Testing the reusability of the derived DFMs within problem classes

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$, $DFMGP_{RD2}$ and standard GP on the problem classes listed in table 5.2 of chapter 5.

Table 9.25 shows the results obtained by running $DFMGP_{GP}$ on unseen instances of the sextic problem class; here the GP is trained on the sextic training set defined in table 5.2 of chapter 5; subsequently, $DFMGP_{GP}$ applying the derived DFM is run on the test set; tests 1 to 10 represent test set problems defined in table 5.2. Table 9.25 shows the best solution quality, b , achieved over 30 runs of GP/ $DFMGP$; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. Table 9.25 highlights the GP approaches that achieve the best mean quality scores on each problem.

Table 9.25 shows that $DFMGP_{GP}$ achieves among the best results on all problems. Standard GP with BP2 performs competitively on the second test problem, producing the same average fitness as $DFMGP_{GP}$ over the 30 runs; however, $DFMGP_{GP}$ produces a better best fitness on the problem. As argued in section 9.2.2, BP2's high performance on the sextic problem class is attributed to these problems being inherently modular, and BP being suited to modular problems. Nevertheless, $DFMGP_{GP}$ outperforms BP2 by producing the best results for all the test problems. The justification for $DFMGP_{GP}$'s performance advantage is discussed in section 9.3.4.

TABLE 9.25: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Sextic Class - Test Set Quality Scores

	$DFM-GP_{GP}$	$DFM-GP_{RD2}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.99$ $\mu = 0.94$ $\sigma = 0.04$	$b = 0.89$ $\mu = 0.69$ $\sigma = 0.18$	$b = 0.90$ $\mu = 0.85$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.68$ $\sigma = 0.11$	$b = 0.93$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.89$ $\mu = 0.85$ $\sigma = 0.03$	$b = 0.89$ $\mu = 0.82$ $\sigma = 0.06$	$b = 0.92$ $\mu = 0.86$ $\sigma = 0.04$	$b = 0.90$ $\mu = 0.82$ $\sigma = 0.06$
Test 2	$b = 0.83$ $\mu = 0.68$ $\sigma = 0.10$	$b = 0.73$ $\mu = 0.50$ $\sigma = 0.22$	$b = 0.74$ $\mu = 0.65$ $\sigma = 0.09$	$b = 0.67$ $\mu = 0.56$ $\sigma = 0.07$	$b = 0.80$ $\mu = 0.68$ $\sigma = 0.11$	$b = 0.36$ $\mu = 0.25$ $\sigma = 0.10$	$b = 0.74$ $\mu = 0.63$ $\sigma = 0.13$	$b = 0.72$ $\mu = 0.62$ $\sigma = 0.09$	$b = 0.31$ $\mu = 0.22$ $\sigma = 0.06$
Test 3	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.07$	$b = 0.79$ $\mu = 0.40$ $\sigma = 0.24$	$b = 0.85$ $\mu = 0.81$ $\sigma = 0.06$	$b = 0.84$ $\mu = 0.78$ $\sigma = 0.06$	$b = 0.89$ $\mu = 0.85$ $\sigma = 0.05$	$b = 0.43$ $\mu = 0.30$ $\sigma = 0.07$	$b = 0.89$ $\mu = 0.74$ $\sigma = 0.13$	$b = 0.81$ $\mu = 0.70$ $\sigma = 0.09$	$b = 0.29$ $\mu = 0.20$ $\sigma = 0.05$
Test 4	$b = 0.69$ $\mu = 0.52$ $\sigma = 0.09$	$b = 0.65$ $\mu = 0.29$ $\sigma = 0.18$	$b = 0.60$ $\mu = 0.37$ $\sigma = 0.09$	$b = 0.62$ $\mu = 0.38$ $\sigma = 0.07$	$b = 0.67$ $\mu = 0.42$ $\sigma = 0.08$	$b = 0.29$ $\mu = 0.10$ $\sigma = 0.04$	$b = 0.50$ $\mu = 0.33$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.30$ $\sigma = 0.09$	$b = 0.17$ $\mu = 0.13$ $\sigma = 0.04$
Test 5	$b = 0.70$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.65$ $\mu = 0.23$ $\sigma = 0.22$	$b = 0.52$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.43$ $\mu = 0.27$ $\sigma = 0.06$	$b = 0.52$ $\mu = 0.40$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.10$ $\sigma = 0.07$	$b = 0.40$ $\mu = 0.22$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.20$ $\sigma = 0.07$	$b = 0.09$ $\mu = 0.05$ $\sigma = 0.02$
Test 6	$b = 0.72$ $\mu = 0.46$ $\sigma = 0.09$	$b = 0.59$ $\mu = 0.22$ $\sigma = 0.19$	$b = 0.20$ $\mu = 0.13$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.06$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.07$	$b = 0.14$ $\mu = 0.08$ $\sigma = 0.04$	$b = 0.33$ $\mu = 0.17$ $\sigma = 0.05$	$b = 0.30$ $\mu = 0.17$ $\sigma = 0.05$	$b = 0.04$ $\mu = 0.02$ $\sigma = 0.02$
Test 7	$b = 0.64$ $\mu = 0.52$ $\sigma = 0.11$	$b = 0.61$ $\mu = 0.35$ $\sigma = 0.18$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.06$	$b = 0.54$ $\mu = 0.46$ $\sigma = 0.05$	$b = 0.61$ $\mu = 0.50$ $\sigma = 0.06$	$b = 0.21$ $\mu = 0.11$ $\sigma = 0.07$	$b = 0.59$ $\mu = 0.44$ $\sigma = 0.13$	$b = 0.53$ $\mu = 0.42$ $\sigma = 0.09$	$b = 0.19$ $\mu = 0.11$ $\sigma = 0.05$
Test 8	$b = 0.40$ $\mu = 0.26$ $\sigma = 0.08$	$b = 0.33$ $\mu = 0.15$ $\sigma = 0.21$	$b = 0.24$ $\mu = 0.16$ $\sigma = 0.09$	$b = 0.29$ $\mu = 0.19$ $\sigma = 0.07$	$b = 0.33$ $\mu = 0.20$ $\sigma = 0.09$	$b = 0.20$ $\mu = 0.09$ $\sigma = 0.05$	$b = 0.24$ $\mu = 0.18$ $\sigma = 0.09$	$b = 0.19$ $\mu = 0.11$ $\sigma = 0.09$	$b = 0.13$ $\mu = 0.07$ $\sigma = 0.05$
Test 9	$b = 0.72$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.58$ $\mu = 0.22$ $\sigma = 0.19$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.10$	$b = 0.27$ $\mu = 0.20$ $\sigma = 0.07$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.08$	$b = 0.20$ $\mu = 0.15$ $\sigma = 0.07$	$b = 0.26$ $\mu = 0.13$ $\sigma = 0.09$	$b = 0.26$ $\mu = 0.16$ $\sigma = 0.07$	$b = 0.17$ $\mu = 0.11$ $\sigma = 0.06$
Test 10	$b = 0.90$ $\mu = 0.78$ $\sigma = 0.09$	$b = 0.83$ $\mu = 0.40$ $\sigma = 0.19$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.10$	$b = 0.81$ $\mu = 0.72$ $\sigma = 0.07$	$b = 0.87$ $\mu = 0.77$ $\sigma = 0.08$	$b = 0.55$ $\mu = 0.43$ $\sigma = 0.11$	$b = 0.80$ $\mu = 0.69$ $\sigma = 0.09$	$b = 0.81$ $\mu = 0.69$ $\sigma = 0.09$	$b = 0.41$ $\mu = 0.33$ $\sigma = 0.06$

Statistical tests, identical to those reported in table 9.23, are conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{RD2}$ and standard GP in the sextic class. Table 9.26 shows the resulting p-values; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. The results in table 9.26 ascertain that $DFMGP_{GP}$ largely outperforms both $DFMGP_{RD2}$ and standard GP at the 5% level of significance. Also, standard GP with BP2 achieves on par performance with $DFMGP_{GP}$ on two of the problems.

TABLE 9.26: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Sextic Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-GP_{GP}$ vs. OF	$DFM-GP_{GP}$ vs. BP1	$DFM-GP_{GP}$ vs. BP2	$DFM-GP_{GP}$ vs. FS	$DFM-GP_{GP}$ vs. DSS	$DFM-GP_{GP}$ vs. HP	$DFM-GP_{GP}$ vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.01	0.00	0.27	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.00	0.24	0.00	0.00	0.00	0.00

Table 9.27 shows the results obtained by running DFMGP on unseen instances of the Keijzer problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of chapter 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.27 shows that $DFMGP_{GP}$ achieves among the best results on all problems. Standard GP with HP produces the same average fitness as $DFMGP_{GP}$ on test 10; nevertheless $DFMGP_{GP}$ produces a better best fitness on the problem. Overall, standard GP is shown to achieve varied results with no particular fitness measure demonstrating a consistent performance advantage.

TABLE 9.27: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Keijzer-6 Class - Test Set Quality Scores

	$DFM-GP_{GP}$	$DFM-GP_{RD2}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.07$	$b = 0.55$ $\mu = 0.29$ $\sigma = 0.22$	$b = 0.55$ $\mu = 0.43$ $\sigma = 0.10$	$b = 0.54$ $\mu = 0.43$ $\sigma = 0.09$	$b = 0.57$ $\mu = 0.44$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.32$ $\sigma = 0.20$	$b = 0.60$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.52$ $\mu = 0.41$ $\sigma = 0.08$	$b = 0.28$ $\mu = 0.18$ $\sigma = 0.06$
Test 2	$b = 0.91$ $\mu = 0.83$ $\sigma = 0.06$	$b = 0.92$ $\mu = 0.32$ $\sigma = 0.24$	$b = 0.84$ $\mu = 0.73$ $\sigma = 0.09$	$b = 0.89$ $\mu = 0.76$ $\sigma = 0.10$	$b = 0.82$ $\mu = 0.73$ $\sigma = 0.08$	$b = 0.64$ $\mu = 0.56$ $\sigma = 0.11$	$b = 0.90$ $\mu = 0.75$ $\sigma = 0.07$	$b = 0.86$ $\mu = 0.75$ $\sigma = 0.10$	$b = 0.53$ $\mu = 0.39$ $\sigma = 0.11$
Test 3	$b = 0.82$ $\mu = 0.74$ $\sigma = 0.13$	$b = 0.75$ $\mu = 0.32$ $\sigma = 0.19$	$b = 0.75$ $\mu = 0.63$ $\sigma = 0.07$	$b = 0.72$ $\mu = 0.61$ $\sigma = 0.06$	$b = 0.75$ $\mu = 0.64$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.34$ $\sigma = 0.11$	$b = 0.72$ $\mu = 0.63$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.64$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.17$ $\sigma = 0.05$
Test 4	$b = 0.81$ $\mu = 0.72$ $\sigma = 0.10$	$b = 0.70$ $\mu = 0.42$ $\sigma = 0.30$	$b = 0.71$ $\mu = 0.65$ $\sigma = 0.06$	$b = 0.70$ $\mu = 0.62$ $\sigma = 0.06$	$b = 0.72$ $\mu = 0.65$ $\sigma = 0.05$	$b = 0.44$ $\mu = 0.32$ $\sigma = 0.09$	$b = 0.78$ $\mu = 0.67$ $\sigma = 0.05$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.07$	$b = 0.33$ $\mu = 0.20$ $\sigma = 0.10$
Test 5	$b = 0.65$ $\mu = 0.60$ $\sigma = 0.07$	$b = 0.50$ $\mu = 0.20$ $\sigma = 0.33$	$b = 0.42$ $\mu = 0.31$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.40$ $\sigma = 0.08$	$b = 0.58$ $\mu = 0.44$ $\sigma = 0.13$	$b = 0.20$ $\mu = 0.14$ $\sigma = 0.04$	$b = 0.49$ $\mu = 0.36$ $\sigma = 0.10$	$b = 0.43$ $\mu = 0.31$ $\sigma = 0.11$	$b = 0.29$ $\mu = 0.07$ $\sigma = 0.05$
Test 6	$b = 0.62$ $\mu = 0.49$ $\sigma = 0.07$	$b = 0.50$ $\mu = 0.22$ $\sigma = 0.21$	$b = 0.40$ $\mu = 0.29$ $\sigma = 0.09$	$b = 0.41$ $\mu = 0.34$ $\sigma = 0.08$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.12$ $\mu = 0.09$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.27$ $\sigma = 0.07$	$b = 0.20$ $\mu = 0.10$ $\sigma = 0.11$	$b = 0.07$ $\mu = 0.05$ $\sigma = 0.01$
Test 7	$b = 0.41$ $\mu = 0.28$ $\sigma = 0.08$	$b = 0.21$ $\mu = 0.10$ $\sigma = 0.22$	$b = 0.14$ $\mu = 0.12$ $\sigma = 0.01$	$b = 0.29$ $\mu = 0.18$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.20$ $\sigma = 0.06$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.02$	$b = 0.25$ $\mu = 0.18$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.15$ $\sigma = 0.05$	$b = 0.08$ $\mu = 0.05$ $\sigma = 0.02$
Test 8	$b = 0.30$ $\mu = 0.22$ $\sigma = 0.06$	$b = 0.22$ $\mu = 0.10$ $\sigma = 0.19$	$b = 0.11$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.11$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.16$ $\mu = 0.10$ $\sigma = 0.05$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.18$ $\mu = 0.12$ $\sigma = 0.05$	$b = 0.15$ $\mu = 0.12$ $\sigma = 0.03$	$b = 0.08$ $\mu = 0.04$ $\sigma = 0.02$
Test 9	$b = 0.25$ $\mu = 0.19$ $\sigma = 0.04$	$b = 0.19$ $\mu = 0.09$ $\sigma = 0.13$	$b = 0.16$ $\mu = 0.09$ $\sigma = 0.04$	$b = 0.16$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.18$ $\mu = 0.11$ $\sigma = 0.05$	$b = 0.12$ $\mu = 0.08$ $\sigma = 0.02$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.15$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.08$ $\mu = 0.03$ $\sigma = 0.02$
Test 10	$b = 0.22$ $\mu = 0.12$ $\sigma = 0.09$	$b = 0.18$ $\mu = 0.10$ $\sigma = 0.10$	$b = 0.14$ $\mu = 0.09$ $\sigma = 0.03$	$b = 0.14$ $\mu = 0.10$ $\sigma = 0.02$	$b = 0.18$ $\mu = 0.10$ $\sigma = 0.05$	$b = 0.10$ $\mu = 0.07$ $\sigma = 0.02$	$b = 0.14$ $\mu = 0.10$ $\sigma = 0.03$	$b = 0.16$ $\mu = 0.12$ $\sigma = 0.03$	$b = 0.05$ $\mu = 0.03$ $\sigma = 0.01$

Table 9.28 shows the result of statistical tests conducted to ascertain the performance advantage of $DFM-GP_{GP}$ over $DFMGP_{RD2}$ and standard GP in the Keijzer class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.28 indicates that $DFMGP_{GP}$ largely outperforms both $DFMGP_{RD2}$ and standard GP at the 5% level of significance on the problems.

Table 9.29 shows the results obtained by running DFMGP on unseen instances of the par-N problem class: tests 1 to 5 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.29 shows that $DFMGP_{GP}$ achieves the best result on all problems. Standard GP with BP2 also performs competitively in the class: this is attributed to BP exploiting the inherent modularity of the even-N parity problems [1]. Nevertheless, $DFMGP_{GP}$ consistently outperforms standard GP with BP2.

TABLE 9.28: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Keijzer-6 Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-$ GP_{GP} vs. OF	$DFM-$ GP_{GP} vs. BP1	$DFM-$ GP_{GP} vs. BP2	$DFM-$ GP_{GP} vs. FS	$DFM-$ GP_{GP} vs. DSS	$DFM-$ GP_{GP} vs. HP	$DFM-$ GP_{GP} vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00

TABLE 9.29: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Even-N Parity Class - Test Set Quality Scores

	$DFM-$ GP_{GP}	$DFM-$ GP_{RD2}	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.68$ $\mu = 0.63$ $\sigma = 0.05$	$b = 0.57$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.02$	$b = 0.59$ $\mu = 0.55$ $\sigma = 0.04$	$b = 0.51$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.55$ $\mu = 0.54$ $\sigma = 0.01$	$b = 0.51$ $\mu = 0.50$ $\sigma = 0.00$
Test 2	$b = 0.65$ $\mu = 0.60$ $\sigma = 0.02$	$b = 0.55$ $\mu = 0.50$ $\sigma = 0.06$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.53$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 3	$b = 0.60$ $\mu = 0.56$ $\sigma = 0.03$	$b = 0.57$ $\mu = 0.50$ $\sigma = 0.04$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 4	$b = 0.60$ $\mu = 0.56$ $\sigma = 0.02$	$b = 0.53$ $\mu = 0.50$ $\sigma = 0.03$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.53$ $\mu = 0.51$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 5	$b = 0.56$ $\mu = 0.52$ $\sigma = 0.02$	$b = 0.52$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$

Table 9.30 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{RD2}$ and standard GP in the par-N class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. The results in table 9.30 show that $DFMGP_{GP}$ outperforms both $DFMGP_{RD1}$ and standard GP at the 5% level of significance on all the tackled problems.

TABLE 9.30: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Even-N Parity Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-$ GP_{GP} vs. OF	$DFM-$ GP_{GP} vs. BP1	$DFM-$ GP_{GP} vs. BP2	$DFM-$ GP_{GP} vs. FS	$DFM-$ GP_{GP} vs. DSS	$DFM-$ GP_{GP} vs. HP	$DFM-$ GP_{GP} vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 9.31 shows the results obtained by running DFMGP on unseen instances of the mult-N problem class: tests 1 to 5 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approach is highlighted in the table. Table 9.31 shows that $DFMGP_{GP}$ achieves the best result on all problems. Conversely, standard GP achieves varied results with no particular fitness measure demonstrating a consistent performance advantage.

Table 9.32 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{RD2}$ and standard GP in the multi-N class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.32 shows that $DFMGP_{GP}$ outperforms both $DFMGP_{RD2}$ and standard GP at the 5% level of significance on all the tackled problems.

TABLE 9.31: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: N-bit Multiplier Class - Test Set Quality scores

	$DFM-GP_{GP}$	$DFM-GP_{RD2}$	Standard GP						
			OF	BP1	BP2	FS	DSS	HP	NS1
Test 1	$b = 0.86$ $\mu = 0.83$ $\sigma = 0.01$	$b = 0.80$ $\mu = 0.60$ $\sigma = 0.09$	$b = 0.75$ $\mu = 0.74$ $\sigma = 0.01$	$b = 0.77$ $\mu = 0.75$ $\sigma = 0.01$	$b = 0.78$ $\mu = 0.75$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.71$ $\sigma = 0.01$	$b = 0.74$ $\mu = 0.73$ $\sigma = 0.01$	$b = 0.74$ $\mu = 0.72$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.57$ $\sigma = 0.01$
Test 2	$b = 0.84$ $\mu = 0.80$ $\sigma = 0.02$	$b = 0.70$ $\mu = 0.55$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.72$ $\mu = 0.70$ $\sigma = 0.02$	$b = 0.72$ $\mu = 0.70$ $\sigma = 0.01$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.02$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.02$	$b = 0.60$ $\mu = 0.54$ $\sigma = 0.06$	$b = 0.69$ $\mu = 0.67$ $\sigma = 0.01$
Test 3	$b = 0.78$ $\mu = 0.76$ $\sigma = 0.02$	$b = 0.72$ $\mu = 0.50$ $\sigma = 0.03$	$b = 0.68$ $\mu = 0.66$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.71$ $\mu = 0.69$ $\sigma = 0.01$	$b = 0.68$ $\mu = 0.66$ $\sigma = 0.01$	$b = 0.73$ $\mu = 0.70$ $\sigma = 0.01$	$b = 0.60$ $\mu = 0.52$ $\sigma = 0.04$	$b = 0.62$ $\mu = 0.60$ $\sigma = 0.01$
Test 4	$b = 0.64$ $\mu = 0.54$ $\sigma = 0.02$	$b = 0.56$ $\mu = 0.50$ $\sigma = 0.02$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.52$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$
Test 5	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.02$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$	$b = 0.50$ $\mu = 0.50$ $\sigma = 0.00$

TABLE 9.32: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: N-bit Multiplier Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-GP_{GP}$ vs. OF	$DFM-GP_{GP}$ vs. BP1	$DFM-GP_{GP}$ vs. BP2	$DFM-GP_{GP}$ vs. FS	$DFM-GP_{GP}$ vs. DSS	$DFM-GP_{GP}$ vs. HP	$DFM-GP_{GP}$ vs. NS1
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 9.33 shows the results obtained by running DFMGP on unseen instances of the tartarus problem class: tests 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approach is highlighted in the table. Table 9.33 shows that $DFMGP_{GP}$ achieves among the best results on all problems. Standard GP with NS2 achieves the same average fitness as $DFMGP_{GP}$ on test 8; however, $DFMGP_{GP}$ achieves a better best fitness on the problem. Standard GP with NS2 achieves competitive performance on the problems in the class due to mitigating against deception [4]. Nevertheless, table 9.33 shows that $DFMGP_{GP}$ largely outperforms standard GP by consistently producing the best result on all the test instances.

Table 9.34 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{RD2}$ and standard GP in the tartarus class; in the table, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. The p-values shown in table 9.34 show that standard GP with NS2 performs on par with $DFMGP_{GP}$ on test 8. Nevertheless, $DFMGP_{GP}$ largely outperforms both $DFMGP_{RD2}$ and standard GP at the 5% level of significance on the tackled problems.

Table 9.35 shows the results obtained by running DFMGP on unseen instances of the deceptive tartarus problem class: test 1 to 10 represent the test set problems defined in table 5.2 of section 5. The best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs are shown. The best performing GP approaches are highlighted in the table. Table 9.35 shows that $DFMGP_{GP}$ achieves the

TABLE 9.33: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Tartarus Class - Test Set Quality Scores

	$DFM-GP_{GP}$	$DFM-GP_{RD2}$	Standard GP							
			OF	BP1	BP2	FS	DSS	HP	NS1	NS2
Test 1	$b = 0.58$ $\mu = 0.55$ $\sigma = 0.02$	$b = 0.47$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.37$ $\sigma = 0.05$	$b = 0.44$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.36$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.23$ $\sigma = 0.01$	$b = 0.42$ $\mu = 0.36$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.06$	$b = 0.24$ $\mu = 0.23$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.02$
Test 2	$b = 0.51$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.40$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.25$ $\sigma = 0.05$	$b = 0.36$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.36$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.19$ $\mu = 0.18$ $\sigma = 0.02$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.04$	$b = 0.36$ $\mu = 0.27$ $\sigma = 0.04$	$b = 0.19$ $\mu = 0.18$ $\sigma = 0.01$	$b = 0.39$ $\mu = 0.36$ $\sigma = 0.01$
Test 3	$b = 0.77$ $\mu = 0.74$ $\sigma = 0.04$	$b = 0.69$ $\mu = 0.29$ $\sigma = 0.05$	$b = 0.67$ $\mu = 0.52$ $\sigma = 0.11$	$b = 0.63$ $\mu = 0.52$ $\sigma = 0.08$	$b = 0.60$ $\mu = 0.52$ $\sigma = 0.07$	$b = 0.09$ $\mu = 0.06$ $\sigma = 0.02$	$b = 0.64$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.65$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.11$ $\mu = 0.07$ $\sigma = 0.02$	$b = 0.63$ $\mu = 0.57$ $\sigma = 0.05$
Test 4	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.03$	$b = 0.54$ $\mu = 0.32$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.35$ $\sigma = 0.08$	$b = 0.44$ $\mu = 0.37$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.25$ $\sigma = 0.02$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.37$ $\sigma = 0.04$	$b = 0.26$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.54$ $\mu = 0.50$ $\sigma = 0.02$
Test 5	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.06$	$b = 0.62$ $\mu = 0.33$ $\sigma = 0.05$	$b = 0.53$ $\mu = 0.46$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.50$ $\mu = 0.44$ $\sigma = 0.05$	$b = 0.33$ $\mu = 0.29$ $\sigma = 0.04$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.04$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.03$	$b = 0.35$ $\mu = 0.28$ $\sigma = 0.03$	$b = 0.59$ $\mu = 0.50$ $\sigma = 0.05$
Test 6	$b = 0.64$ $\mu = 0.56$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.32$ $\sigma = 0.08$	$b = 0.44$ $\mu = 0.35$ $\sigma = 0.06$	$b = 0.44$ $\mu = 0.35$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.25$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.04$	$b = 0.37$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.29$ $\mu = 0.29$ $\sigma = 0.00$	$b = 0.38$ $\mu = 0.35$ $\sigma = 0.05$
Test 7	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.06$	$b = 0.44$ $\mu = 0.32$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.34$ $\mu = 0.29$ $\sigma = 0.06$	$b = 0.18$ $\mu = 0.17$ $\sigma = 0.01$	$b = 0.44$ $\mu = 0.38$ $\sigma = 0.04$	$b = 0.37$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.22$ $\mu = 0.21$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.35$ $\sigma = 0.05$
Test 8	$b = 0.52$ $\mu = 0.44$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.27$ $\sigma = 0.05$	$b = 0.40$ $\mu = 0.33$ $\sigma = 0.07$	$b = 0.43$ $\mu = 0.34$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.39$ $\sigma = 0.05$	$b = 0.26$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.39$ $\sigma = 0.04$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.44$ $\sigma = 0.05$
Test 9	$b = 0.57$ $\mu = 0.49$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.25$ $\sigma = 0.05$	$b = 0.45$ $\mu = 0.30$ $\sigma = 0.05$	$b = 0.46$ $\mu = 0.37$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.35$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.22$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.04$	$b = 0.40$ $\mu = 0.35$ $\sigma = 0.05$	$b = 0.26$ $\mu = 0.25$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.05$
Test 10	$b = 0.56$ $\mu = 0.48$ $\sigma = 0.06$	$b = 0.52$ $\mu = 0.28$ $\sigma = 0.05$	$b = 0.43$ $\mu = 0.29$ $\sigma = 0.06$	$b = 0.46$ $\mu = 0.40$ $\sigma = 0.05$	$b = 0.50$ $\mu = 0.42$ $\sigma = 0.05$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.06$	$b = 0.42$ $\mu = 0.32$ $\sigma = 0.07$	$b = 0.25$ $\mu = 0.24$ $\sigma = 0.01$	$b = 0.49$ $\mu = 0.44$ $\sigma = 0.06$

TABLE 9.34: $DFMGP_{GP}$ vs. $DFMGP_{RD2}$ /Standard GP: Tartarus Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{RD2}$	$DFM-GP_{GP}$ vs. OF	$DFM-GP_{GP}$ vs. BP1	$DFM-GP_{GP}$ vs. BP2	$DFM-GP_{GP}$ vs. FS	$DFM-GP_{GP}$ vs. DSS	$DFM-GP_{GP}$ vs. HP	$DFM-GP_{GP}$ vs. NS1	$DFM-GP_{GP}$ vs. NS2
Test 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39
Test 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

best results on all problems. As in the results shown for $DFMGP_{GA}$ in section 9.2.2, standard GP with BP2 and standard GP with NS2 perform competitively in the class. As argued in section 9.2.2, BP2's performance advantage is attributed to the fitness measure mitigating against the loss of useful subtrees, which would otherwise be discarded by OF measures. Furthermore, NS2's performance advantage shown in table 9.35 is attributed to the task-specific behavior descriptors mitigating against deception on the deceptive tasks. Nevertheless, table 9.35 shows that $DFMGP_{GP}$ largely outperforms standard GP by consistently producing the best result on all the test instances.

As in the case with the results obtained for the GA approach in section 9.2.2, the results in this section show that the DFMs evolved by the GP approach are reusable within the problem classes: the DFMs solve unseen problem instances to optimality. Furthermore, the importance of using the GP to train the DFMs is ascertained by the fact that DFMGP applying randomly generated DFMs does not fair as well as DFMGP applying the evolved DFMs.

9.3.3 Testing the reusability of the derived DFMs on real world problems

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$, $DFMGP_{RD2}$ and standard GP on the real-world problems listed in table 5.3 of chapter 5.

Table 9.37 shows the results obtained on the real-world problems. Similar to the results presented in section 9.2.3, for the regression problems in the table, $DFMGP_{P1}$ represents $DFMGP_{GP}$ employing the DFM trained on the sextic training set, $DFMGP_{P2}$ represents $DFMGP_{GP}$ employing the DFM trained on the Keijzer training set, and $DFMGP_{UN}$ represents $DFMGP_{GP}$ employing the DFM trained on the combined sextic and Keijzer training sets. For the Boolean problems in the table, $DFMGP_{P1}$ represents $DFMGP_{GP}$ employing the DFM trained on the par-N training set, $DFMGP_{P2}$ represents $DFMGP_{GP}$ employing the DFM trained on the mult-N training set, and $DFMGP_{UN}$ represents $DFMGP_{GP}$ employing the DFM trained on the combined par-N and mult-N training sets. Table 9.37 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 GP/DFMGP runs. The best performing GP approach is highlighted in the table.

TABLE 9.37: $DFMGP_{GP}$ vs. Standard GP: Real World Quality Scores

	$DFM-GP_{P1}$	$DFM-GP_{P2}$	$DFM-GP_{UN}$	Standard GP						
				OF	BP1	BP2	FS	DSS	HP	NS1
Dow	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.47$ $\mu = 0.43$ $\sigma = 0.02$	$b = 0.51$ $\mu = 0.47$ $\sigma = 0.03$	$b = 0.43$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.37$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.37$ $\sigma = 0.02$	$b = 0.41$ $\mu = 0.36$ $\sigma = 0.02$	$b = 0.43$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.41$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.36$ $\mu = 0.34$ $\sigma = 0.01$
Abalone	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.02$	$b = 0.18$ $\mu = 0.16$ $\sigma = 0.01$	$b = 0.19$ $\mu = 0.17$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.14$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.14$ $\sigma = 0.01$	$b = 0.13$ $\mu = 0.12$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.12$ $\mu = 0.12$ $\sigma = 0.00$
Sensor	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.93$ $\sigma = 0.03$	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.03$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.99$ $\mu = 0.93$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.95$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.89$ $\sigma = 0.02$	$b = 0.84$ $\mu = 0.81$ $\sigma = 0.01$
Decoder	$b = 0.96$ $\mu = 0.94$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.94$ $\sigma = 0.02$	$b = 0.92$ $\mu = 0.88$ $\sigma = 0.02$	$b = 0.94$ $\mu = 0.89$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.90$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.95$ $\sigma = 0.01$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.90$ $\sigma = 0.02$	$b = 0.80$ $\mu = 0.76$ $\sigma = 0.02$

As in the results obtained for the GA approach in section 9.2.3, table 9.37 shows that one of the DFMGP approaches achieves the best result on each of the tackled problems. $DFMGP_{UN}$ achieves the best result on the Dow and abalone problems, while $DFMGP_{P2}$ achieves the best result on the sensor and decoder problems. Statistical tests are conducted to ascertain the significance of the differences observed. For each DFMGP approach, the result obtained on each problem, (μ_0, σ_0) , is compared to the same for the other DFMGP approaches or standard GP, (μ_1, σ_1) : whereby, a pairwise z-test, specified as follows $H_O : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Tables 9.38, 9.39 and 9.40 show the resulting p-values for $DFMGP_{P1}$, $DFMGP_{P2}$ and $DFMGP_{UN}$ respectively; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted.

Tables 9.38, 9.39 and 9.40 show that $DFMGP_{P2}$ consistently outperforms standard GP and the other DFMGP approaches on the sensor and decoder problems; on the other hand, $DFMGP_{P1}$ and $DFMGP_{UN}$ do not perform reliably on the problems. The tables also show that $DFMGP_{UN}$ largely outperforms standard GP and the other DFMGP approaches on the Dow and abalone problems. $DFMGP_{P2}$ performs on par with $DFMGP_{UN}$ on the Abalone problem. Nevertheless, $DFMGP_{P1}$ and $DFMGP_{P2}$ do not perform as reliably as $DFMGP_{UN}$ on the Dow and Abalone problems. As in the case with the results obtained for the GA approach

TABLE 9.38: $DFMGP_{P1}$ vs. $DFMGP_{P2}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{P1}$ vs. $DFMGP_{P2}$	$DFMGP_{P1}$ vs. $DFMGP_{UN}$	$DFM-$ GP_{P1} vs. OF	$DFM-$ GP_{P1} vs. BP1	$DFM-$ GP_{P1} vs. BP2	$DFM-$ GP_{P1} vs. FS	$DFM-$ GP_{P1} vs. DSS	$DFM-$ GP_{P1} vs. HP	$DFM-$ GP_{P1} vs. NS1
Dow	0.00	0.00	0.14	0.39	0.19	0.00	0.16	0.28	0.00
Abalone	0.00	0.00	0.22	0.14	0.10	0.00	0.10	0.11	0.00
Sensor	0.00	0.29	0.00	0.00	0.19	0.00	0.00	0.00	0.00
Decoder	0.00	0.34	0.00	0.00	0.00	0.02	0.03	0.03	0.00

TABLE 9.39: $DFMGP_{P2}$ vs. $DFMGP_{P1}/DFMGP_{UN}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{P2}$ vs. $DFMGP_{P1}$	$DFMGP_{P2}$ vs. $DFMGP_{UN}$	$DFM-$ GP_{P2} vs. OF	$DFM-$ GP_{P2} vs. BP1	$DFM-$ GP_{P2} vs. BP2	$DFM-$ GP_{P2} vs. FS	$DFM-$ GP_{P2} vs. DSS	$DFM-$ GP_{P2} vs. HP	$DFM-$ GP_{P2} vs. NS1
Dow	0.00	0.00	0.36	0.41	0.34	0.00	0.33	0.13	0.00
Abalone	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Decoder	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLE 9.40: $DFMGP_{UN}$ vs. $DFMGP_{P1}/DFMGP_{P2}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{UN}$ vs. $DFMGP_{P1}$	$DFMGP_{UN}$ vs. $DFMGP_{P2}$	$DFM-$ GP_{UN} vs. OF	$DFM-$ GP_{UN} vs. BP1	$DFM-$ GP_{UN} vs. BP2	$DFM-$ GP_{UN} vs. FS	$DFM-$ GP_{UN} vs. DSS	$DFM-$ GP_{UN} vs. HP	$DFM-$ GP_{UN} vs. NS1
Dow	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Abalone	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sensor	0.32	0.00	0.03	0.03	0.10	0.00	0.00	0.00	0.00
Decoder	0.36	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00

in section 9.2.3, the inference gained from the data is that DFMGP has the capacity to outperform standard GP. However, it is difficult to a priori ascertain the particular DFMGP approach that will achieve superior performance. In this regard, it would be useful if simple heuristics exist that can be used to a priori detect the properties of the real-world problems, such that training sets with the same properties can be used to deduce suitable DFMs for DFMGP.

9.3.4 Analysis of the derived DFMs

This section presents an analysis of the derived DFMs to identify the fitness measures that suit the different phases of search for different problems.

Table 9.41 lists the DFMs evolved by the GP for the problem classes defined in table 5.2 of chapter 5. Each DFM displayed is the best DFM found over the 30 runs of the GP approach on the given training problems; while there are elements of the GP population with similar fitness over the 30 runs, there are no other elements with the same fitness. For illustrative purposes, the DFMs in table 9.41 are expressed in their simplest form (i.e. each parse-tree branch is evaluated to express it in the simplest arithmetic/logical form).

TABLE 9.41: Evolved DFMs - Problem Classes

Class	Evolved DFM
Sextic	<pre> graph TD IFGLT --> 24 IFGLT --> G IFGLT --> B </pre>
Keijzer	<pre> graph TD IFGLT --> 34 IFGLT --> IFGLT2[IFGLT] IFGLT --> IFGGT IFGLT2 --> 25 IFGLT2 --> D IFGLT2 --> plus[+] IFGGT --> 75 IFGGT --> F IFGGT --> B plus --> E plus --> x x --> C x --> C </pre>
Par-N	<pre> graph TD IFGLT --> 18 IFGLT --> G IFGLT --> IFGLT2[IFGLT] IFGLT2 --> 70 IFGLT2 --> plus[+] IFGLT2 --> C plus --> C plus --> E </pre>
Multi-N	<pre> graph TD IFGLT --> 58 IFGLT --> D IFGLT --> IFGLT2[IFGLT] IFGLT2 --> 89 IFGLT2 --> C IFGLT2 --> plus[+] plus --> B plus --> E </pre>
Tart.	<pre> graph TD IFGLT --> 65 IFGLT --> H IFGLT --> IFGLT2[IFGLT] IFGLT2 --> 75 IFGLT2 --> B IFGLT2 --> IFGLT3[IFGLT] IFGLT3 --> 82 IFGLT3 --> plus[+] IFGLT3 --> B plus --> H plus --> B </pre>

Table 9.41 shows that a number of the DFMs arithmetically combine the fitness measures. In the literature,

TABLE 9.41: Evolved DFMs - Problem Classes (contd.)

Class	Evolved DFM
Dec. tart.	

fitness measures are arithmetically combined in order to simultaneously optimize different objectives. For example, in [66] (and in this study), the behavioral fitness of a solution program is calculated as a combination of the OF score and two terms that quantify the usefulness of the solution's subprograms. Another example of arithmetically combined fitness measures is novelty-fitness aggregation [72], where the fitness measure used is a weighted sum of the OF and novelty scores. As a further example, in parsimony studies [267], the OF score is combined with a parsimony penalty to mitigate bloat. The literature reports varied results with respect to arithmetically combining fitness measures. For instance, the efforts in [66] (and in this study - see chapter 4) prove successful, whereby combining the OF and subprogram usefulness scores improves on the performance of OF-GP on modular problems. However, in [72], novelty-fitness aggregation does not achieve performance gain over OF-GP; this may be due to the NS behavior descriptor used in [72]. In [267], combining the OF score with a parsimony penalty enforces parsimony at the expense of degrading the performance of GP, whereby GP is shown to achieve lower quality scores. In the current study, GP is used to search the space of fitness measures combinations for DFMGP. This is advantageous, because direct feedback can be obtained of the particular fitness measure combinations that are producing the desired results; furthermore, the combinations are optimized with the progress of the GP search. Therefore, the expectation is that the best-fitness DFMs found by the GP approach contain optimized combinations of the fitness measures.

As in the case with the GA-evolved DFMs in table 9.20 of section 9.2.4, the DFMs shown in table 9.41 select explorative fitness measures for the preliminary DFMGP generations; subsequently the more exploitative fitness measures are selected in later DFMGP generations. NS1 (encoded as G) is selected for the preliminary generations in the sextic and par-N classes. NS2 (encoded as H) is selected for the preliminary generations in the tart. and dec-tart. classes. As argued in section 9.2.4, NS supports exploration by explicitly rewarding diversification from prior behaviors; NS in fact inhibits exploitation, because the fitness measure maintains the same level of selective pressure in favor of diversification, even when near-optimal solutions are found. Hence NS works better when confined to the initial generations of an evolutionary algorithm, preceded by a more exploitative fitness measure in later generations [204]. FS (encoded as D), selected for the preliminary generations in the Keijzer and mult-N classes, is also an explorative measure. As argued in section 9.2.4, FS supports exploration by maintaining the diversity of the GP population, permitting GP to explore new niches (or optima); FS inhibits exploitation and convergence in later generations, because the fitness measure enforces the maintenance of niches at all GP generations. Hence, like NS, FS works better when confined to the initial generations, and preceded by a more exploitative fitness measure in later generations [6, 15].

Similar to the results shown in section 9.2.4, table 9.41 also shows that BP1 and BP2 (encoded as B and C

respectively) are selected for the middle and/or final DFMGP generations in all classes. As argued in section 9.2.4, BP promotes the retention of useful subtrees, facilitating their exploitation (or refinement). Whereas useful subtrees can exist in the randomly generated initial population for trivial problems, more complex problems may require that further exploration is conducted in the preliminary GP generations, as a precursor to BP: thus, useful subtrees can be made available for BP to exploit in later generations.

Overall, DFMGP applying the shift from explorative to exploitative fitness measures shown in table 9.41 is expected to outperform standard GP, because the current fitness measure is selected to support the on-going phase of GP. The ensuing discussion analyzes the DFMs in more detail.

Table 9.41 shows that best-fitness DFM evolved for the sextic class does not arithmetically combine the employed fitness measures. This means that the GP evolution did not find a better DFM that arithmetically combines the fitness measures; this may be due to premature convergence at the meta-level, or lack of existence of a better combination of the given fitness measures. Section 9.4.4 will compare this DFM to the GA-evolved DFM for the same problem class; the comparison shows that the GP-evolved DFM is similar to the GA-evolved DFM. Hence the result obtained for the sextic class is important, because it shows that the GP approach is capable of defaulting to a DFM that resembles the GA-evolved DFM.

Table 9.41 also shows the DFM evolved for the Keijzer class. While section 9.4.4 will show that this DFM is similar to the GA-evolved DFM for the same class, the key difference between the two is that the DFM in table 9.41 arithmetically combines DSS (encoded as E) and BP2 (encoded as C) in the middle DFMGP generations. Section 3.5.1 of chapter 3 established that DSS supports exploration by maintaining the population diversity. Hence the combination of DSS and BP2 is a mix of explorative and exploitative measures. According to the literature [266], the success of an evolutionary algorithm depends on the ability to maintain the delicate balance between exploration and exploitation. Therefore, rather than simply switching between purely explorative and exploitative measures, it may be useful to combine the different types of measures, whereby the fitness measure directs exploration and exploitation to occur simultaneously when required.

A similar trend is seen with the DFMs evolved for the par-N, mult-N, tart and dec-tart classes: the DFMs contain arithmetic combinations of explorative and exploitative measures. The DFM for the par-N class contains a combination of DSS and BP2. In turn, the DFM for the mult-N class contains a combination of DSS and BP1. The DFMs for the tart class contains a combination of NS2 and BP1. Lastly, the DFM for the dec-tart class contains a combination of NS, OF and BP1. The implication is that the DFMs drive exploration and exploitation to occur simultaneously (or in parallel) when this is required in DFMGP.

Table 9.42 shows the DFM evolved from the combined training set of the sextic and Keijzer classes; the table also shows the same for the combined training set of the par-N and mult-N classes.

The DFMs in table 9.42 are similar to those shown in table 9.41, whereby the explorative fitness measures FS and NS (encoded as D and G respectively) are selected in the preliminary DFMGP generations. Furthermore, the more exploitative BP1 and BP2 (encoded as B and C respectively) are selected for the middle and later DFMGP generations. The DFMs in table 9.42 also contain arithmetic combinations of explorative and exploitative measures. The DFM evolved for the regression problems contains a combination of the explorative DSS (encoded as E) and the exploitative BP1 and BP2. In turn, the DFM evolved for the Boolean problems contains a combination of the explorative NS1 (encoded as G) and the exploitative BP2. The DFMs also combine two explorative measures, FS and NS1, in the preliminary DFMGP generations. A possible reason for this is that FS is the preferred fitness measure for the preliminary generations in the Keijzer class, while NS1 is preferred for the preliminary generations in the sextic class; therefore, when presented with a training set that combines the sextic and Keijzer classes, the GP evolving the DFMs combines FS and NS1 in order to accommodate the different problems in the training set. A similar pattern is seen with combining FS and NS1 on the Boolean training set. Hence, in this case, the arithmetic combinations seem to increase the generality of the evolved DFMs, accommodating the different classes of problems in the training set.

In summary, tables 9.41 and 9.42 show that the GP approach used to evolve the DFMs for DFMGP has the

TABLE 9.42: Evolved DFMs - Combined training sets

Class	Evolved DFM
Sextic + Keijzer	
Par-N + Mult-N	

ability to select the fitness measures that suit the particular on-going phase of GP. For the problem classes tackled, explorative fitness measures are selected for the initial DFMGP generations, while the more exploitative fitness measures are selected in later DFMGP generations. The GP approach also has the ability to arithmetically combine the fitness measures in the evolved DFMs. This means that rather than simply switching between purely explorative and exploitative phases in $DFMGP_{GP}$, the fitness measures can be combined to direct the two processes to occur simultaneously when required.

9.4 Comparison of $GA/DFMGP_{GA}$ and $GP/DFMGP_{GP}$

This section presents a comparison of the GA and GP approaches results. Section 9.4.1 compares the effectiveness of the $GA/DFMGP_{GA}$ approach with that of the $GP/DFMGP_{GP}$ approach. Subsequently, section 9.4.2 compares the reusability of the derived DFMs within problem classes. Next, section 9.4.3 compares the reusability of the derived DFMs on real world problems. Finally, section 9.4.4 compares the derived DFMs.

9.4.1 Comparing the effectiveness of $DFMGP_{GP}$ with $DFMGP_{GA}$

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$ with that of $DFMGP_{GA}$ on the benchmark problems listed in table 5.1 of chapter 5.

Table 9.43 shows a comparison of the results obtained by running $DFMGP_{GP}$ and $DFMGP_{GA}$ on the tackled problems. Essentially, table 9.43 presents a side-by-side comparison of the results shown in table 9.22 of section 9.3.1 for $DFMGP_{GP}$, and in table 9.1 of section 9.2.1 for $DFMGP_{GA}$. The table shows the best solution quality, b , achieved over 30 runs of $DFMGP_{GP}/DFMGP_{GA}$; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. Table 9.43 highlights the DFMGP approach that achieves the best mean quality scores on each problem.

TABLE 9.43: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Sextic	$b = 0.99$ $\mu = 0.94$ $\sigma = 0.03$	$b = 0.98$ $\mu = 0.94$ $\sigma = 0.03$
Keijzer	$b = 0.92$ $\mu = 0.83$ $\sigma = 0.08$	$b = 0.87$ $\mu = 0.77$ $\sigma = 0.08$
Par-7	$b = 0.90$ $\mu = 0.78$ $\sigma = 0.06$	$b = 0.84$ $\mu = 0.73$ $\sigma = 0.05$
Mult-3	$b = 0.98$ $\mu = 0.97$ $\sigma = 0.01$	$b = 0.99$ $\mu = 0.97$ $\sigma = 0.01$
Tart	$b = 0.72$ $\mu = 0.67$ $\sigma = 0.03$	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.04$
Dec-tart	$b = 0.79$ $\mu = 0.73$ $\sigma = 0.03$	$b = 0.74$ $\mu = 0.69$ $\sigma = 0.02$

Table 9.43 shows that both $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve near-optimal performance on the sextic and mult-3 problems. Furthermore, $DFMGP_{GP}$ achieves the best result on the Keijzer, par-7, tart and dec-tart problems. Statistical tests are conducted to ascertain the significance of the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$: the result obtained by running $DFMGP_{GP}$, (μ_0, σ_0) , is compared to that obtained by running $DFMGP_{GA}$ on the same problem (μ_1, σ_1) : here, a pairwise z-test, specified as follows $H_0 : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Table 9.44 shows the resulting p-values, and highlights the values that indicate statistical significance (at $\alpha = 0.05$).

TABLE 9.44: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Sextic	0.39
Keijzer	0.03
Par-7	0.01
Mult-3	0.35
Tart	0.03
Dec-tart	0.04

Table 9.44 ascertains that $DFMGP_{GP}$ significantly outperforms $DFMGP_{GA}$ on the Keijzer, par-7, tart and dec-tart problems. The key difference between $DFMGP_{GP}$ and $DFMGP_{GA}$ is the representation used for the dynamic fitness measures. Tables 9.41 and 9.42 of section 9.3.4 showed that the DFMs used in $DFMGP_{GP}$ are abstract syntax trees that facilitate arithmetic combinations of the fitness measures on the DFMGP generations. On the other hand, tables 9.20 and 9.21 of section 9.2.4 showed that the DFMs used in $DFMGP_{GA}$ are fitness measure sequences employing a restrictive representation, which dictates that only a single fitness measure can be applied individually on each DFMGP generation. When evolving the DFMs, the GP approach adapts the fitness measures to be combined into the DFMs, and also adapts the structure and size of the DFMs. Conversely, evolution of the DFMs in the GA approach is restricted to adapting fixed-length DFMs. Hence the GP approach offers more flexibility in terms of adapting the DFMs to suit the given problem (or problem class). Section 9.4.4 compares the structure of the DFMs used in $DFMGP_{GP}$ with that of the DFMs used in $DFMGP_{GA}$, and discusses the reasons for the associated performance gains in more detail.

Table 9.45 shows the total time taken to train the GP and subsequently run $DFMGP_{GP}$ applying the evolved DFM (this time is abbreviated as GP + $DFMGP_{GP}$); the time is compared to the total time taken to

train the GA and subsequently run $DFMGP_{GA}$ applying the evolved DFM (this time is abbreviated as GA + $DFMGP_{GA}$).

TABLE 9.45: GP + $DFMGP_{GP}$ vs. GA + $DFMGP_{GA}$: Execution Time (milliseconds)

	GP + $DFMGP_{GP}$	GA + $DFMGP_{GA}$
Sextic	3.00×10^7	2.30×10^7
Keijzer	4.90×10^7	4.25×10^7
Par-7	2.05×10^8	1.90×10^8
Mult-3	1.98×10^8	1.69×10^8
Tart	1.62×10^8	1.41×10^8
Dec-tart	1.54×10^8	1.17×10^8

Table 9.45 shows that the time taken to train the GP and subsequently execute $DFMGP_{GP}$ is slightly higher than the time taken to train the GA and subsequently execute $DFMGP_{GA}$. This is due to bloat occurring while training the GP, which is a direct consequence of the variable-length representation used in the GP. Nevertheless, sections 9.2.2, 9.3.2, 9.2.3 and 9.3.3 showed that the DFMs evolved by the GA and GP approaches are reusable on problem classes, and also on real-world problems. Hence the time taken to train the GA and GP approaches proves worthwhile, because this need only be done *once* for the problems in a given class.

9.4.2 Comparing the reusability of the derived DFMs within problem classes

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$ with that of $DFMGP_{GA}$ on the problem classes listed in table 5.2 of chapter 5. The ensuing results show that $DFMGP_{GP}$ generally outperforms $DFMGP_{GA}$ on the problem classes. The sextic class presents an anomaly, whereby $DFMGP_{GP}$ performs on par with $DFMGP_{GA}$. Section 9.4.4 will show that in the sextic class, the GP approach generates a DFM that is similar to the GA-evolved DFM. Here, the GP does not find a better DFM; this may be due to premature convergence at the meta-level, or lack of existence of a better combination of the given fitness measures. Section 9.4.4 discusses this in more detail. Overall, this section shows that $DFMGP_{GP}$ outperforms $DFMGP_{GA}$, and at the least, the DFMGP approaches are seen to achieve on par performance.

Table 9.46 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the sextic problem class. Essentially, table 9.46 presents a side-by-side comparison of the results shown in table 9.25 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.4 of section 9.2.2 for $DFMGP_{GA}$. The table shows the best solution quality, b , achieved over 30 runs of GP/DFMGP; the table also shows the mean, μ , and standard deviation, σ , of the best solution quality over the 30 runs. The best performing GP approaches are highlighted in the table.

Statistical tests, identical to those reported in table 9.44, are conducted to ascertain the significance of the quantitative differences shown in table 9.46. Table 9.47 shows the resulting p-values; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. Table 9.47 shows that $DFMGP_{GP}$ achieves on par performance with $DFMGP_{GA}$ on all problems. Therefore, $DFMGP_{GP}$ does not outperform $DFMGP_{GA}$ in the sextic class.

Table 9.48 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the Keijzer problem class. The table presents a side-by-side comparison of the results shown in table 9.27 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.6 of section 9.2.2 for $DFMGP_{GA}$. Table 9.48 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

Table 9.48 shows that $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve the same average solution quality on test 10. However, $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the Keijzer class by consistently producing the best result on all the test instances.

TABLE 9.46: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Sextic Class - Test Set Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.99$ $\mu = 0.94$ $\sigma = 0.04$	$b = 0.98$ $\mu = 0.94$ $\sigma = 0.03$
Test 2	$b = 0.83$ $\mu = 0.68$ $\sigma = 0.10$	$b = 0.80$ $\mu = 0.68$ $\sigma = 0.09$
Test 3	$b = 0.97$ $\mu = 0.90$ $\sigma = 0.07$	$b = 0.99$ $\mu = 0.90$ $\sigma = 0.08$
Test 4	$b = 0.69$ $\mu = 0.52$ $\sigma = 0.09$	$b = 0.69$ $\mu = 0.52$ $\sigma = 0.09$
Test 5	$b = 0.70$ $\mu = 0.50$ $\sigma = 0.08$	$b = 0.72$ $\mu = 0.50$ $\sigma = 0.08$
Test 6	$b = 0.72$ $\mu = 0.46$ $\sigma = 0.09$	$b = 0.70$ $\mu = 0.46$ $\sigma = 0.09$
Test 7	$b = 0.64$ $\mu = 0.52$ $\sigma = 0.11$	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.09$
Test 8	$b = 0.40$ $\mu = 0.26$ $\sigma = 0.08$	$b = 0.39$ $\mu = 0.25$ $\sigma = 0.08$
Test 9	$b = 0.72$ $\mu = 0.47$ $\sigma = 0.07$	$b = 0.69$ $\mu = 0.47$ $\sigma = 0.07$
Test 10	$b = 0.90$ $\mu = 0.78$ $\sigma = 0.09$	$b = 0.90$ $\mu = 0.77$ $\sigma = 0.09$

TABLE 9.47: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Sextic Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.48
Test 2	0.44
Test 3	0.44
Test 4	0.49
Test 5	0.44
Test 6	0.44
Test 7	0.32
Test 8	0.41
Test 9	0.44
Test 10	0.41

Table 9.49 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ in the Keijzer class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.49 indicates that $DFMGP_{GP}$ largely outperforms both $DFMGP_{GA}$ at the 5% level of significance on the problems.

Table 9.50 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the par-N problem class. The table presents a side-by-side comparison of the results shown in table 9.29 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.8 of section 9.2.2 for $DFMGP_{GA}$. Table 9.50 shows the best, b , mean, μ ,

TABLE 9.48: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Keijzer-6 Class - Test Set Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.07$	$b = 0.62$ $\mu = 0.50$ $\sigma = 0.08$
Test 2	$b = 0.91$ $\mu = 0.83$ $\sigma = 0.06$	$b = 0.89$ $\mu = 0.76$ $\sigma = 0.08$
Test 3	$b = 0.82$ $\mu = 0.74$ $\sigma = 0.13$	$b = 0.81$ $\mu = 0.69$ $\sigma = 0.11$
Test 4	$b = 0.81$ $\mu = 0.72$ $\sigma = 0.10$	$b = 0.79$ $\mu = 0.67$ $\sigma = 0.09$
Test 5	$b = 0.65$ $\mu = 0.60$ $\sigma = 0.07$	$b = 0.62$ $\mu = 0.51$ $\sigma = 0.07$
Test 6	$b = 0.62$ $\mu = 0.49$ $\sigma = 0.07$	$b = 0.59$ $\mu = 0.47$ $\sigma = 0.07$
Test 7	$b = 0.41$ $\mu = 0.28$ $\sigma = 0.08$	$b = 0.41$ $\mu = 0.26$ $\sigma = 0.09$
Test 8	$b = 0.30$ $\mu = 0.22$ $\sigma = 0.06$	$b = 0.27$ $\mu = 0.18$ $\sigma = 0.05$
Test 9	$b = 0.25$ $\mu = 0.19$ $\sigma = 0.04$	$b = 0.22$ $\mu = 0.14$ $\sigma = 0.06$
Test 10	$b = 0.22$ $\mu = 0.12$ $\sigma = 0.09$	$b = 0.20$ $\mu = 0.12$ $\sigma = 0.09$

TABLE 9.49: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Keijzer Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.00
Test 2	0.00
Test 3	0.00
Test 4	0.00
Test 5	0.00
Test 6	0.18
Test 7	0.17
Test 8	0.00
Test 9	0.00
Test 10	0.45

and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

Table 9.50 shows that $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve the same average solution quality on test 5. However, $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the par-N class by consistently producing the best result on all the test instances.

Table 9.51 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ in the par-N class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are

TABLE 9.50: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Even-N Parity Class - Test Set Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.68$ $\mu = 0.63$ $\sigma = 0.05$	$b = 0.65$ $\mu = 0.59$ $\sigma = 0.04$
Test 2	$b = 0.65$ $\mu = 0.60$ $\sigma = 0.02$	$b = 0.59$ $\mu = 0.54$ $\sigma = 0.02$
Test 3	$b = 0.62$ $\mu = 0.56$ $\sigma = 0.03$	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.03$
Test 4	$b = 0.60$ $\mu = 0.56$ $\sigma = 0.02$	$b = 0.55$ $\mu = 0.52$ $\sigma = 0.01$
Test 5	$b = 0.56$ $\mu = 0.52$ $\sigma = 0.02$	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$

highlighted in the table. The results in table 9.51 show that $DFMGP_{GP}$ largely outperforms both $DFMGP_{GA}$ and standard GP at the 5% level of significance on the tackled problems.

TABLE 9.51: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Even-N Parity Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.00
Test 2	0.00
Test 3	0.03
Test 4	0.00
Test 5	0.34

Table 9.52 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the mult-N problem class. The table presents a side-by-side comparison of the results shown in table 9.31 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.10 of section 9.2.2 for $DFMGP_{GA}$. Table 9.52 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

Table 9.52 shows that $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the Keijzer class by producing the best result on most of the test instances. However, $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve the same average solution quality on tests 4 and 5. Modelling digital multipliers is a difficult task for evolutionary techniques, especially when the number of bits in the multiplicands is greater than three [234]; hence a possible reason for $DFMGP_{GP}$'s lack of performance gain on tests 4 and 5 is that $DFMGP$ has reached a performance threshold.

Table 9.53 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ in the mult-N class; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted in the table. Table 9.53 shows that $DFMGP_{GP}$ outperforms $DFMGP_{GA}$ and standard GP at the 5% level of significance on tests 1, 2 and 3. On the other hand, $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve on par performance on tests 4 and 5.

Table 9.54 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the tartarus problem class. The table presents a side-by-side comparison of the results shown in table 9.33 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.12 of section 9.2.2 for $DFMGP_{GA}$. Table 9.54 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

TABLE 9.52: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: N-bit Multiplier Class - Test Set Quality scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.86$ $\mu = 0.83$ $\sigma = 0.01$	$b = 0.82$ $\mu = 0.79$ $\sigma = 0.01$
Test 2	$b = 0.84$ $\mu = 0.80$ $\sigma = 0.02$	$b = 0.79$ $\mu = 0.77$ $\sigma = 0.01$
Test 3	$b = 0.78$ $\mu = 0.76$ $\sigma = 0.02$	$b = 0.76$ $\mu = 0.73$ $\sigma = 0.01$
Test 4	$b = 0.64$ $\mu = 0.54$ $\sigma = 0.02$	$b = 0.59$ $\mu = 0.54$ $\sigma = 0.01$
Test 5	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.02$	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$

TABLE 9.53: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: N-bit Multiplier Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.00
Test 2	0.00
Test 3	0.00
Test 4	0.33
Test 5	0.49

Table 9.54 shows that $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve the same average solution quality on tests 8 and 10. Nevertheless, $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the tartarus class by consistently producing the best result on all the test instances.

Table 9.55 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ in the tartarus class; in the table, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. The p-values shown in table 9.55 show that $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ at the 5% level of significance on the tackled problems.

Table 9.56 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on unseen instances of the deceptive tartarus problem class. The table presents a side-by-side comparison of the results shown in table 9.35 of section 9.3.2 for $DFMGP_{GP}$, and in table 9.14 of section 9.2.2 for $DFMGP_{GA}$. Table 9.56 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

Table 9.56 shows that $DFMGP_{GP}$ and $DFMGP_{GA}$ achieve the same average solution quality on tests 6 and 10. Nevertheless, $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the deceptive tartarus class by consistently producing the best result on all the test instances.

Table 9.57 shows the result of statistical tests conducted to ascertain the performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ in the deceptive tartarus class; in the table, the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. The p-values shown in table 9.57 show that $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ at the 5% level of significance on the tackled problems.

Overall, the results in this section show that $DFMGP_{GP}$ largely outperforms $DFMGP_{GA}$ on the tackled problem classes. $DFMGP_{GP}$'s performance gain is attributed to the fact that the DFMs evolved for $DFMGP_{GP}$ incorporate arithmetic combinations of the fitness measures. Therefore, in $DFMGP_{GP}$, rather than simply switching between purely explorative and exploitative measures, combinations of the explorative and exploitative measures direct the two processes to occur simultaneously when required.

TABLE 9.54: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Tartarus Class - Test Set Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.58$ $\mu = 0.55$ $\sigma = 0.02$	$b = 0.53$ $\mu = 0.51$ $\sigma = 0.01$
Test 2	$b = 0.51$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.45$ $\mu = 0.43$ $\sigma = 0.01$
Test 3	$b = 0.77$ $\mu = 0.74$ $\sigma = 0.04$	$b = 0.73$ $\mu = 0.69$ $\sigma = 0.05$
Test 4	$b = 0.58$ $\mu = 0.54$ $\sigma = 0.03$	$b = 0.57$ $\mu = 0.53$ $\sigma = 0.03$
Test 5	$b = 0.69$ $\mu = 0.62$ $\sigma = 0.06$	$b = 0.64$ $\mu = 0.55$ $\sigma = 0.06$
Test 6	$b = 0.64$ $\mu = 0.56$ $\sigma = 0.06$	$b = 0.62$ $\mu = 0.50$ $\sigma = 0.06$
Test 7	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.06$	$b = 0.48$ $\mu = 0.38$ $\sigma = 0.06$
Test 8	$b = 0.52$ $\mu = 0.44$ $\sigma = 0.06$	$b = 0.52$ $\mu = 0.44$ $\sigma = 0.06$
Test 9	$b = 0.57$ $\mu = 0.49$ $\sigma = 0.06$	$b = 0.54$ $\mu = 0.45$ $\sigma = 0.06$
Test 10	$b = 0.56$ $\mu = 0.48$ $\sigma = 0.06$	$b = 0.54$ $\mu = 0.48$ $\sigma = 0.06$

TABLE 9.55: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Tartarus Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.00
Test 2	0.00
Test 3	0.00
Test 4	0.40
Test 5	0.00
Test 6	0.00
Test 7	0.00
Test 8	0.49
Test 9	0.00
Test 10	0.38

TABLE 9.56: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Deceptive Tartarus Class - Test Set Quality Scores

	$DFMGP_{GP}$	$DFMGP_{GA}$
Test 1	$b = 0.54$ $\mu = 0.52$ $\sigma = 0.01$	$b = 0.50$ $\mu = 0.48$ $\sigma = 0.01$
Test 2	$b = 0.54$ $\mu = 0.50$ $\sigma = 0.01$	$b = 0.49$ $\mu = 0.46$ $\sigma = 0.01$
Test 3	$b = 0.79$ $\mu = 0.76$ $\sigma = 0.01$	$b = 0.75$ $\mu = 0.72$ $\sigma = 0.01$
Test 4	$b = 0.50$ $\mu = 0.47$ $\sigma = 0.01$	$b = 0.46$ $\mu = 0.43$ $\sigma = 0.01$
Test 5	$b = 0.76$ $\mu = 0.72$ $\sigma = 0.02$	$b = 0.73$ $\mu = 0.69$ $\sigma = 0.02$
Test 6	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.03$
Test 7	$b = 0.60$ $\mu = 0.53$ $\sigma = 0.03$	$b = 0.60$ $\mu = 0.48$ $\sigma = 0.03$
Test 8	$b = 0.54$ $\mu = 0.48$ $\sigma = 0.04$	$b = 0.51$ $\mu = 0.45$ $\sigma = 0.03$
Test 9	$b = 0.53$ $\mu = 0.48$ $\sigma = 0.03$	$b = 0.49$ $\mu = 0.45$ $\sigma = 0.03$
Test 10	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.04$	$b = 0.48$ $\mu = 0.43$ $\sigma = 0.04$

TABLE 9.57: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Deceptive Tartarus Class - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Test 1	0.00
Test 2	0.00
Test 3	0.00
Test 4	0.00
Test 5	0.00
Test 6	0.50
Test 7	0.00
Test 8	0.00
Test 9	0.00
Test 10	0.48

9.4.3 Comparing the reusability of the derived DFMs on real world problems

This section presents the results obtained from comparing the performance of $DFMGP_{GP}$ with that of $DFMGP_{GA}$ on the real-world problems listed in table 5.3 of chapter 5.

Table 9.58 shows the results obtained by running $DFMGP_{GP}/DFMGP_{GA}$ on the real-world problems. The table presents a side-by-side comparison of the results shown in table 9.37 of section 9.3.3 for $DFMGP_{GP}$, and in table 9.16 of section 9.2.3 for $DFMGP_{GA}$. For the regression problems in the table, $DFMGP_{GP}(P1)$ and $DFMGP_{GA}(P1)$ indicate DFMGP approaches trained on the sextic training set. Similarly, the suffix ($P2$) indicates DFMGP approaches trained on the Keijzer training set. Lastly, the suffix (UN) indicates DFMGP approaches trained on the combined sextic and Keijzer training sets. In a similar fashion, for the Boolean problems in the table, the suffixes ($P1$), ($P2$) and (UN) indicate DFMGP approaches trained on the par-N, mult-N, and combined par-N and mult-N training sets respectively. Table 9.58 shows the best, b , mean, μ , and standard deviation, σ , of the best solution quality achieved over 30 $DFMGP_{GP}/DFMGP_{GA}$ runs. The best performing GP approaches are highlighted in the table.

TABLE 9.58: $DFMGP_{GP}$ vs. $DFMGP_{GA}$: Real World Quality Scores

	$DFMGP_{GP}$ ($P1$)	$DFMGP_{GP}$ ($P2$)	$DFMGP_{GP}$ (UN)	$DFMGP_{GA}$ ($P1$)	$DFMGP_{GA}$ ($P2$)	$DFMGP_{GA}$ (UN)
Dow	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.47$ $\mu = 0.43$ $\sigma = 0.02$	$b = 0.51$ $\mu = 0.47$ $\sigma = 0.03$	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.44$ $\mu = 0.39$ $\sigma = 0.02$	$b = 0.48$ $\mu = 0.42$ $\sigma = 0.03$
Abalone	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.02$	$b = 0.18$ $\mu = 0.16$ $\sigma = 0.01$	$b = 0.19$ $\mu = 0.17$ $\sigma = 0.01$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.02$	$b = 0.14$ $\mu = 0.13$ $\sigma = 0.01$	$b = 0.17$ $\mu = 0.15$ $\sigma = 0.01$
Sensor	$b = 0.95$ $\mu = 0.93$ $\sigma = 0.03$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.96$ $\mu = 0.93$ $\sigma = 0.03$	$b = 0.95$ $\mu = 0.90$ $\sigma = 0.03$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.95$ $\mu = 0.90$ $\sigma = 0.03$
Decoder	$b = 0.96$ $\mu = 0.94$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.97$ $\mu = 0.94$ $\sigma = 0.02$	$b = 0.95$ $\mu = 0.92$ $\sigma = 0.02$	$b = 1.00$ $\mu = 0.99$ $\sigma = 0.02$	$b = 0.95$ $\mu = 0.91$ $\sigma = 0.02$

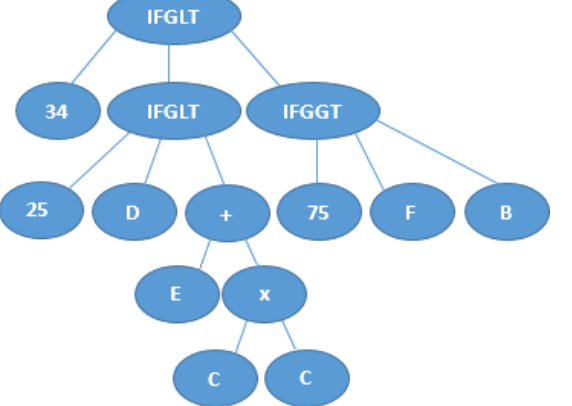
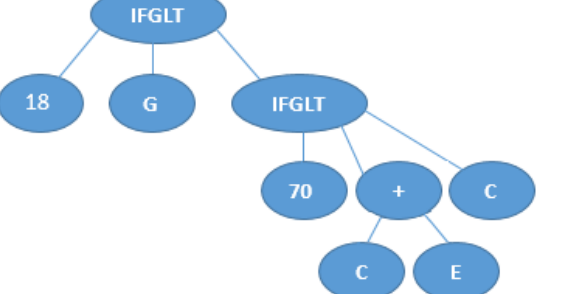
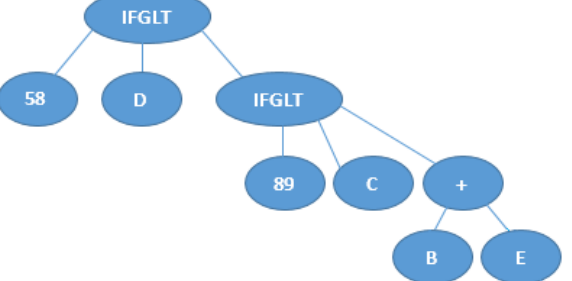
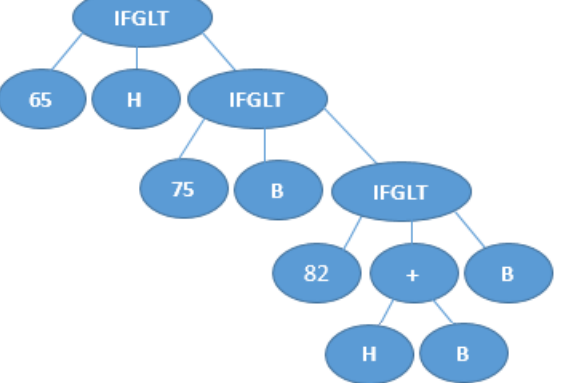
Table 9.58 shows that $DFMGP_{GP}$ demonstrates a performance advantage consistent with the previous section. $DFMGP_{GP}(UN)$ achieves the best result on the regression problems. Furthermore, both $DFMGP_{GP}(P2)$ and $DFMGP_{GA}(P2)$ achieve near-optimal performance on the Boolean problems.

Statistical tests are conducted to ascertain the significance of the quantitative differences shown in table 9.58. On each problem, the result obtained by running the best performing $DFMGP_{GP}$ approach, (μ_0, σ_0) , is compared to the result obtained by running the best performing $DFMGP_{GA}$ approach (μ_1, σ_1) : here, a pairwise z-test, specified as follows $H_O : \mu_0 = \mu_1, H_A : \mu_0 > \mu_1$, is conducted. Table 9.59 shows the resulting p-values; the p-values that indicate statistical significance (at $\alpha = 0.05$) are highlighted. Table 9.59 indicates that the best $DFMGP_{GP}$ approach significantly outperforms $DFMGP_{GA}$ on the regression problems. In turn, the best $DFMGP_{GP}$ and $DFMGP_{GA}$ approaches achieve near-optimal, and hence on par performance on the Boolean problems.

TABLE 9.59: $DFMGP_{GP}$ vs. $DFMGP_{GA}$ /Standard GP: Real-World Problems - Statistical Tests

	$DFMGP_{GP}$ vs. $DFMGP_{GA}$
Dow	0.01
Abalone	0.00
Sensor	0.50
Decoder	0.50

TABLE 9.60: Evolved DFMs - Problem Classes (contd.)

Class	GA approach: Best DFM	GP approach: Best DFM
Keijzer	<p>DDDDDDDDDDDDDDDDDDDDDDDDDDDDDD 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 EEAAAAACCCBBBBBBBBBBBBBBBBBB 2526272829303132333435363738394041424344454647484950 BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB 51525354555657585960616263646566676869707172737475 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 767778798081828384858687888990919293949596979899100</p>	
Par-N	<p>GGGGGGGGGGGGGGGGGGGGGGGGGGGGGG 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 CCCCCCGGGGGGGGGGGGGGGGGGGGG 26272829303132333435363738394041424344454647484950 EEEEEEEEEEEEEEEEEEEEEEEEEEGGGGG 51525354555657585960616263646566676869707172737475 CCCCCCCCCCCCCCCCCCCCCCEEFE E 767778798081828384858687888990919293949596979899100</p>	
Mult-N	<p>DDDDDDDDDDDDDDDDDDDDDDDDDDDDDD 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 DDDDDDDDDDDDDDDDDDDDDDDDDDDDD 26272829303132333435363738394041424344454647484950 BDDDDDDCCAAACCCCCDDCCCCC 51525354555657585960616263646566676869707172737475 CCEEECCCCCCCCCEEEEBBBBC A 767778798081828384858687888990919293949596979899100</p>	
Tart.	<p>HHHHHHHHHHHHHHHHHHHHHHHHHHHHHH 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 HAHHEHHHHHHHHHHHHHHHHHHHHHHHH 25262728293031323334353637383940414243444546474849 HHHHHHHHHHHHHHHHHHHHHHHHHHHHH 5051525354555657585960616263646566676869707172737475 AAAAABBBBBBBBBBBBBBBBBBBBBBB 767778798081828384858687888990919293949596979899100</p>	

9.5 Summary

This chapter presented the results of the two developed approaches, the GA and GP approaches for deriving DFMs for DFMGP. Both DFMGP applying the GA-evolved DFMs ($DFMGP_{GA}$) and DFMGP applying the GP-evolved DFMs ($DFMGP_{GP}$) were found to be effective on benchmark problems from different problem domains, and reliably outperformed standard GP applying the fitness measures individually on the tackled problems. Furthermore, the derived DFMs were shown to be reusable on problem classes, whereby both $DFMGP_{GA}$ and $DFMGP_{GP}$ reliably outperformed standard GP on unseen problem instances from the same problem class used to derive the DFMs; this was shown to be true for problem classes from different problem domains. In addition, the derived DFMs were also shown to be reusable on real-world problems, whereby both $DFMGP_{GA}$ and $DFMGP_{GP}$ reliably outperformed standard GP on unseen real-world problems from the same problem domain as the training problems used to derive the DFMs. Overall, the benefit that $DFMGP_{GA}$ and $DFMGP_{GP}$ have over standard GP is that using a different fitness measure or combination of fitness measures on different generations contributes to obtaining the required balance between exploration and exploitation needed by the algorithm. A shortcoming of $DFMGP_{GA}$ and $DFMGP_{GP}$ is that the computational effort is higher than standard GP. This can be expected as two populations are evolved simultaneously, one for the higher (or meta-) level algorithm evolving the DFMs and a second for the GP algorithm solving the problem. However, given that evolved DFMs are reusable the benefit outweighs this shortcoming.

The performance of $DFMGP_{GP}$ was also compared to that of $DFMGP_{GA}$. The results showed that $DFMGP_{GP}$ is more effective than $DFMGP_{GA}$, whereby the former approach reliably outperformed the latter on benchmark problems from different problem domains. $DFMGP_{GP}$ also outperformed $DFMGP_{GA}$ on unseen problem instances from the problem classes. Furthermore, $DFMGP_{GP}$ outperformed $DFMGP_{GA}$ on unseen real-world problems. The advantage that $DFMGP_{GP}$ has over $DFMGP_{GA}$ is that $DFMGP_{GP}$ allows for different fitness measures to be combined into a measure to be applied on a generation, while $DFMGP_{GA}$ applies a single fitness measure per generation. Furthermore, the variable representation used by $DFMGP_{GP}$ allows greater flexibility for a fitness measure specific to the particular class of problems to be induced.

Chapter 10

Conclusions and Future Work

10.1 Introduction

This chapter presents the overall conclusions based on the research findings of the thesis. Section 10.2 presents the outcomes of the research with respect to the six objectives outlined in chapter 1. Subsequently, section 10.3 provides a discussion revisiting the contributions made by the thesis, mentioned in 1. Finally, section 10.4 provides directions for future work building upon this thesis. Lastly, section 10.5 summarizes the chapter.

10.2 Objectives and Conclusions

- Apply GAs for evolving DFMs for DFMGP.

A genetic algorithm (GA) was implemented to evolve DFMs for DFMGP. Each GA chromosome encoded a candidate sequence of fitness measures to be applied over the generations of DFMGP. The GA was used to select the best fitness measure to employ on each generation of DFMGP for the given problem (or set of problems). The fitness measures made available for the selection improve on different aspects of GP; the fitness measures were also found to suit different problems in the experiment conducted in chapter 4; furthermore, a critical analysis of the literature anticipated that some of the fitness measures are better suited to specific phases of the GP search. The goal in providing a diverse subset of fitness measures for the DFMs was for the GA approach to be able to produce optimal DFMs for varied problems and problem classes.

DFMGP applying the GA-evolved DFMs was shown to achieve near-optimal performance on a number of benchmark problems. Furthermore, the GA evolution was found to be an important component of deriving the DFMs, as DFMGP applying the evolved DFMs reliably outperformed DFMGP applying randomly generated DFMs on a number of benchmark problems from different problem domains.

- Apply GP for evolving DFMs for DFMGP.

A genetic programming algorithm (GP) was also implemented to evolve DFMs for DFMGP. Each GP chromosome encoded a candidate arithmetic and logical combination of fitness measures. The GP was used to select the best combination of fitness measures to employ on each generation of DFMGP for the given problem (or set of problems). The fitness measures made available for the selection were the same as used for the GA approach, whereby the goal was for the GP approach to be able to produce optimal DFMs for varied problems and problem classes.

As in the case with the GA approach, DFMGP applying the GP-evolved DFMs achieved near-optimal performance on a number of benchmark problems. The GP evolution was also found to be an important component of deriving the DFMs, as DFMGP applying the evolved DFMs reliably outperformed DFMGP applying randomly generated DFMs on a number of benchmark problems from different problem domains.

- Compare the performance of DFMGP with the conventional GP approach.

The performance of DFMGP applying the GA-evolved DFMs ($DFMGP_{GA}$) and DFMGP applying the GP-evolved DFMs ($DFMGP_{GP}$) was compared to that of standard GP applying each of the fitness measures individually on benchmark problems from different problem domains. Both $DFMGP_{GP}$ and $DFMGP_{GA}$ outperformed standard GP at the 5% level of significance on the majority of the tackled problems, with a few exceptions seen, where standard GP performed on par with DFMGP. An analysis of the DFMs used in DFMGP showed that both the GA and GP approaches used to evolve the DFMs selected fitness measures that support exploration for the preliminary DFMGP generations, and fitness measures that support exploitation for later DFMGP generations. Exploration promotes coverage of the search space and is more suited to the preliminary GP generations, whereas exploitation is used to refine promising solutions when good points in the search space have been discovered and is more suited to later GP generations [22]. The performance advantage of DFMGP over standard GP is therefore justified by the premise that the fitness measures used on the DFMGP generations support the more suitable search in the on-going phase of GP.

- Compare the performance of GAs and GP in evolving DFMs.

The performance of $DFMGP_{GA}$ was also compared to that of $DFMGP_{GP}$ on benchmark problems from different problem domains. $DFMGP_{GP}$ was shown to outperform $DFMGP_{GA}$ at the 5% level of significance on the majority of the tackled problems, with a few exceptions seen, where $DFMGP_{GP}$ performed on par with $DFMGP_{GA}$. The key differences between $DFMGP_{GP}$ and $DFMGP_{GA}$ are 1) the meta-algorithm used to derive the DFMs a priori (i.e. GP versus GA), and 2) the representation used for the DFMs. When GP is used to derive the DFMs, the candidate DFMs are variable-length abstract syntax trees that encode combinations of the fitness measures. The GP approach adapts the fitness measures to be combined into the DFMs, and also adapts the structure and size of the DFMs. Conversely, when GA is used to derive the DFMs, the candidate DFMs are fixed-length fitness measure sequences. Hence evolution of the DFMs in the GA approach is restricted to adapting fixed-length fitness measure sequences. Importantly, the key advantage of the variable-length representation used in the GP approach is that it allows the DFMs to grow in complexity to suit the given problem (or problem class).

An analysis of the evolved DFMs showed that the DFMs used in $DFMGP_{GP}$ arithmetically combine explorative and exploitative fitness measures for use on some of the DFMGP generations. According to the literature [266], the success of an evolutionary algorithm depends on the ability to maintain the delicate balance between exploration and exploitation. The performance advantage of $DFMGP_{GP}$ over $DFMGP_{GA}$ is justified by the fact that in the former approach, rather than simply switching between purely explorative and exploitative measures, the combinations of the explorative and exploitative measures direct the two processes to occur simultaneously when required in DFMGP.

- Assess the reusability of the evolved DFMs.

Two aspects were involved in achieving this objective. The first aspect was to assess the reusability of the evolved DFMs for problem classes. A set of problem classes from different problem domains was identified for this assessment. For each problem class, both the GA and GP approaches were trained on a subset of problem instances from the class, the training set; subsequently the performances of $DFMGP_{GP}$ and $DFMGP_{GA}$ applying the DFMs evolved for the given problem class were compared to that of standard GP applying each of the fitness measures individually on unseen problem instances from the class, the test set. Here, a derived DFM was said to be reusable if DFMGP applying the DFM performed better the standard GP approach on the test instances. The results showed that both $DFMGP_{GP}$ and $DFMGP_{GA}$ outperformed standard GP at the 5% level of significance on the majority of the test instances across the different problem classes. Hence the evolved DFMs were shown to be reusable for problem classes. This is an important result, considering that both the GA and GP approaches exert considerable computational effort to evolve the DFMs. The implications of reusability on the problem classes are that the total time necessary for the derivations is reduced because the DFMs need only be evolved *once* for a given problem class.

The second aspect of this objective was to assess the reusability of the evolved DFMs for complex real-world problems. A set of real-world problems from different problem domains was identified for this assessment. After the DFMs were evolved for the problem classes, both $DFMGP_{GP}$ and $DFMGP_{GA}$ applying the corresponding DFM were executed on unseen real-world problems, whereby for each problem class, the unseen problems tested came from the same problem domain as the problems in the class. In addition, more general DFMs were evolved for each problem domain by training the GA and GP approaches on the combined set of all the training problems from the problem classes in the domain; $DFMGP_{GP}$ and $DFMGP_{GA}$ applying these DFMs were also executed on the real-world problems. The results showed that for both $DFMGP_{GP}$ and $DFMGP_{GA}$, at least one of the DFMs induced a DFMGP that outperformed standard GP at the 5% level of significance on the tackled problems. Hence DFMGP has the capacity to outperform standard GP on unseen real-world problems. Nevertheless, it is difficult to a priori ascertain the particular DFM that will achieve this performance gain on the real-world problems. The performance of DFMGP on unseen problems has to do with how similar the unseen problems are compared to the training problems used to derive the DFMs with respect to problem properties addressed by the different fitness measures e.g. local optima, deceptiveness, bloat. Therefore, definitive training sets can only be obtained if prior knowledge exists of shared problem properties. Ultimately, it would be useful if simple heuristics exist that can be used to a priori detect the properties of the real-world problems. Subsequently, training sets with the same properties can be used to deduce suitable DFMs for DFMGP.

- Analyse the best performing DFMs to identify the fitness measures that are most useful in the different phases of search for different problems.

The best DFMs found by the GA and GP approach were analyzed. The analysis found that different DFMs were evolved for different problem classes. By virtue of Wolpert and Macready's [14] No Free Lunch (NFL) theorems, there is no universally optimal DFM; rather, different DFMs were evolved for the different classes, because different fitness measures suit the different classes.

The analysis of the DFMs also found that both the GA- and GP-evolved DFMs employed a shift from explorative to exploitative fitness measures in the course of DFMGP. The fitness measures that support exploration were selected for the preliminary DFMGP generations, whereas the more exploitative fitness measures were selected for the middle and later DFMGP generations. Therefore, in the evolved DFMs, the fitness measure(s) selected on each generation supported the on-going phase of GP search. This pattern was observed to be consistent for DFMs evolved for different problems and problem classes.

The best DFMs found by the GP approach were also found to arithmetically combine explorative and exploitative fitness measures on some of the DFMGP generations. These combinations of explorative and exploitative measures directed the two processes to occur simultaneously in DFMGP when required. The DFMs containing the arithmetic combinations were associated with performance gains, whereby $DFMGP_{GP}$ applying these DFMs outperformed $DFMGP_{GA}$.

Overall, the analysis of the DFMs showed that the benefit that $DFMGP_{GP}$ and $DFMGP_{GA}$ have over standard GP is that using a different fitness measure or combination of fitness measures on different generations contributes to obtaining the required balance between exploration and exploitation needed by the algorithm.

10.3 Contributions

This thesis has made the following contributions

- A detailed survey of the fitness measures prescribed for GP was conducted in chapter 3. To the author's knowledge, such an extensive survey of the different fitness measures has not previously been conducted. The fitness measures were also categorized based on their *modi operandi*, hence highlighting

the different types of fitness prescribed in the literature, namely objective fitness, divide-and-conquer fitness, fitness sharing, dynamic fitness, subjective fitness and novelty search. Importantly, the fitness measures were surveyed in the same format to facilitate their comparison: motivation, implementation, variants, advantages, disadvantages and discussion.

- A comparison of state-of-the-art fitness measures was conducted in chapter 4. There is currently a lack of comparison between the different fitness measures proposed in the literature: most studies simply compare against an OF measure, otherwise state-of-the-art fitness measures are not compared. The aim of the comparison conducted in chapter 4 was to evaluate the effect that the different fitness measures have for different problems. The study looked at how the different fitness measures address each of the limitations they aim to overcome for different problems. An all-round comparison of the fitness measures was conducted, whereby the performance of the fitness measures was assessed based on different criteria, namely the solution quality achieved, generalization capability, population diversity maintained, structural complexity of the evolving population, as well as the time taken to run GP applying the given fitness measure. Furthermore, the fitness measures were compared on a number of problems from different problem domains. Overall, the comparison verified the NFL theorems: no one fitness measure achieved the best result on all tackled problems, rather the performance of the fitness measures was shown to depend on the properties of the specific problem being tackled.
- Dynamic Fitness Measure Genetic Programming (DFMGP), the use of different fitness measures on the different GP generations, was proposed in chapter 5. DFMGP is justified by the preliminary work done in chapter 4, where different fitness measures are shown to suit different problems. Furthermore, a study conducted by McKay [6, 15] anticipates that applying different fitness measures throughout the problem solving process should be more effective than applying a single fitness measure individually throughout the algorithm; McKay [6, 15] anticipates this idea, but does not conduct a thorough investigation to this effect. In the current study, the term dynamic fitness measure (DFM) is used to refer to the alternation of fitness measures on the DFMGP generations. A high-level search algorithm is needed to search the space of DFMs for DFMGP, because selecting the best fitness measure to apply on each GP generation is in itself a combinatorial optimization problem, due to a combinatorial explosion of possible permutations of the fitness measures on the GP generations. Both GA and GP were proposed to search the space of DFMs for DFMGP.

Following the proposal of DFMGP in the current study, an experiment was conducted to evaluate the effectiveness of the approach in chapter 9. The study shows that DFMGP is more effective than the conventional GP approach. An analysis of the DFMs evolved by the GA and GP approaches reveals that the fitness measures applied in the preliminary DFMGP generations support exploration while those applied in later DFMGP generations support exploitation. Exploration and exploitation are the two cornerstones of problem solving by search [22]: exploration, a de facto global search, is used to promote coverage of the search space; in turn, exploitation, a de facto local search, is used to refine promising solutions when good points in the search space have been discovered. GP search is a constant balance between exploration and exploitation, with the former being more suited to the preliminary generations, and the latter, later generations. DFMGP's performance advantage over standard GP is therefore justified by the premise that the fitness measure used on each generation supports the more suitable search in the on-going phase of GP.

- The study compared the use of GP versus the use of a GA to evolve DFMs for DFMGP in chapter 9. The results show that the GP approach yields more effective DFMs. While the GA approach derives a sequence of fitness measures to be applied in DFMGP, the GP approach produces an arithmetic function combining the fitness measures. Therefore, rather than simply switching between explorative and exploitative fitness measures, the latter approach offers the opportunity to combine the two types of fitness measure, driving exploration and exploitation to occur simultaneously when required in DFMGP. This

proves useful, because rather than moving between strict explorative and exploitative phases, GP search ideally maintains a constant trade-off between the two modes of search [22].

- The results in chapter 9 showed that the DFMs derived by GA and GP approaches are reusable, i.e. DFMs can be evolved for a problem class and yield good results on unseen problem instances of the class. The DFMs evolved for a problem class can also be reused on unseen complex, real-world problems from the same problem domain, where DFMGP is shown to achieve better results than standard GP. Importantly, the implications of reusability are that the DFMs are not problem-specific, rather, GP practitioners can save time by deriving more general problem solvers.

10.4 Future Work

Based on the results of this research on DFMGP and different approaches for evolving the DFMs for DFMGP, potential future work will be discussed. Future extensions of the research presented in this thesis include:

10.4.1 Coevolving the parameters used by the fitness measures in DFMGP

In the current research, the parameters used for the fitness measures applied in DFMGP were tuned a priori. These included the fitness measure specific parameters, as well as the selection method and genetic operator application rates used for each fitness measure within DFMGP. The parameters were tuned by applying the SMAC (the Sequential Model-based Algorithm Configuration) parameter tuning algorithm [39] to estimate the optimal configuration for the each fitness measure when used in standard GP. This parameter tuning approach was adopted as an approximation of the optimal configuration for each of the fitness measures in DFMGP. However, in reality, the optimal configurations for the fitness measures when used in standard GP may differ from that when used within DFMGP.

Given the above arguments, a parallel evolutionary algorithm could provide a better solution to parameter tuning. Alternatively, a tuning mechanism built into the GA or GP approach used to evolve the DFMs for DFMGP, based on parameter design approaches such as the Taguchi orthogonal arrays [268], could also be employed. The results obtained from coevolving the parameters used by the fitness measures would be compared to those obtained in this study.

10.4.2 Detecting problem properties

The research found that while DFMGP has the capacity to outperform standard GP on unseen real-world problems, it is difficult to a priori ascertain the problems that can be used to train the DFMs for DFMGP. In this regard, future work will look at designing simple heuristics that can be used to a priori detect the properties of real-world problems. Subsequently, training sets with the same properties can be used to deduce suitable DFMs for DFMGP. For example, the heuristic proposed by Krawiek and Wieloch [158] to detect modularity in Boolean function synthesis GP can be modified and made applicable to different problems. Another relevant study is the work done by Mersmann et. al. [269], who propose exploratory landscape analysis (ELA) techniques for evolutionary algorithms; ELA outlines a number of heuristics for detecting problem properties, including modularity, fitness plateaus, local optima, etc. in the symbolic regression domain. Future work will involve investigating how ELA can be used to detect problem properties for real-world GP problems, and whether suitable training sets for DFMGP can be recommended based on the analysis. Future work will also investigate similar (and more general) heuristics that can be applied to problems from different domains.

10.4.3 Testing other evolutionary algorithm approaches for evolving DFMs

Given this success of the GP approach for evolving DFMs for DFMGP, future work will look at other variations of genetic programming, namely, grammar-based GP [270] and grammatical evolution [271] to evolve the

DFMs. It is anticipated that these approaches would produce better results as the structure of the DFMs will be restricted and the search space reduced. Furthermore, grammatical evolution will reduce the bloat and hence introns in the evolved DFMs.

10.4.4 Testing DFMs on other evolutionary algorithms

Given the effectiveness of dynamic fitness measures for GP, the use of such measures for other evolutionary algorithms, such as genetic algorithms, will also be investigated.

10.5 Summary

This chapter presented a summary of the findings of the research of this thesis and the outcomes of the objectives and how they were fulfilled. Finally, future work based on the observations made during this research was presented.

Bibliography

- [1] J.R. Koza. *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge MA: The MIT press, 1992.
- [2] W. Banzhaf et al. *Genetic Programming: An Introduction*. San Francisco, USA: Morgan Kaufmann, Inc., 1998.
- [3] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading MA: Addison-Wesley, 1989.
- [4] J. Lehman and K.O. Stanley. "Efficiently evolving programs through the search for novelty". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10 (Portland, Oregon, USA)*. Ed. by J. Branke et al. The ACM Press, 2010, pp. 837–844.
- [5] K. Krawiec and U. O'Reilly. "Behavioral programming: a broader and more detailed take on semantic GP". In: *Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation - GECCO'14 (Vancouver, BC, Canada)*. Ed. by C. Igel et al. The ACM Press, 2014, pp. 935–942.
- [6] R.I. McKay. "Fitness Sharing in Genetic Programming". In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation - GECCO'00 (Las Vegas, Nevada, USA)*. Morgan Kaufmann Publishers, 2000, pp. 435–442.
- [7] M. A. Haeri, M. M. Ebadzadeh, and G. Folino. "Improving GP generalization: a variance-based layered learning approach". In: *Genetic Programming and Evolvable Machines* 16(1) (2015), pp. 27–55.
- [8] Eggermont J. and J.I. van Hemert. "Adaptive genetic programming applied to new and existing simple regression problems". In: *LNCS 2038, Proceedings of the 4th European Conference on Genetic Programming - EuroGP'01 (Lake Como, Italy)*. Springer Berlin Heidelberg, 2001, pp. 23–35.
- [9] J. Eggermont, A.E. Eiben, and J.I. van Hemert. "Adapting the fitness function in GP for data mining". In: *LNCS 1598, Proceedings of the 2nd European Conference on Genetic Programming - EuroGP'99 (Goteborg, Sweden)*. Ed. by R. Poli et al. Springer Berlin Heidelberg, 1999, pp. 193–202.
- [10] C. Gathercole and P. Ross. "Tackling the boolean even N parity problem with genetic programming and limited-error fitness". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 119–127.
- [11] C.W. Lasarczyk, P. Dittrich, and W. Banzhaf. "Dynamic subset selection based on a fitness case topology". In: *Evolutionary Computation* 12(2) (2004), pp. 223–242.
- [12] P.J. Angeline and J.B. Pollack. "Competitive Environments Evolve Better Solutions for Complex Tasks". In: *Proceedings of the 5th International Conference on Genetic Algorithms - ICGA'93 (University of Illinois at Urbana-Champaign, USA)*. Ed. by S. Forrest. Morgan Kaufmann Publishers, 1993, pp. 264–270.
- [13] L. Panait and S. Luke. "A comparative study of two competitive fitness functions". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation - GECCO'02 (New York, USA)*. Ed. by J. Branke et al. Morgan Kaufmann Publishers, 2002, pp. 503–511.
- [14] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1(1) (1997), 67–82.
- [15] R.I. McKay. "An investigation of fitness sharing in genetic programming". In: *The Australian Journal of Intelligent Information Processing Systems* 7(1/2) (2001), pp. 43–51.

- [16] J. McDermott et al. "Genetic programming needs better benchmarks". In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation - GECCO'12 (Philadelphia, Pennsylvania, USA)*. Ed. by T. Soule et al. The ACM Press, 2012, pp. 791–798.
- [17] L. Dioşan and M. Oltean. "Evolutionary design of evolutionary algorithms". In: *Genetic Programming and Evolvable Machines* 10(3) (2009), pp. 263–306.
- [18] A. E. Eiben and S. K. Smit. "Evolutionary algorithm parameters and methods to tune them". In: *Autonomous Search*. Ed. by Y. Hamadi, E. Monfroy, and F. Saubion. Springer Berlin Heidelberg, 2011, pp. 15–36.
- [19] B. Edmonds. "Meta-genetic programming: Co-evolving the operators of variation". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 9(1) (2001), pp. 13–29.
- [20] L. Dioşan and M. Oltean. "Evolving crossover operators for function optimization". In: *LNCS 3905, Proceedings of the 9th European Conference on Genetic Programming - EuroGP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, pp. 97–108.
- [21] S.O. Haraldsson and J.R. Woodward. "Automated design of algorithms and genetic improvement: contrast and commonalities". In: *Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation - GECCO'14 (Vancouver, BC, Canada)*. Ed. by C. Igel et al. The ACM Press, 2014, pp. 1373–1380.
- [22] A.E. Eiben and C.A. Schippers. "On evolutionary exploration and exploitation". In: *Fundamenta Informaticae* 35(1) (1998), pp. 35–50.
- [23] C. Darwin. *On the origin of species by means of natural selection*. Facsimile ed. Cambridge, Mass.: Harvard University Press, 1964.
- [24] J. H. Holland. *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.
- [25] M. Mitchell. *An introduction to genetic algorithms*. Cambridge MA: The MIT press, 1998.
- [26] J.J. Grefenstette. "Optimization of control parameters for genetic algorithms". In: *IEEE Transactions on systems, man, and cybernetics* 16(1) (1986), pp. 122–128.
- [27] P. Cortez, M. Rocha, and J. Neves. "A meta-genetic algorithm for time-series forecasting". In: *Proceedings of the 10th Portuguese Conference on Artificial Intelligence - EPIA'01 (Oporto, Portugal)*. Ed. by L. Torgo. 2001, pp. 21–31.
- [28] V. Cicirello and S.F. Smith. "Modelling GA performance for control parameter optimization". In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation - GECCO'00 (Las Vegas, Nevada, USA)*. Morgan Kaufmann Publishers, 2000, pp. 235–242.
- [29] Z. Brain and M. Addicoat. "Using Meta-Genetic algorithms to tune parameters of Genetic Algorithms to find lowest energy Molecular Conformers". In: *Proceedings of the 11th International Conference on Artificial Life - ALIFE XII (Odense, Denmark)*. Ed. by H. Fellersmann. The MIT Press, 2010, pp. 378–385.
- [30] M. Sipper et al. "Investigating the parameter space of evolutionary algorithms". In: *arXiv* (2017), preprint arXiv:1706.04119.
- [31] G. G. Mitchell, B. McMullin, and J. Decraene. "A cost benefit operator for efficient multi level genetic algorithm searches". In: *Proceedings of the 2007 Congress on Evolutionary Computation - CEC'07 (Singapore)*. The IEEE Press, 2007, pp. 1344–1350.
- [32] D. E. Goldberg and K. Deb. "A comparative analysis of selection schemes used in genetic algorithms". In: *Foundations of genetic algorithms*. Ed. by G. Rawlins. Morgan-Kaufmann, 1991, pp. 69–93.
- [33] S. Picek and M. Golub. "Comparison of a crossover operator in binary-coded genetic algorithms". In: *World Scientific and Engineering Academy and Society (WSEAS) transactions on computers* 9 (2010), pp. 1064–1073.

- [34] R. Hinterding. "Gaussian mutation and self-adaptation in numeric genetic algorithms". In: *Proceedings of the 2nd IEEE Conference on Evolutionary Computation (Piscataway, NJ, USA)*. The IEEE Press, 1995, 384–389.
- [35] G. Syswerda. "Simulated Crossover in Genetic Algorithms". In: *Proceedings of the 2nd Workshop on Foundations of Genetic Algorithms*. Ed. by L. Whitley. Morgan-Kaufmann Publishers, 1993, 239–255.
- [36] Z. Michalewicz, C.Z. Janikow, and J.B. Krawczyk. "A modified genetic algorithm for optimal control problems". In: *Computers Mathematics with Applications* 23(12) (1992), pp. 83–94.
- [37] S. K. Smit and A. E. Eiben. "Comparing parameter tuning methods for evolutionary algorithms". In: *Proceedings of the 2009 Congress on Evolutionary Computation - CEC'09 (Trondheim, Norway)*. The IEEE Press, 2009, pp. 399–406.
- [38] F. Hutter et al. "ParamILS: an automatic algorithm configuration framework". In: *Journal of Artificial Intelligence Research* 36(1) (2009), pp. 267–306.
- [39] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *Proceedings of the 5th international conference on Learning and Intelligent Optimization - LION'05 (Rome, Italy)*. Ed. by C. C. Coello. Springer Berlin Heidelberg, 2011, pp. 507–523.
- [40] C. Ansótegui, M. Sellmann, and K. Tierney. "A gender-based genetic algorithm for the automatic configuration of algorithms". In: *LNCS 5732, Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming - CP'09 (Lisbon, Portugal)*. Ed. by I.P. Gent. Springer Berlin Heidelberg, 2009, pp. 142–157.
- [41] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [42] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by simulated annealing". In: *Science* 220(4598) (1983), pp. 671–680.
- [43] D. Whitley et al. "Alternative evolutionary algorithms for evolving programs: evolution strategies and steady state GP". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation - GECCO'06 (Seattle, Washington, USA)*. Ed. by M. Keijzer et al. The ACM Press, 2006, pp. 919–926.
- [44] K. E. Kinneer Jr. "Evolving a sort: Lessons in genetic programming". In: *Proceedings of the IEEE International Conference on Neural Networks (San Francisco, California, USA)*. The IEEE Press, 1993, pp. 881–888.
- [45] W.B. Langdon et al. "Genetic programming: An introduction and tutorial, with a survey of techniques and applications". In: *Computational intelligence: A compendium*. Springer Berlin Heidelberg, 2008, pp. 927–1028.
- [46] D.J. Montana. *Strongly typed genetic programming*. Tech. rep. 7866. Bolt Beranek and Newman, Inc., 1994.
- [47] B.J. Ross. *Searching for search algorithms: Experiments in meta-search*. Tech. rep. CS-02-23. Department of Computer Science, Brock University, 2002.
- [48] L. Dioşan and M. Oltean. "Evolving crossover operators for function optimization". In: *LNCS 3905, Proceedings of the 9th European Conference on Genetic Programming - EuroGP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, pp. 97–108.
- [49] Y. Fang and J. Li. "A Review of Tournament Selection in Genetic Programming". In: *Proceedings of the 5th International Symposium on Advances in Computation and Intelligence - ISICA 2010 (Wuhan, China)*. Ed. by Z. Cai, C. Hu, and Y. Liu Z. Kang. Springer Berlin Heidelberg, 2010, pp. 181–192.
- [50] W. B. Langdon. "Size fair and homologous tree crossovers for tree genetic programming". In: *Genetic Programming and Evolvable Machines* 1(1-2) (2000), pp. 95–119.

- [51] R. Poli and W.B. Langdon. "On the search properties of different crossover operators in genetic programming". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, pp. 293–301.
- [52] Q. U. Nguyen, X. H. Nguyen, and M. O'Neill. "Semantic aware crossover for genetic programming: the case for real-valued function regression". In: *Proceedings of the 9th international conference on Artificial evolution - EA'09 (Strasbourg, France)*. Springer Berlin Heidelberg, 2009, pp. 170–181.
- [53] H. Majeed and C. Ryan. "A less destructive, context-aware crossover operator for GP". In: *LNCS 3905, Proceedings of the 9th European Conference on Genetic Programming - EuroGP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, 36–48.
- [54] H. Majeed and C. Ryan. "Using context-aware crossover to improve the performance of GP". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation - GECCO'06 (Seattle, Washington, USA)*. Ed. by M. Keijzer et al. The ACM Press, 2006, pp. 847–854.
- [55] R. Poli and W.B. Langdon. "Genetic programming with one-point crossover". In: (1998). Ed. by P.K. Chawdhry, R. Roy, and R.K. Pant, pp. 180–189.
- [56] S. Gustafson. "An analysis of diversity in genetic programming". PhD thesis. Nottingham, UK: University of Nottingham, 2004.
- [57] E. K. Burke, S. Gustafson, and G. Kendall. "Diversity in genetic programming: An analysis of measures and correlation with fitness". In: *IEEE Transactions on Evolutionary Computation* 8(1) (2004), pp. 47–62.
- [58] N.F. McPhee and N.J. Hopper. "Analysis of genetic diversity through population history". In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - GECCO'99 (Orlando, Florida, USA)*. The ACM Press, 1999, pp. 1112–1120.
- [59] M. Walker. "Comparing the performance of incremental evolution to direct evolution". In: *Proceedings of the 2nd International Conference on Autonomous Robots and Agents - ICARA 2004 (Palmerston North, New Zealand)*. Ed. by S. Mukhopadhyay and G. Gupta. 2004.
- [60] J.F. Winkeler and B.S. Manjunath. "Incremental evolution in genetic programming". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, pp. 403–411.
- [61] D. Jackson. "Partitioned incremental evolution of hardware using genetic programming". In: *Proceedings of the 2007 Congress on Evolutionary Computation - CEC'07 (Singapore)*. The IEEE Press, 2007, pp. 86–97.
- [62] G.J. Barlow, C.K. Oh, and E. Grant. "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multiobjective genetic programming". In: *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems - CIS'04 (Singapore)*. The IEEE Press, 2004, pp. 689–694.
- [63] W. H. Hsu and S. M. Gustafson. "Genetic Programming And Multi-agent Layered Learning By Reinforcements". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation - GECCO'02 (New York, USA)*. Ed. by J. Branke et al. Morgan Kaufmann Publishers, 2002, pp. 764–771.
- [64] W. H. Hsu et al. "Empirical comparison of incremental reuse strategies in genetic programming for keep-away soccer". In: *Late Breaking Papers, Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation - GECCO'04 (Seattle, Washington, USA)*. Ed. by K. Deb et al. Morgan Kaufmann Publishers, 2004.
- [65] J. K. Kishore et al. "Application of genetic programming for multicategory pattern classification". In: *IEEE Transactions on Evolutionary Computation* 4(3) (2000), pp. 242–258.

- [66] K. Krawiec and J. Swan. "Pattern-guided genetic programming". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 949–956.
- [67] T.F. Bersano-Begey. "Controlling Exploration, diversity and escaping local optima in GP: Adapting weights of training sets to model resource consumption". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 7–10.
- [68] A. Ekárt and S. Z. Németh. "A metric for genetic programs and fitness sharing". In: *LNCS 1802, Proceedings of the 3rd European Conference on Genetic Programming - EuroGP'00 (Edinburgh, Scotland, UK)*. Ed. by R. Poli et al. Springer-Verlag, 2000, pp. 259–270.
- [69] A. Ekárt and S. Z. Németh. "Maintaining the diversity of genetic programs". In: *LNCS 2278, Proceedings of the 5th European Conference on Genetic Programming - EuroGP'02 (Kinsale, Ireland)*. Ed. by J.A. Foster et al. Springer-Verlag, 2002, pp. 162–171.
- [70] M. Gaudesi, G. Squillero, and A. Tonda. "Universal information distance for genetic programming". In: *Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation - GECCO'14 (Vancouver, BC, Canada)*. Ed. by C. Igel et al. The ACM Press, 2014, pp. 137–138.
- [71] C. Gathercole and P. Ross. "Dynamic training subset selection for supervised learning in genetic programming". In: *LNCS 866, Proceedings of the 3rd International Conference on Parallel Problem Solving From Nature - PPSN III (Jerusalem, Israel)*. Ed. by Y. Davidor et al. Springer Berlin Heidelberg, 1994, pp. 312–321.
- [72] J. Doucette and M.I. Heywood. "Novelty-based fitness: An evaluation under the santa fe trail". In: *LNCS 6021, Proceedings of the 13th European Conference in Genetic Programming - EuroGP'10 (Istanbul, Turkey)*. Ed. by A.I. Esparcia-Alcazar et al. Springer Berlin Heidelberg, 2010, pp. 50–61.
- [73] E. Naredo and L. Trujillo. "Searching for novel clustering programs". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 1093–1100.
- [74] E. Naredo, L. Trujillo, and Y. Martínez. "Searching for novel classifiers". In: *LNCS 7831, Proceedings of the 16th European Conference in Genetic Programming - EuroGP'13 (Vienna, Austria)*. Ed. by K. Krawiec et al. Springer Berlin Heidelberg, 2013, pp. 145–156.
- [75] J. Lehman and K.O. Stanley. "Novelty Search and the Problem with Objectives". In: *Genetic Programming in Theory and Practice IX (GPTP 2011)*. Springer, 2011. Chap. 3, pp. 37–56.
- [76] Y. Martinez et al. "Searching for novel regression functions". In: *Proceedings of the 2013 Congress on Evolutionary Computation - CEC'13 (Cancun, Mexico)*. The IEEE Press, 2013, pp. 16–23.
- [77] J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Subprograms*. Cambridge MA: The MIT press, 1994.
- [78] J.R. Koza et al. *Genetic programming III: Darwinian invention and problem solving*. San Francisco, CA: Morgan Kaufmann, 1999.
- [79] M. Santini and A. Tettamanzi. "Genetic programming for financial time series prediction". In: *LNCS 2038, Proceedings of the 4th European Conference on Genetic Programming - EuroGP'01 (Lake Como, Italy)*. Springer Berlin Heidelberg, 2001, pp. 361–370.
- [80] N. Wagner et al. "Time series forecasting for dynamic environments: the DyFor genetic program model". In: *IEEE Transactions on Evolutionary Computation* 11(4) (2007), pp. 433–452.
- [81] D. Jakobović and L. Budin. "Dynamic scheduling with genetic programming". In: *LNCS 3905, Proceedings of the 9th European Conference on Genetic Programming - EuroGP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, pp. 73–84.

- [82] J. R. Koza. "A genetic approach to the truck backer upper problem and the intertwined spiral problem". In: *Proceedings of the International Joint Conference on Neural Networks - IJCNN'92 (Baltimore, Maryland, USA)*. IEEE Press, 1992, pp. 310–318.
- [83] H. Jäske. "Prediction of Sunspots by GP". In: *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications - 2NWGA (Vassa, Finland)*. Ed. by J.T. Alander. Vaasan yliopisto, 1996, 79–88.
- [84] J. F. Winkeler and B. S. Manjunath. "Genetic programming for object detection". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 330–335.
- [85] J. Mouret and S. Doncieux. "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity". In: *Proceedings of the 2009 Congress on Evolutionary Computation - CEC'09 (Trondheim, Norway)*. The IEEE Press, 2009, pp. 1161–1168.
- [86] S. Nolfi and D. Floreano. "How co-evolution can enhance the adaptive power of artificial evolution - implications for evolutionary robotics". In: *Proceedings of the First European Workshop on Evolutionary Robotics - EvoRobot 98 (Paris, France)*. Springer-Verlag, Berlin Heidelberg, 1998, pp. 2–8.
- [87] J. Urzelai et al. "Incremental robot shaping". In: *Connection Science Journal* 10(384) (1998), pp. 341–360.
- [88] E. Uchibe, M. Yanase, and M. Asada. "Behavior generation for a mobile robot based on the adaptive fitness function". In: *Robotics and Autonomous Systems* 40(2) (2002), pp. 69–77.
- [89] W.F. Punch, D. Zongker, and E.D. Goodman. "The royal tree problem, a benchmark for single and multi-population genetic programming". In: *Advances in Genetic Programming*. Ed. by P.J. Angeline and K.E. Kinnear. Vol. 2. Cambridge MA, USA: The MIT Press, 1996, pp. 299–316.
- [90] W.F. Punch. "How effective are multiple populations in genetic programming". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, pp. 308–313.
- [91] W.B. Langdon and R. Poli. "An analysis of the MAX problem in genetic programming". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 222–230.
- [92] C. Gathercole and P. Ross. "An adverse interaction between crossover and restricted tree depth in genetic programming". In: *Proceedings of the 1st Annual Conference on Genetic Programming - GP'96 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. The MIT Press, 1996, pp. 291–296.
- [93] N. S. Chaudhari, A. Purohit, and A. Tiwari. "Genetic Programming for Classification". In: *International Journal of Computer and Electronics Engineering, IJCEE* 1 (2009), pp. 69–76.
- [94] L. Vanneschi, M. Castelli, and L. Manzoni. "The K landscapes: a tunably difficult benchmark for genetic programming". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO'11 (Dublin, Ireland)*. Ed. by N. Krasnogor et al. The ACM Press, 2011, pp. 1467–1474.
- [95] T. H. Hoang et al. "ORDERTREE: a new test problem for genetic programming". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation - GECCO'06 (Seattle, Washington, USA)*. Ed. by M. Keijzer et al. The ACM Press, 2006, pp. 807–814.
- [96] W. Weimer et al. "Automatically finding patches using genetic programming". In: *Proceedings of the 31st International Conference on Software Engineering - ICSE'09 (Vancouver, Canada)*. The IEEE Press, 2009, pp. 364–374.
- [97] D. P. Muni and N. R. Pal. "Evolution of fuzzy classifiers using genetic programming". In: *Fuzzy Information and Engineering* 4 (1) (2012), pp. 29–49.
- [98] W. B. Langdon. "Evolving Data Structures with Genetic Programming". In: *Proceedings of the 6th International Conference on Genetic Algorithms - ICGA'95 (Pittsburgh, PA, USA)*. Ed. by L.J. Eshelman. Morgan Kaufmann Publishers, 1995, pp. 295–302.

- [99] W.B. Langdon and R. Poli. "Why ants are hard". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, pp. 193–201.
- [100] K. Krawiec. *Behavioral Program Synthesis with Genetic Programming*. Springer, 2016.
- [101] S. Luke and L. Panait. "A comparison of bloat control methods for genetic programming". In: *Evolutionary Computation* 14(3) (2006), pp. 309–344.
- [102] W. B. Langdon and R. Poli. "Fitness causes bloat". In: *Proceedings of the 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (London, UK)*. Ed. by P.K. Chawdhry et al. Springer Berlin Heidelberg, 1997, pp. 13–22.
- [103] W.B. Langdon. *Fitness causes bloat in variable size representations*. Tech. rep. CSRP-97-14. University of Birmingham, School of Computer Science, 1997.
- [104] P. Nordin and W. Banzhaf. "Complexity Compression and Evolution". In: *Proceedings of the 6th International Conference on Genetic Algorithms - ICGA'95 (Pittsburgh, PA, USA)*. Ed. by L.J. Eshelman. Morgan Kaufmann Publishers, 1995, pp. 310–317.
- [105] W. A. Tackett. "Genetic Programming for Feature Discovery and Image Discrimination". In: *Proceedings of the 5th International Conference on Genetic Algorithms - ICGA'93 (University of Illinois at Urbana-Champaign, USA)*. Ed. by S. Forrest. Morgan Kaufmann Publishers, 1993, pp. 303–311.
- [106] K. E. Kinnear Jr. "Generality and Difficulty in Genetic Programming: Evolving a Sort". In: *Proceedings of the 5th International Conference on Genetic Algorithms - ICGA'93 (University of Illinois at Urbana-Champaign, USA)*. Ed. by S. Forrest. Morgan Kaufmann Publishers, 1993, pp. 287–294.
- [107] W. B. Langdon. "Quadratic Bloat in Genetic Programming". In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation - GECCO'00 (Las Vegas, Nevada, USA)*. Morgan Kaufmann Publishers, 2000, pp. 451–458.
- [108] R. Poli and N. F. McPhee. "Parsimony pressure made easy". In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation - GECCO'08 (Atlanta, Georgia, USA)*. Ed. by M. Keijzer et al. The ACM Press, 2008, pp. 1267–1274.
- [109] T. Soule and R. B. Heckendorn. "Some considerations on the reason for bloat". In: *Genetic Programming and Evolvable Machines* 3(3) (2002), pp. 283–309.
- [110] N. F. McPhee and J.D. Miller. "Accurate Replication in Genetic Programming". In: *Proceedings of the 6th International Conference on Genetic Algorithms - ICGA'95 (Pittsburgh, PA, USA)*. Ed. by L.J. Eshelman. Morgan Kaufmann Publishers, 1995, pp. 303–309.
- [111] T. Soule and R. B. Heckendorn. "An analysis of the causes of code growth in genetic programming". In: *Genetic Programming and Evolvable Machines* 3(3) (2002), pp. 283–309.
- [112] M.J. Streeter. "The root causes of code growth in genetic programming". In: *LNCS 2610, Proceedings of the 6th European Conference on Genetic Programming - EuroGP'03 (Essex, UK)*. Ed. by C. Ryan et al. Springer-Verlag, 2003, 449–458.
- [113] W.B. Langdon. *Fitness causes bloat: Simulated annealing, hill climbing and populations*. Tech. rep. CSRP-97-22. University of Birmingham, School of Computer Science, 1997.
- [114] L. Altenberg. "The evolution of evolvability in genetic programming". In: *Advances in genetic programming* 3 (1994), pp. 47–74.
- [115] W. B. Langdon et al. "The Evolution of Size and Shape". In: *Advances in Genetic Programming*. Ed. by L. Spector et al. Vol. 3. Cambridge MA, USA: The MIT Press, 1999. Chap. 8.
- [116] P. J. Angeline. "Genetic programming and emergent intelligence". In: *Advances in genetic programming*. Vol. 1. 1994, pp. 75–98.

- [117] G. P. Wagner and L. Altenberg. "Perspective: complex adaptations and the evolution of evolvability". In: *Evolution* 50 (1996), pp. 967–976.
- [118] T. Blickle and L. Thiele. "Genetic programming and redundancy". In: *Proceedings of the Workshop on Genetic Algorithms within the Framework of Evolutionary Computation, 18th German Annual Conference on Artificial Intelligence - K1'94 (Saarbrücken, Germany)*. Ed. by B. Nebel et al. Springer Berlin Heidelberg, 1994, pp. 33–38.
- [119] G. R. Price. "Selection and covariance". In: *Nature* 227 (1970), pp. 520–521.
- [120] R. Curry, P. Lichodziejewski, and M.I. Heywood. "Scaling genetic programming to large datasets using hierarchical dynamic subset selection". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37(4) (2007), pp. 1065–1073.
- [121] Zhang B.T. and D.Y. Cho. "Genetic programming with active data selection". In: *Proceedings of the 2nd Asia-Pacific Conference on Simulated Evolution and Learning - SEAL'98, Volume 1 (Canberra, Australia)*. Ed. by B. McKay et al. Springer Berlin Heidelberg, 1998, pp. 146–153.
- [122] L. Wetmore, M.I. Heywood, and A.N. Zincir-Heywood. "Speeding up the self-organizing feature map using dynamic subset selection". In: *Neural processing letters* 22(1) (2005), pp. 17–32.
- [123] F. Fernandez et al. "A distributed computing environment for genetic programming using MPI". In: *Recent advances in parallel virtual machine and message passing interface: 7th European PVM/MPI Users' Group Meeting (New York, NY, USA)*. Ed. by J. J. Dongarra, P. Kacsuk, and N. Podhorszki. Springer-Verlag Inc., 2000, pp. 322–329.
- [124] D. Song, M.I. Heywood, and A.N. Zincir-Heywood. "Training genetic programming on half a million patterns: an example from anomaly detection". In: *IEEE Transactions on Evolutionary Computation* 9(3) (2005), pp. 225–239.
- [125] A.A. Freitas. "A genetic programming framework for two data mining tasks: classification and generalized rule induction". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 96–101.
- [126] M. L. Raymer et al. "Genetic programming for improved data mining: application to the biochemistry of protein interactions". In: *Proceedings of the 1st Annual Conference on Genetic Programming - GP'96 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. The MIT Press, 1996, pp. 375–380.
- [127] I. Goncalves and S. Silva. "Balancing learning and overfitting in genetic programming with interleaved sampling of training data". In: *LNCS 7831, Proceedings of the 16th European Conference in Genetic Programming - EuroGP'13 (Vienna, Austria)*. Ed. by K. Krawiec et al. Springer Berlin Heidelberg, 2013, 73–84.
- [128] I. Goncalves and S. Silva. "Experiments on controlling overfitting in genetic programming". In: *LNCS 7026, Proceedings of the 15th Portuguese Conference on Artificial Intelligence - EPIA'11 (Lisbon, Portugal)*. APPIA, 2011.
- [129] Y. Liu and T. Khoshgoftaar. "Reducing overfitting in genetic programming models for software quality classification". In: *Proceedings of the 9th IEEE International Symposium on High-Assurance Systems Engineering - HASE'05 (Heidelberg, Germany)*. 2005, pp. 56–65.
- [130] R.A. Watson and J.B. Pollack. "Coevolutionary dynamics in a minimal substrate". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation - GECCO'01 (San Francisco, California, USA)*. Ed. by L. Spector et al. Morgan Kaufmann Publishers, 2001, pp. 702–709.
- [131] K. Sims. "Evolving 3D morphology and behavior by competition". In: *Artificial life* 1(4) (1994), pp. 353–372.
- [132] S. Luke. "Genetic programming produced competitive soccer softbot teams for robocup97". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, pp. 214–222.

- [133] S. Luke. "Evolving soccerbots: A retrospective". In: *Proceedings of the 12th Annual Conference of the Japanese Society for Artificial Intelligence - JSAI'98 (Tokyo, Japan)*. Morgan Kaufmann Publishers, 1998.
- [134] A. Hauptman and M. Sipper. "GP-endchess: Using genetic programming to evolve chess endgame players". In: *LNCS 3447, Proceedings of the 8th European Conference on Genetic Programming - EuroGP'05 (Lausanne, Switzerland)*. Ed. by M. Keijzer et al. Springer, 2005, pp. 120–131.
- [135] G. F. Miller and D. Cliff. "Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics". In: *Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3 (Brighton, England)*. Ed. by D. Cliff et al. 1994, pp. 411–420.
- [136] C.W. Reynolds. "Competition, coevolution and the game of tag". In: *Proceedings of the 4th Artificial Life Conference*. Ed. by R. Brooks and P. Maes. The MIT Press, 1994, pp. 59–69.
- [137] D. Floreano and S. Nolfi. "God save the red queen! Competition in co-evolutionary robotics". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 398–406.
- [138] C. Onga, Huang J, and Tzeng G. "Building credit scoring models using genetic programming". In: *Expert Systems with Applications* 29(1) (2005), pp. 41–47.
- [139] M. Brezocnik and M. Kovacic. "Prediction of surface roughness with genetic programming". In: *Journal of materials processing technology* 157 (2004), pp. 28–36.
- [140] A. Johari, G. Habibagahi, and A. Ghahramani. "Prediction of soil–water characteristic curve using genetic programming". In: *Journal of Geotechnical and Geoenvironmental Engineering* 132(5) (2006), pp. 661–665.
- [141] M. Kovacic et al. "Prediction of the bending capability of rolled metal sheet by genetic programming". In: *Materials and manufacturing processes* 22(5) (2007), pp. 634–640.
- [142] K. E. Kinnear Jr. "Fitness landscapes and difficulty in genetic programming". In: *Proceedings of the 1st IEEE Conference on Evolutionary Computation - CEC'94 (Orlando, Florida, USA)*. The IEEE Press, 1994, pp. 142–147.
- [143] L. Vanneschi et al. "Fitness clouds and problem hardness in genetic programming". In: *Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation - GECCO'04 (Seattle, Washington, USA)*. Ed. by K. Deb et al. Morgan Kaufmann Publishers, 2004, 690–701.
- [144] L. Vanneschi et al. "Negative slope coefficient. A measure to characterize genetic programming". In: *LNCS 2278, Proceedings of the 9th European Conference on Genetic Programming - Euro GP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, 178–189.
- [145] M. Clergue et al. "Fitness Distance Correlation And Problem Difficulty For Genetic Programming". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation - GECCO'02 (New York, USA)*. Ed. by J. Branke et al. Morgan Kaufmann Publishers, 2002, 724–732.
- [146] M. Tomassini et al. "A study of fitness distance correlation as a difficulty measure in genetic programming". In: *Evolutionary Computation* 13(2) (2005), pp. 213–239.
- [147] E. Galvan-Lopez et al. "Defining locality as a problem difficulty measure in genetic programming". In: *Genetic Programming and Evolvable Machines* 12(4) (2011), pp. 365–401.
- [148] K. Krawiec and A. Solar-Lezama. "Improving Genetic Programming with Behavioral Consistency Measure". In: *Proceedings of the 13th International Conference on Parallel Problem Solving From Nature - PPSN XIII (Ljubljana, Slovenia)*. Ed. by T. Bartz-Beielstein et al. Springer Berlin Heidelberg, 2014, pp. 434–443.
- [149] K. Krawiec and U. M. O'Reilly. "Behavioral Search Drivers for Genetic Programming". In: *LNCS 8599, Proceedings of the 17th European Conference in Genetic Programming - EuroGP'14 (Grenada, Spain)*. Ed. by M. Nicolau et al. Springer Berlin Heidelberg, 2014, 210–221.

- [150] K. Krawiec. "Genetic programming: where meaning emerges from program code". In: *Genetic Programming and Evolvable Machines* 15(1) (2014), pp. 75–77.
- [151] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog. "Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming". In: *IEEE Transactions on Evolutionary Computation* 13(2) (2009), pp. 333–349.
- [152] S. Silva, M. Castelli, and L. Vanneschi. "Measuring bloat, overfitting and functional complexity in genetic programming". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10 (Portland, Oregon, USA)*. Ed. by J. Branke et al. The ACM Press, 2010, pp. 877–884.
- [153] M. O'Neill et al. "Open issues in genetic programming". In: *Genetic Programming and Evolvable Machines* 11(3-4) (2010), pp. 339–363.
- [154] R. Azad and C. Ryan. "Variance based selection to improve test set performance in genetic programming". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO'11 (Dublin, Ireland)*. Ed. by N. Krasnogor et al. The ACM Press, 2011, pp. 1315–1322.
- [155] J. Torresen. "Two-step incremental evolution of a prosthetic hand controller based on digital logic gates". In: *Proceedings of the 4th International Conference on Evolvable Systems: From Biology to Hardware - ICES'01 (Tokyo, Japan)*. Ed. by K. Tanaka et al. Springer Berlin Heidelberg, 2001, pp. 1–13.
- [156] J. Torresen. "Evolving multiplier circuits by training set and training vector partitioning". In: *LNCS 2606, Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware - ICES'03 (Trondheim, Norway)*. Springer Berlin Heidelberg, 2003, pp. 228–237.
- [157] D. P. Muni, N. R. Pal, and J. Das. "A novel approach to design classifiers using genetic programming". In: *IEEE Transactions on Evolutionary Computation* 8(2) (2004), pp. 183–196.
- [158] K. Krawiec and B. Wieloch. "Analysis of semantic modularity for genetic programming". In: *Foundation Of Computing And Decision Sciences* 34(4) (2009), p. 265.
- [159] J.B. Mouret and S. Doncieux. "Incremental evolution of animats' behaviors as a multiobjective optimization". In: *Proceedings of the 10th International Conference on Simulation of Adaptive Behavior: From Animals to Animats10 (Osaka, Japan)*. Ed. by M. Asada et al. 2008, pp. 210–219.
- [160] J. R. Quinlan. *C4.5: Programs for machine learning*. San Mateo: Morgan Kaufmann, 1992.
- [161] R. M. Gray. *Entropy and information theory*. Springer Science and Business Media, 2011.
- [162] Q. Nguyen et al. "An investigation of fitness sharing with semantic and syntactic distance metrics". In: *LNCS 7244, Proceedings of the 15th European Conference in Genetic Programming - EuroGP'12 (Malaga, Spain)*. Ed. by A. Moraglio et al. Springer Berlin Heidelberg, 2012, 109–120.
- [163] S.W. Mahfoud. "Nicheing methods for genetic algorithms". PhD thesis. Urbana, IL: University of Illinois at Urbana-Champaign, 1995.
- [164] D. E. Goldberg and J. J. Richardson. "Genetic algorithms with sharing for multimodal function optimization". In: *Proceedings of the 2nd International Conference on Genetic Algorithms - ICGA'87 (Hillsdale, NJ, USA)*. Ed. by J.J. Grefenstette. Lawrence Erlbaum Associates, 1987, pp. 41–49.
- [165] B. Sareni and L. Krahenbuhl. "Fitness sharing and niching methods revisited". In: *IEEE Transactions on Evolutionary Computation* 2(3) (1998), pp. 97–106.
- [166] L. Trujillo et al. "A behavior-based analysis of modal problems". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 1047–1054.
- [167] L. Vanneschi, M. Castelli, and S. Silva. "A survey of semantic methods in genetic programming". In: *Genetic Programming and Evolvable Machines* 15(2) (2014), pp. 195–214.

- [168] T. Weise, R. Chiong, and K. Tang. "Evolutionary optimization: Pitfalls and booby traps". In: *Journal of Computer Science and Technology* 27(5) (2012), pp. 907–936.
- [169] C. Igel and K. Chellapilla. "Investigating the influence of depth and degree of genotypic change on fitness in genetic programming". In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - GECCO'99 (Orlando, Florida, USA)*. The ACM Press, 1999, pp. 1061–1068.
- [170] K. Krawiec and M. Pawlak. "Genetic Programming with Alternative Search Drivers for Detection of Retinal Blood Vessels". In: *Proceedings of the 18th European Conference on Applications of Evolutionary Computation - EvoApplications 2015 (Copenhagen, Denmark)*. Ed. by A.M. Mora and G. Squillero. Springer International Publishing, 2015, pp. 554–566.
- [171] C.W. Reynolds. "An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion". In: *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. Ed. by J.A. Meyer, H.L. Roitblat, and S.W. Wilson. Cambridge, MA: The MIT Press, 1993, pp. 384–392.
- [172] R.I. McKay. "Variants of genetic programming for species distribution modelling-fitness sharing, partial functions, population evaluation". In: *Ecological Modelling* 146(1) (2001), pp. 231–241.
- [173] D.A. Augusto, H.J.C. Barbosa, and N.F.F. Ebecken. "Coevolutionary multi-population genetic programming for data classification". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10 (Portland, Oregon, USA)*. Ed. by J. Branke et al. The ACM Press, 2010, pp. 933–940.
- [174] I. Goncalves et al. "Random sampling technique for overfitting control in genetic programming". In: *LNCS 7244, Proceedings of the 15th European Conference in Genetic Programming - EuroGP'12 (Malaga, Spain)*. Ed. by A. Moraglio et al. Springer Berlin Heidelberg, 2012, pp. 218–229.
- [175] J. Fitzgerald, R. Azad, and C. Ryan. "A bootstrapping approach to reduce over-fitting in genetic programming". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 1113–1120.
- [176] B.J. Ross. "The Effects of Randomly Sampled Training Data on Program Evolution". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10 (Portland, Oregon, USA)*. Ed. by J. Branke et al. The ACM Press, 2010, pp. 443–450.
- [177] J. Hugues and J.B. Pollack. "Co-evolving intertwined spirals". In: *Proceedings of the 5th Annual Conference on Evolutionary Programming (San Diego, CA, USA)*. The MIT Press, 1996, pp. 461–468.
- [178] N. Williams and M. Mitchell. "Investigating the success of spatial coevolution". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation - GECCO'05 (Washington DC, USA)*. Ed. by H. Beyer et al. Morgan Kaufmann Publishers, 2005, pp. 523–530.
- [179] M.J. Aitkenhead. "A co-evolving decision tree classification method". In: *Expert Systems with Applications* 34(1) (2008), pp. 18–25.
- [180] E.V. Siegel. "Competitively evolving decision trees against fixed training cases for natural language processing". In: *Advances in Genetic Programming*. Vol. 19. The MIT Press, 1994, pp. 409–423.
- [181] L. Pagie and P. Hogeweg. "Evolutionary consequences of coevolving targets". In: *Evolutionary Computation* 5(4) (1997), pp. 401–418.
- [182] R. Kala. "Multi-robot path planning using co-evolutionary genetic programming". In: *Expert Systems with Applications* 39(3) (2012), pp. 3817–3831.
- [183] M. E. Roberts and E. Claridge. "Cooperative coevolution of image feature construction and object detection". In: *LNCS 3242, Proceedings of the 8th International Conference on Parallel Problem Solving From Nature - PPSN VIII (Birmingham, UK)*. Ed. by X. Yao et al. Springer Berlin Heidelberg, 2004, pp. 902–911.

- [184] L. Vanneschi et al. "Heterogeneous cooperative coevolution: strategies of integration between GP and GA". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation - GECCO'06 (Seattle, Washington, USA)*. Ed. by M. Keijzer et al. The ACM Press, 2006, pp. 361–368.
- [185] R. P. Wiegand. "An analysis of cooperative coevolutionary algorithms". PhD thesis. George Mason University, 2003.
- [186] A. Samuel. "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 44(1-2) (2000), pp. 206–226.
- [187] R. Axelrod. "The evolution of strategies in the iterated prisoner's dilemma". In: *Genetic Algorithms and Simulated Annealing*. Ed. by L. Davis. Cambridge University Press New York, NY, 1989, pp. 32–41.
- [188] W.D. Hillis. "Co-evolving parasites improve simulated evolution as an optimization procedure". In: *Physica D: Nonlinear Phenomena* 42(1) (1990), pp. 228–234.
- [189] C.D. Rosin and R.K. Belew. "GP-gammon: Genetically programming backgammon players". In: *Evolutionary Computation* 5(1) (1997), pp. 1–29.
- [190] T. Haynes and S. Sen. "Evolving behavioral strategies in predators and prey". In: *Adaptation and Learning in Multi-Agent Systems* (1996), pp. 113–126.
- [191] P. Funes et al. "Animal-Animat Coevolution: Using the Animal Population as Fitness Function". In: *Proceedings of the 5th International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 5 (Zürich, Switzerland)*. The MIT Press, 1998, pp. 525–533.
- [192] Y. Azaria and M. Sipper. "GP-gammon: Genetically programming backgammon players". In: *Genetic Programming and Evolvable Machines* 6(3) (2005), pp. 283–300.
- [193] M. Sipper et al. "Designing an evolutionary strategizing machine for game playing and beyond". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 37(4) (2007), pp. 583–593.
- [194] S. Luke et al. "Co-evolving soccer softbot team coordination with genetic programming". In: *Proceedings of RoboCup-97: Robot Soccer World Cup I (Nagoya, Japan)*. Ed. by H.Kitano et al. Springer Verlag, Berlin, 1997, pp. 398–411.
- [195] G.J. Ferrer and W.N. Martin. "Using genetic programming to evolve board evaluation functions". In: *Proceedings of the 1995 IEEE Conference on Evolutionary Computation Volume 2 (Perth, Australia)*. The IEEE Press, 1995, pp. 747–752.
- [196] J. Jannink. "Cracking and co-evolving randomizers". In: *Advances in Genetic Programming*. Cambridge, MA, USA: The MIT Press, 1994. Chap. 20, pp. 425–443.
- [197] S. Nolfi and D. Floreano. "Coevolving predator and prey robots: Do 'arms races' arise in artificial evolution?" In: *Artificial Life* 4(4) (1998), pp. 311–335.
- [198] A. Hauptman and M. Sipper. "Emergence of complex strategies in the evolution of chess endgame players". In: *Advances in Complex Systems* 10(1) (2007), pp. 35–59.
- [199] L. Vanneschi et al. "A Survey of Problem Difficulty in Genetic Programming". In: *Proceedings of the 9th Congress of the Italian Association for Artificial Intelligence - AIIA'05 (Milan, Italy)*. Ed. by S. Bandini and S. Manzoni. Springer, 2005, 66–77.
- [200] M. Nguyen S.and Zhang, M. Johnston, and K. C. Tan. "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming". In: *IEEE Transactions on Evolutionary Computation* 18(2) (2014), pp. 193–208.
- [201] M. A. Potter. "The design and analysis of a computational model of cooperative coevolution". PhD thesis. George Mason University, 1997.

- [202] S. Kistemaker and S. Whiteson. "Critical factors in the performance of novelty search". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO'11 (Dublin, Ireland)*. Ed. by N. Krasnogor et al. The ACM Press, 2011, pp. 965–972.
- [203] J. Lehman, K.O. Stanley, and R. Miikkulainen. "Effective diversity maintenance in deceptive domains". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 215–222.
- [204] J. Mouret. "Novelty-based multiobjectivization". In: *Workshop on Exploring New Horizons in Evolutionary Design of Robots, Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (St Louis, Missouri, USA)*. The IEEE Press, 2009, pp. 139–154.
- [205] J. Lehman and K.O. Stanley. "Revising the evolutionary computation abstraction: minimal criteria novelty search". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10 (Portland, Oregon, USA)*. Ed. by J. Branke et al. The ACM Press, 2010, pp. 103–110.
- [206] J. Gomes, P. Urbano, and A.L. Christensen. "Progressive minimal criteria novelty search". In: *LNAI 7637, Proceedings of the 13th Ibero-American Conference on Artificial Intelligence - IBERAMIA'12 (Cartagena de Indias, Colombia)*. Springer Berlin Heidelberg, 2012, 281–290.
- [207] S. O. Kimbrough et al. "On a Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch". In: *European Journal of Operational Research* 190(2) (2008), 310–327.
- [208] A. Liapis, G. N. Yannakakis, and J. Togelius. "Enhancements to constrained novelty search: Two-population novelty search for generating game content". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 343–350.
- [209] K. Deb. *Multi-objective optimization using evolutionary algorithms (Vol. 16)*. New York: John Wiley and Sons, 2001.
- [210] L. Trujillo, E. Naredo, and Y. Martínez. "Preliminary study of bloat in genetic programming with behavior-based search". In: *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV, volume 227 of Advances in Intelligent Systems and Computing* (2013), 293–305.
- [211] S. Doncieux and J.B. Mouret. "Behavioral diversity measures for Evolutionary Robotics". In: *Proceedings of the 2010 Congress on Evolutionary Computation - CEC'10 (Barcelona, Spain)*. The IEEE Press, 2010, pp. 1–8.
- [212] G. Cuccu and F. Gomez. "When novelty is not enough". In: *LNCS6624, Proceedings of the European Conference on the Applications of Evolutionary Computation - EvoApplications 2011 (Torino, Italy)*. Ed. by C. Di Chio et al. Springer Berlin Heidelberg, 2011, pp. 234–243.
- [213] S. R. Garner. "Weka: The waikato environment for knowledge analysis". In: *Proceedings of the New Zealand computer science research students conference (Hamilton, New Zealand)*. 1995, pp. 57–64.
- [214] F. Dobslaw. "Recent development in automatic parameter tuning for metaheuristics". In: *Proceedings of the 19th Annual Conference of Doctoral Students - WDS'10 (Charles University, Prague, Czech Republic)*. Matfyzpress, 2010, 54–63.
- [215] S. Luke and L. Spector. "A Comparison of Crossover and Mutation in Genetic Programming". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 222–230.
- [216] C. Gathercole and P. Ross. "Small populations over many generations can beat large populations over few generations in genetic programming". In: *Proceedings of the 2nd Annual Conference on Genetic Programming - GP'97 (Stanford University, CA, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1997, pp. 111–118.

- [217] M. Walker, H. Edwards, and C. Messom. "Success effort and other statistics for performance comparisons in genetic programming". In: *Proceedings of the 2007 Congress on Evolutionary Computation - CEC'07 (Singapore)*. The IEEE Press, 2007, pp. 4631–4638.
- [218] J.R. Koza. "What is Genetic Programming (GP)?, How Genetic Programming Works". In: URL: <http://www.genetic-programming.com/> (2007).
- [219] E. Alpaydin. *Introduction to machine learning*. Cambridge MA: The MIT press, 2014.
- [220] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley and Sons, 2012.
- [221] C. Gagné et al. "Genetic programming, validation sets, and parsimony pressure". In: *LNCS 3905, Proceedings of the 9th European Conference on Genetic Programming - EuroGP'06 (Budapest, Hungary)*. Ed. by P. Collet et al. Springer, 2006, pp. 109–120.
- [222] A. E. Eiben and M. Jelasity. "A critical note on experimental research methodology in EC." In: *Proceedings of the 2002 Congress on Evolutionary Computation - CEC'02 (Honolulu, Hawaii)*. The IEEE Press, 2002, pp. 582–587.
- [223] M. Keijzer. "Improving symbolic regression with interval arithmetic and linear scaling". In: *LNCS 2610, Proceedings of the 6th European Conference on Genetic Programming - EuroGP'03 (Essex, UK)*. Ed. by C. Ryan et al. Springer-Verlag, 2003, pp. 70–82.
- [224] A. Agapitos et al. "Higher order functions for kernel regression". In: *LNCS 8599, Proceedings of the 17th European Conference in Genetic Programming - EuroGP'14 (Grenada, Spain)*. Ed. by M. Nicolau et al. Springer Berlin Heidelberg, 2014, 1–12.
- [225] M. Kommenda et al. "Symbolic regression with sampling". In: *Proceedings of the 22nd European Modeling and Simulation Symposium - EMSS 2010 (Fes, Morocco)*. 2010, pp. 13–18.
- [226] E.K Burke et al. "Advanced population diversity measures in genetic programming". In: *LNCS 2439, Proceedings of the 7th International Conference on Parallel Problem Solving From Nature - PPSN VII (Granada, Spain)*. Ed. by J.J. Merelo et al. Springer Berlin Heidelberg, 2002, pp. 341–350.
- [227] E.K Burke, S. Gustafson, and G. Kendall. "A Survey And Analysis Of Diversity Measures In Genetic Programming". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation - GECCO'02 (New York, USA)*. Ed. by J. Branke et al. Morgan Kaufmann Publishers, 2002, pp. 716–723.
- [228] Q.U. Nguyen. "Examining Semantic Diversity and Semantic Locality of Operators in Genetic Programming". PhD thesis. University College Dublin, 2011.
- [229] Q.U. Nguyen et al. "Semantically-Based Crossover in Genetic Programming: Application to Real-valued Symbolic Regression". In: *Genetic Programming and Evolvable Machines 12* (2011), pp. 91–119.
- [230] K. Krawiec and P. Lichocki. "Approximating geometric crossover in semantic space". In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation - GECCO'09 (Montreal, Quebec, Canada)*. Ed. by G. Raidl et al. Morgan Kaufmann Publishers, 2009, pp. 987–994.
- [231] E. Galván-López et al. "Using semantics in the selection mechanism in Genetic Programming: A simple method for promoting semantic diversity". In: *Proceedings of the 2013 Congress on Evolutionary Computation - CEC'13 (Cancun, Mexico)*. The IEEE Press, 2013, pp. 2972–2979.
- [232] D. R. White et al. "Better GP benchmarks: community survey results and proposals". In: *Genetic Programming and Evolvable Machines 14*(1) (2013), pp. 3–29.
- [233] M. Streeter and L.A. Becker. "Automated discovery of numerical approximation formulae via genetic programming". In: *Genetic Programming and Evolvable Machines 4* (2003), pp. 255–286.
- [234] J. A. Walker and J. F. Miller. "Improving the evolvability of digital multipliers using embedded cartesian genetic programming and product reduction". In: *In Evolvable Systems: From Biology to Hardware* (2005), pp. 131–142.

- [235] Z. Emigdio et al. "Evaluating the effects of local search in genetic programming". In: *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*. Ed. by E. Tantar et al. Springer, Cham., 2014, pp. 213–228.
- [236] *Evostar 2010 Competitions page*. URL: <http://dces.essex.ac.uk/research/evostar/competitions.html>.
- [237] P. G. Espejo, S. Ventura, and F. Herrera. "A survey on the application of genetic programming to classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 40(2) (2010), pp. 121–144.
- [238] T. Loveard and V. Ciesielski. "Representing classification problems in genetic programming". In: *Proceedings of the 2001 Congress on Evolutionary Computation - CEC'01 (Seoul, South Korea)*. The IEEE Press, 2001, pp. 1070–1077.
- [239] M.C.J. Bot and W.B. Langdon. "Application of genetic programming to induction of linear classification trees". In: *LNCS 1802, Proceedings of the 3rd European Conference on Genetic Programming - EuroGP'00 (Edinburgh, Scotland, UK)*. Ed. by R. Poli et al. Springer-Verlag, 2000, 247–258.
- [240] R.E. Marmelstein and G.B. Lamont. "Pattern classification using a hybrid genetic program decision tree approach". In: *Proceedings of the 3rd Annual Conference on Genetic Programming - GP'98 (University of Wisconsin, Madison, Wisconsin, USA)*. Ed. by J.R. Koza et al. Morgan Kaufmann Publishers, 1998, 223–231.
- [241] E. Dufourq and N. Pillay. "A Comparison of Genetic Programming Representations for Binary Data Classification". In: *Proceedings of the 3rd World Congress on Information and Communication Technologies WICT'13 (Hanoi, Vietnam)*. 2013, 134–140.
- [242] C.S. Kuo, T.P. Hong, and C.L. Chen. "Applying genetic programming technique in classification trees". In: *Soft Computing* 11(12) (2007), 1165–1172.
- [243] I. De Falco, A. Della Cioppa, and E. Tarantino. "Discovering interesting classification rules with genetic programming". In: *Applied Soft Computing* 1(4) (2002), 257–269.
- [244] L. Wilkinson, A. Anand, and D. Tuan. "CHIRP: a new classifier based on composite hypercubes on iterated random projections". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 2011 (San Diego, CA)*. ACM New York, 2011, 6–14.
- [245] D.J. Newman et al. "UCI Repository of machine learning databases". In: Irvine, CA, USA: University of California, Department of Information and Computer Science, 2015. URL: <http://archive.ics.uci.edu/ml/>.
- [246] J. Kaiser. "Dealing with missing values in data". In: 5 (1 2014), pp. 42–51.
- [247] I. Kushchu. "Genetic programming and evolutionary generalization". In: *IEEE Transactions on Evolutionary Computation* 6(5) (2002), pp. 431–442.
- [248] G. Dick. "An effective parse tree representation for tartarus". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation - GECCO'13 (Amsterdam, the Netherlands)*. Ed. by C. Blum et al. The ACM Press, 2013, pp. 909–916.
- [249] A. Trenaman. "Concurrent genetic programming, tartarus and dancing agents". In: *LNCS 1598, Proceedings of the 2nd European Conference on Genetic Programming - EuroGP'99 (Goteborg, Sweden)*. Ed. by R. Poli et al. Springer Berlin Heidelberg, 1999, 270–282.
- [250] A. Teller. "The evolution of mental models". In: *Advances in Genetic Programming*. MIT Press, 1994. Chap. 9, 199–219.
- [251] J. R. Koza. "Simultaneous Discovery of Reusable Detectors and Subroutines Using Genetic Programming". In: *Proceedings of the 5th International Conference on Genetic Algorithms - ICGA'93 (University of Illinois at Urbana-Champaign, USA)*. Ed. by S. Forrest. Morgan Kaufmann Publishers, 1993, pp. 295–302.

- [252] J.R. Koza. "Evolution of subsumption using genetic programming". In: *Proceedings of European Conference on Artificial Life*. Ed. by P. Bourguine and F. Varela. Cambridge, MA: MIT Press, 1992.
- [253] Oracle Corporation [US]. *Java*. Version Version 8 update 91. Jan. 19, 2016. URL: <https://java.com/en/download/>.
- [254] Microsoft Corporation]. *Microsoft Excel*. Version Office 2010. Jan. 19, 2016. URL: <http://www.wolfram.com/mathematica/>.
- [255] Wolfram Corporation]. *Mathematica*. Version Version 10. Jan. 19, 2016. URL: <http://www.wolfram.com/mathematica/>.
- [256] S. Jackson. *Research Methods and Statistics: A Critical Thinking Approach*. Cengage Learning, 2015.
- [257] J. A. Walker and J. F. Miller. "Investigating the performance of module acquisition in cartesian genetic programming". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation - GECCO'05 (Washington DC, USA)*. Ed. by H. Beyer et al. Morgan Kaufmann Publishers, 2005, pp. 1649–1656.
- [258] T. K. Ho and M. Basu. "Complexity measures of supervised classification problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(3) (2002), pp. 289–300.
- [259] B. T. Zhang and H. Mühlenbein. "Balancing accuracy and parsimony in genetic programming". In: *Evolutionary Computation* 3(1) (1995), pp. 17–38.
- [260] C. Johnson. *What is Research in Computing Science?* URL: Available: http://www.dcs.gla.ac.uk/~johnson/teaching/research_skills/research.html.
- [261] O. Muntean, L. Dioşan, and M. Oltean. "Best SubTree genetic programming". In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation - GECCO'07 (London, England UK)*. Ed. by D. Thierens and H. Lipson. The ACM Press, 2007, pp. 1667–1673.
- [262] J. V. Hansen. "Genetic programming experiments with standard and homologous crossover methods". In: *Genetic Programming and Evolvable Machines* 4(1) (2003), pp. 53–66.
- [263] T.R. Kuphaldt. *Lessons in Electric Circuits - Vol. IV - Digital*. Open Book Project, 2006, chapter 7, pg 9. URL: <https://www.allaboutcircuits.com/textbook/digital/chpt-7/convert-ing-truth-tables-boolean-expressions/>.
- [264] M. M. Mano, C. R. Kime, and T. Martin. *Logic and computer design fundamentals (Vol. 3)*. 2008.
- [265] M. Collins. "Finding needles in haystacks is harder with neutrality". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation - GECCO'05 (Washington DC, USA)*. Ed. by H. Beyer et al. Morgan Kaufmann Publishers, 2005, 1613–1618.
- [266] M. Črepinšek, S. H. Liu, and M. Mernik. "Exploration and exploitation in evolutionary algorithms: A survey". In: *ACM Computing Surveys (CSUR)* 45(3) (2013), p. 35.
- [267] T. Soule and J. A. Foster. "Effects of code growth and parsimony pressure on populations in genetic programming". In: *Evolutionary Computation* 6(4) (1998), pp. 293–309.
- [268] R.N. Kacker. "Off-line quality control, parameter design, and the Taguchi method". In: (1989), pp. 51–76.
- [269] O. Mersmann et al. "Exploratory landscape analysis". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO'11 (Dublin, Ireland)*. Ed. by N. Krasnogor et al. The ACM Press, 2011, pp. 829–836.
- [270] R. I. McKay et al. "Grammar-based genetic programming: a survey". In: 11 (3–4 2010), pp. 365–396.
- [271] M. O'Neill and C. Ryan. "Grammatical evolution". In: *IEEE Transactions on Evolutionary Computation* 5(4) (2001), pp. 349–358.