



Universidad
de La Laguna

UNIVERSIDAD DE LA LAGUNA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

TESIS DOCTORAL

Ajuste de Parámetros en Algoritmos Evolutivos Secuenciales y Paralelos

Un Enfoque basado en Hiper-heurísticas y
Estrategias de Control de Parámetros

Parameter Setting in Sequential and Parallel Evolutionary Algorithms

An approach based on Hyper-heuristics and
Parameter Control Strategies

Eduardo Manuel Segredo González

Dirigida por Dra. Coromoto León Hernández y Dr. Carlos Segura González

- 2014 -

Dra. **Coromoto León Hernández**, con N.I.F. 78605216-W, profesora titular de la Escuela Técnica Superior de Ingeniería Informática, adscrita al Departamento de Ingeniería Informática y al área de Lenguajes y Sistemas Informáticos, y Dr. **Carlos Segura González**, con N.I.F. 78704244-S

C E R T I F I C A N

Que la presente memoria titulada:

*“Ajuste de Parámetros en Algoritmos Evolutivos Secuenciales y Paralelos.
Un Enfoque basado en Hiper-heurísticas y Técnicas de Control de Parámetros”*

ha sido realizada bajo su dirección por D. **Eduardo Manuel Segredo González**, con N.I.F. 78564242-Z, y constituye su Tesis para optar al grado de Doctor por la Universidad de La Laguna.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna, a 13 de junio de 2014.

Doctoral Dissertation

Parameter Setting in Sequential
and Parallel Evolutionary
Algorithms

An Approach based on Hyper-heuristics and
Parameter Control Strategies

Author

Eduardo Manuel Segredo González

Supervised by

Dr. Coromoto León Hernández

Dr. Carlos Segura González

June 2014

Acknowledgements

I would like to thank my advisors Coromoto León and Carlos Segura for their time and constant support. I would also like to acknowledge Gara Miranda and Casiano Rodríguez for their advice and encouragement.

I dedicate this work to my parents, my sister, and my girlfriend for their encouragement, love, and patience, and in general to every member of my family for their full support during all these years.

I am also grateful to my friends and workmates for all those nice moments that have made things easier.

I would like to thank the Spanish Ministry of Science and Innovation for grant FPU-AP2009-0457. This work was also funded by the European Commission (FEDER) and the Spanish Ministry of Science and Innovation as part of the ‘Plan Nacional de I+D+i’, with contract numbers TIN2005-08818-C04-04, TIN2008-06491-C04-02, and TIN2011-25448. The Government of the Canary Islands also funded this work through project PI2007/015. I am also grateful to the HPC-EUROPA2 Project (number: 228398) supported by the European Commission—Capacities Area—Research Infrastructures. Part of this work made use of the HECTOR facilities, the UK’s national high-performance computing service, which is provided by UOE HPCx Ltd at the University of Edinburgh, Cray Inc and NAG Ltd, and funded by the Office of Science and Technology through the EPSRC’s High End Computing Programme. Finally, I would like to acknowledge the usage of the computational facilities provided by the ‘Servicio de Apoyo Informático a la Investigación’ (SAII), which belongs to the University of La Laguna.

Eduardo Manuel Segredo González

Abstract

In recent decades a wide range of algorithmic approaches have been designed to solve both single-objective and multi-objective optimisation problems. The application of exact methods allows the optimal solutions of these optimisation problems to be obtained. However, exact methods are generally not affordable for many complex applications; as a result, a wide variety of approximate algorithms have been developed with the aim of obtaining high-quality solutions in a reasonable amount of time that might be quite close to the optimal solutions. Among the groups of approximate algorithms, the family of *heuristics* and *meta-heuristics* is worth mentioning.

Meta-heuristics are a set of approximate techniques that have become popular for solving optimisation problems. They are high-level problem-independent strategies that guide a set of heuristics in the search of an optimum. A large variety of meta-heuristics have been proposed in the literature. Among them, Evolutionary Algorithms (EAs) are one of the most popular strategies. They are population-based meta-heuristics inspired by biological evolution. EAs have shown great promise for calculating solutions to large and difficult single-objective and multi-objective optimisation problems. Nevertheless, even with the application of meta-heuristics like EAs, the resolution of some complex optimisation problems could involve using a vast amount of computational resources and time. As a result, the parallelisation of EAs has been proposed in order to speed up the process of obtaining high-quality solutions, and consequently deal with such complex problems. Among the different parallel EAs proposed in the literature, the *island-based* model is one of the most frequently used schemes, because it is suitable for parallel architectures and it can be combined with both single-objective and multi-objective methods.

One of the main drawbacks of EAs is that, for some problems, they exhibit a tendency to converge towards local optima due to diversity loss in populations with a finite size—also called *genetic drift*. Genetic drift is the main reason for the appearance of *premature convergence* in EAs. One of the proposals that has gained some popularity in recent years to address the problem of premature convergence

relies on applying multi-objective schemes to single-objective optimisation problems. Several ways of applying the multi-objective concepts have been devised, with diversity-based Multi-Objective Evolutionary Algorithms (MOEAs) being one of the most promising schemes. In these algorithms, a metric of the diversity introduced by each individual is used as an auxiliary objective, whereas the original objective function of the optimisation problem being solved is kept.

Besides the problem of premature convergence, finding the appropriate setting for an EA remains one of the persistent challenges for Evolutionary Computation (EC). In order to completely define a configuration of an EA several *symbolic* parameters, such as the variation or parent selection operators, and different *numeric* parameters, such as the mutation and crossover rates, must be set. In general, the performance of an EA and, consequently, the quality of the resulting solutions, is highly dependent on these parameters. As a result, it is essential that the parameters of an EA be suitably determined. Parameter setting strategies are commonly divided into two categories: parameter *tuning* and parameter *control*. In parameter tuning the objective is to identify the best set of values for the parameters of a given EA, which is then executed using these values, which remain constant for the duration of the run. In contrast, the aim of parameter *control* is to design control strategies that select the most suitable values for the parameters at each stage of the search process while the algorithm is being executed. In recent years, parameter tuning approaches have exhibited some disadvantages with respect to parameter control methods, so recent research developments have focused on control strategies, since it has been theoretically and empirically demonstrated that the most appropriate parameter values change depending on the stage of the optimisation procedure.

Thus, in this dissertation the main aim is twofold. Firstly, to design and study diversity-based MOEAs as methods for dealing with premature convergence in EAs. In this regard, several diversity metrics with parameters are proposed as novel auxiliary objectives, as well as a novel diversity-based survivor selection mechanism which preserves diversity in a population of individuals. Secondly, to design several parameter control strategies with the final aim of being able to simultaneously adapt numeric and symbolic parameters of an EA. These parameter control schemes are based on the usage of *Fuzzy Logic Controllers* and *Hyper-heuristics*, and they are applied, respectively, to adapt numeric and symbolic parameters belonging to the proposed diversity-based MOEAs. Additionally, in order to more quickly obtain high-quality solutions and to enable the usage of some of these techniques in parallel environments, they are integrated with an island-based model. Finally, the validation of the different proposals is carried out by measuring their performance over a set of well-known benchmark problems and real-world complex applications.

Contents

Acknowledgements	i
Abstract	iii
I Foundations, Background, and Contributions	1
1 Introduction	3
1.1 Optimisation Problems	3
1.1.1 Single-objective Optimisation Problems	4
1.1.2 Multi-objective Optimisation Problems	5
1.2 Optimisation Schemes	9
1.2.1 Complexity Theory	10
1.2.2 Exact Algorithms	13
1.2.3 Approximate Algorithms	15
1.2.4 Premature Convergence	19
1.2.5 Parameter Setting	21
1.2.6 Performance Measurement	25
1.3 Parallel and Distributed Computing	26
1.3.1 Parallel Architectures	28
1.3.2 Parallel Programming Models	33
1.3.3 Performance Measurement in Parallel Applications	36
1.4 Research Questions	39
1.5 Main Contributions	40
1.6 Synopsis	43
2 Evolutionary Algorithms	45
2.1 A Brief Historical Introduction	45
2.2 Basic Concepts for Evolutionary Algorithms	49

2.2.1	Parent Selection Mechanisms	52
2.2.2	Crossover Operators	54
2.2.3	Mutation Operators	58
2.2.4	Survivor Selection Methods	60
2.3	Single-objective Evolutionary Algorithms	61
2.3.1	Genetic Algorithms	61
2.4	Multi-objective Evolutionary Algorithms	64
2.4.1	Dominance-based Multi-objective Evolutionary Algorithms . .	65
2.5	Memetic Algorithms	70
2.6	Parallel Evolutionary Algorithms	73
2.6.1	Island-based Model	76
3	Background in MOEAs for Single-objective Optimisation	79
3.1	Diversity-based Multi-objective Evolutionary Algorithms	80
3.1.1	Encoding-independent Measures	81
3.1.2	Genotypic and Phenotypic Measures	82
3.1.3	Behavioural Measures	83
3.2	Multi-objectivisation	84
4	State of the Art in Parameter Setting in Evolutionary Algorithms	89
4.1	Parameter Tuning	89
4.2	Parameter Control	93
4.3	Synergy between Fuzzy Systems and Evolutionary Algorithms	95
4.3.1	Adapting Evolutionary Algorithms using Fuzzy Logic Con- trollers	99
4.4	Hyper-heuristics as Parameter Control Methods	104
II	Algorithmic Proposals	111
5	Advances in Diversity-based MOEAs	113
5.1	Diversity-based Objectives with Parameters	113
5.2	Diversity-based Survivor Selection Scheme	116
6	Innovations in Parameter Control Schemes	121
6.1	Fuzzy Logic Controllers with Multiple Rule Bases	121
6.2	Hyper-heuristics	129
6.3	Dynamic-mapped Island-based Model	131
6.4	Hybrid Control Scheme based on FLCs and Hyper-heuristics	132

III Validation of the Algorithmic Proposals: Benchmark Problems and Complex Applications 135

7	Benchmark Problems	137
7.1	Formal Definition	138
7.2	Optimisation Schemes	140
7.2.1	Single-objective Genetic Algorithms	140
7.2.2	Diversity-based Multi-objective Evolutionary Algorithms	141
7.3	Parameter Control Schemes	141
7.3.1	Fuzzy Logic Controllers and Hyper-heuristics	142
7.3.2	Self-adaptation	142
7.3.3	Hybrid Control Scheme based on Fuzzy Logic Controllers and Hyper-heuristics	143
7.4	Experimental Evaluation and Discussion	144
7.4.1	Performance of the Diversity-based Multi-objective Evolutionary Algorithm	144
7.4.2	On the Application of Diversity-based Objectives with Parameters	148
7.4.3	Improving the Robustness of the Diversity-based Multi-objective Evolutionary Algorithm	152
7.4.4	Analysis of the Parameter Control Schemes Over a Short Evaluation Timeframe	154
7.4.5	Analysis of the Parameter Control Schemes Over a Long Evaluation Timeframe	157
7.4.6	Comparison between Short and Long Evaluation Periods	159
7.4.7	Comparing Parameter Control and Parameter Tuning	160
7.4.8	Analysis of the Hybrid Control Scheme based on Fuzzy Logic Controllers and Hyper-heuristics	164
7.4.9	Analysis of the Diversity-based Survivor Selection Operator	168
8	Antenna Positioning Problem	175
8.1	Formal Definition	176
8.2	Optimisation Schemes	177
8.2.1	Single-objective Iterated Local Search	177
8.2.2	Diversity-based Multi-Objective Evolutionary Algorithms	180
8.2.3	Parallel Homogeneous Island-based Models	180
8.3	Experimental Evaluation and Discussion	181
8.3.1	On the Comparison of Diversity-based Objectives	181

8.3.2	Diversity-based Multi-Objective Evolutionary Algorithms vs. Single-objective Iterated Local Search	184
8.3.3	Analysing the Robustness of the Homogeneous Island-based Model	187
8.3.4	Analysing the Scalability of the Homogeneous Island-based Model	189
9	Frequency Assignment Problem	191
9.1	Formal Definition	194
9.2	Optimisation Schemes	196
9.2.1	Evolutionary Algorithm with Increasing Population Size . . .	196
9.2.2	Diversity-based Multi-objective Memetic Algorithm and Multi-objectivisation by Aggregation	198
9.2.3	Genetic Operators	199
9.2.4	Individual Learning Strategy	200
9.3	Parameter Control Schemes	202
9.3.1	Fuzzy Logic Controllers and Hyper-heuristics	203
9.3.2	Dynamic-mapped Island-based Model	203
9.4	Experimental Evaluation and Discussion	204
9.4.1	On the Comparison of Sequential Memetic Algorithms	205
9.4.2	Long-term Analysis of the Sequential Memetic Algorithms . .	209
9.4.3	Analysing the Robustness of the Dynamic-mapped Island-based Model	211
9.4.4	Analysing the Scalability of the Dynamic-mapped Island-based Model	215
9.4.5	Adapting the parameter p_m of the Neighbour-based Mutation Operator	219
9.4.6	Adapting the parameter R of the Neighbour-based Mutation Operator	224
10	Two-dimensional Packing Problem	231
10.1	Formal Definition	232
10.2	Optimisation Schemes	234
10.2.1	Evolutionary Algorithm with Increasing Population Size . . .	234
10.2.2	Diversity-based Multi-objective Memetic Algorithms and Multi-objectivisation by Aggregation	234
10.2.3	Genetic Operators	236
10.2.4	Individual Learning Strategy	236
10.2.5	Parallel Homogeneous Island-based Models	238

10.3	Parameter Control Schemes	238
10.3.1	Dynamic-mapped Island-based Model	239
10.4	Experimental Evaluation and Discussion	239
10.4.1	Comparison of the Sequential Memetic Algorithms	240
10.4.2	Analysis of the Parallel Homogeneous Island-based Models . .	243
10.4.3	On the comparison of the Memetic Algorithms based on the Non-Dominated Sorting Genetic Algorithm II and the Strength Pareto Evolutionary Algorithm 2	246
10.4.4	Analysing the Robustness of the Dynamic-mapped Island- based Model	250
10.4.5	Analysing the Scalability of the Dynamic-mapped Island-based Model	252
IV	Conclusions and Future Lines of Research	261
V	Appendices	267
A	List of Publications	269
B	Fuzzy Rule Bases	273
B.1	Fuzzy Rule Bases for the FUZZY-A and FUZZY-A-TSK Approaches . .	273
B.2	Fuzzy Rule Bases for the FUZZY-B and FUZZY-B-TSK Approaches . .	281
	Bibliography	285

List of Algorithms

1	Generic pseudocode for an evolutionary algorithm	50
2	Pseudocode of the different single-objective genetic algorithms	63
3	Pseudocode of the Non-Dominated Sorting Genetic Algorithm II . . .	67
4	Pseudocode of the Strength Pareto Evolutionary Algorithm 2	69
5	Generic pseudocode for a memetic algorithm	72
6	Pseudocode of the novel diversity-based survivor selection scheme . .	119
7	Pseudocode of the novel fuzzy logic controller	123
8	Generic pseudocode for an iterated local search	178
9	Pseudocode of the Evolutionary Algorithm with Increasing Popula- tion Size	197
10	Pseudocode of the memetic algorithm based on the Non-Dominated Sorting Genetic Algorithm II	199
11	Pseudocode of the individual learning strategy designed for the Fre- quency Assignment Problem	201
12	Pseudocode of the memetic algorithm based on the Strength Pareto Evolutionary Algorithm 2	235

List of Figures

1.1	Graphical representation of the Pareto dominance concept	7
1.2	Relationship between different complexity classes ($P \neq NP$)	12
1.3	Classification of optimisation methods	14
1.4	General architecture of a shared-memory system	29
1.5	Architecture of a uniform memory access system with m processors and n cores each	30
1.6	Architecture of a non-uniform memory access system with m proces- sors and n cores each	31
1.7	General architecture of a distributed-memory system	32
2.1	Flow chart representing an evolutionary algorithm generation	51
2.2	Parent selection based on a deterministic binary tournament	54
2.3	Operation of the Uniform Crossover	55
2.4	Operation of the One Point Crossover	55
2.5	Operation of the Two-Dimensional Sub-String Crossover	58
2.6	Operation of the bit flip mutation	59
4.1	Three-layer hierarchy of parameter tuning	90
4.2	Taxonomy of parameter setting in evolutionary algorithms	94
4.3	General architecture of a fuzzy logic controller	100
4.4	Membership functions for the linguistic variable <i>fan speed</i>	101
4.5	General framework of a hyper-heuristic	108
5.1	Behaviour of the Non-Dominated Sorting Genetic Algorithm II and the Strength Pareto Evolutionary Algorithm 2 combined with diversity- based objectives without threshold (left-hand side) and with threshold (right-hand side)	115
5.2	Simple function where diversity preservation problems arise	117
6.1	Membership functions for the input and output variables	125

6.2	Multi-level architecture of the novel hybrid parameter control scheme	133
7.1	Chromosome representing an individual for the self-adaptive schemes	143
7.2	Mean of the original objective value achieved by the parameter control methods and by fixed values of the threshold ratio	161
7.3	Median of the original objective value achieved for different threshold ratios	170
7.4	Box plots for the worst-behaved executions considering different stopping criteria	172
7.5	Box plots considering a population size equal to 500 individuals . . .	174
8.1	Evolution of the mean original objective value for different diversity-based objective functions	182
8.2	Run-length distributions for quality level L1	185
8.3	Run-length distributions for quality level L2	186
8.4	Evolution of the mean original objective value for the homogeneous island-based models executed with 4 islands	187
8.5	Run-length distributions for the homogeneous island-based models executed with 4 islands	188
8.6	Run-length distributions for the homogeneous island-based models executed with 4, 8, and 16 islands	189
9.1	Generating a new neighbour by reassigning the frequencies belonging to a certain sector	202
9.2	Evolution of the mean original objective value for the different memetic approaches	206
9.3	Box plots for the best configurations of the memetic approaches . . .	209
9.4	Run-length distributions for the best configurations of the memetic approaches	210
9.5	Evolution of the mean original objective value for the dynamic-mapped island-based model executed with 4 islands	212
9.6	Box plots for the dynamic-mapped island-based model executed with 4 islands	213
9.7	Run-length distributions for the dynamic-mapped island-based model executed with 4 islands	215
9.8	Box plots for the best (left-hand side) and worst (right-hand side) migration stages for the Seattle instance – 4, 8, 16, and 32 islands . .	218
9.9	Box plots for the best (left-hand side) and worst (right-hand side) migration stages for the Denver instance – 4, 8, 16, and 32 islands . .	218

10.1	Assignment of the objective function value for the Two-Dimensional Packing Problem	233
10.2	Generation of new neighbours by the learning process	237
10.3	Evolution of the mean original objective value for the different memetic approaches	241
10.4	Box plots for the homogeneous island-based models executed with 4 islands for 12 hours	243
10.5	Box plots for the homogeneous island-based models executed with 4 islands and considering a fixed computational effort	244
10.6	Run-length distributions for the homogeneous island-based models executed with 4 islands for 12 hours	245
10.7	Evolution of the mean original objective value for the dynamic-mapped island-based model executed with 4 islands	250
10.8	Box plots for the dynamic-mapped island-based model executed with 4 islands	251
10.9	Box plots for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the first instance	254
10.10	Box plots for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the second instance	254
10.11	Run-length distributions for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the first instance	255
10.12	Run-length distributions for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the second instance	256
10.13	Box plots for the dynamic-mapped island-based model with the best migration stage for the first and second instances	258
10.14	Long-term evolution of the mean original objective value for the dynamic-mapped island-based models	259

List of Tables

2.1	Implementation details for the canonical evolutionary algorithms . . .	47
2.2	Evolutionary process versus resolution of an optimisation problem . .	49
5.1	Assignment of the non-domination rank by a diversity-based optimiser using the Non-Dominated Sorting Genetic Algorithm II	118
6.1	Rule bases designed for FUZZY-A and FUZZY-A-TSK (left-hand side); and for FUZZY-B and FUZZY-B-TSK (right-hand side)	127
7.1	Definition of the F1–F11 benchmark functions	138
7.2	Properties of the F1–F11 benchmark functions	139
7.3	Definition of the F12–F19 hybrid composition benchmark functions .	139
7.4	Best configurations for the single-objective and the diversity-based approaches	146
7.5	Median of the error achieved by the best configurations of both optimisation schemes	147
7.6	Percentage of evaluations saved by the best configuration of the diversity-based multi-objective evolutionary algorithm	148
7.7	Statistical comparison between different values of the threshold ratio	148
7.8	Number of evaluations required by different threshold ratios to attain a given quality level	149
7.9	Statistical comparison among different threshold values for each optimisation stage – F4 benchmark function	150
7.10	Statistical comparison among different threshold values for each optimisation stage – F5 benchmark function	150
7.11	Statistical comparison among different threshold values for each optimisation stage – F11 benchmark function	151
7.12	Number of evaluations required by the hyper-heuristic to attain a given quality level	152

7.13	Resources assigned by the hyper-heuristic for benchmark problems with no variability in the threshold ratio	153
7.14	Statistical comparison between the hyper-heuristic and different low-level configurations	154
7.15	Parameterisation of the diversity-based multi-objective evolutionary algorithm	155
7.16	Parameterisation of the HH-ELI and HH-PROB hyper-heuristics	155
7.17	Parameterisation of the FUZZY-A and FUZZY-B fuzzy logic controllers	155
7.18	Mean original objective value for the best parameter control schemes after $5 \cdot 10^5$ evaluations	156
7.19	Mean original objective value for the best parameter control schemes after $2.5 \cdot 10^6$ evaluations	158
7.20	Maximum number of evaluations required to attain the specified quality level and percentage of evaluations saved by each approach	163
7.21	Parameterisation of the diversity-based multi-objective evolutionary algorithm	165
7.22	Parameterisation of the HH-PROB hyper-heuristic	165
7.23	Parameterisation of the FUZZY-A fuzzy logic controller	165
7.24	Values for the parameters of the best fixed configuration	166
7.25	Median of the error attained by the hybrid control scheme and by the best fixed configuration	167
7.26	Probability of selecting the best individuals	171
7.27	Evaluations saved by not using the diversity-based survivor selection scheme	171
7.28	Statistical comparison considering different stopping criteria	173
8.1	Statistical comparison among different diversity-based objectives . . .	183
8.2	Amount of additional resources invested by the diversity-based multi-objective evolutionary algorithm with respect to the iterated local search	186
9.1	Statistical comparison among different memetic approaches for the Seattle instance	207
9.2	Statistical comparison among different memetic approaches for the Denver instance	207
9.3	Statistical comparison among different configurations of the variation stage	208
9.4	Statistical comparison among different migration stages for the Seattle instance – 4 islands	214

9.5	Statistical comparison among different migration stages for the Denver instance – 4 islands	214
9.6	Speedup factors achieved by the dynamic-mapped island-based model for the Seattle instance – 4 islands	215
9.7	Speedup factors achieved by the dynamic-mapped island-based model for the Denver instance – 4 islands	215
9.8	Speedup factors achieved by the dynamic-mapped island-based model for the Seattle instance – 8, 16, and 32 islands	216
9.9	Speedup factors achieved by the dynamic-mapped island-based model for the Denver instance – 8, 16, and 32 islands	216
9.10	Statistical comparison among different migration stages for the Seattle instance – 32 islands	217
9.11	Statistical comparison among different migration stages for the Denver instance – 32 islands	217
9.12	Parameterisation of the diversity-based multi-objective memetic algorithm for different values of the parameter p_m	219
9.13	Parameterisation of the hyper-heuristics HH-ELI and HH-PROB to control the parameter p_m	219
9.14	Parameterisation of the fuzzy logic controllers FUZZY-A, FUZZY-B, FUZZY-A-TSK, and FUZZY-B-TSK to control the parameter p_m	220
9.15	Control and tuning of the parameter p_m – Seattle instance	221
9.16	Control and tuning of the parameter p_m – Denver instance	222
9.17	Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter p_m – Seattle instance	223
9.18	Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter p_m – Denver instance	223
9.19	Parameterisation of the diversity-based multi-objective memetic algorithm for different values of the parameter R	224
9.20	Parameterisation of the hyper-heuristics HH-ELI and HH-PROB to control the parameter R	224
9.21	Parameterisation of the fuzzy logic controllers FUZZY-A, FUZZY-B, FUZZY-A-TSK, and FUZZY-B-TSK to control the parameter R	225
9.22	Control and tuning of the parameter R – Seattle instance	227
9.23	Control and tuning of the parameter R – Denver instance	228
9.24	Number of fixed configurations outperformed by the parameter control approaches adapting the parameter R – Seattle instance	229
9.25	Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter R – Denver instance	229

10.1	Statistical comparison among different memetic approaches for the first instance	242
10.2	Statistical comparison among different memetic approaches for the second instance	242
10.3	Original objective function for the memetic algorithms considering the first instance	248
10.4	Original objective function for the memetic algorithms considering the second instance	249
10.5	Statistical tests for the dynamic-mapped island-based model – 16 islands – 5 hours – First instance	252
10.6	Statistical tests for the dynamic-mapped island-based model – 32 islands – 5 hours – First instance	252
10.7	Statistical tests for the dynamic-mapped island-based model – 4 islands – 11.5 hours – Second instance	253
10.8	Statistical tests for the dynamic-mapped island-based model – 8, 16, 32 islands – 11.5 hours – Second instance	253
10.9	Speedup factors for the dynamic-mapped island-based model with the best and worst migration stages – First instance	255
10.10	Speedup factors for the dynamic-mapped island-based model with the best and worst migration stages – Second instance	256
10.11	Speedup factors for the dynamic-mapped island-based model with the best migration stage for both instances	258
B.1	Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation	273
B.2	Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation	274
B.3	Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation	275
B.4	Rule base number 6 to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation	276
B.5	Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters	277

B.6	Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters	278
B.7	Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters	279
B.8	Rule base number 6 to control the threshold ratio th of the diversity-based objectives with parameters	280
B.9	Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters	281
B.10	Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters	282
B.11	Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters	283
B.12	Rule base number 6 to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters	284

List of Acronyms

2DPP	Two-Dimensional Packing Problem
ABC	Artificial Bee Colony
ACO	Ant Colony Optimisation
ADI	Average Distance to all Individuals
ADI-THR	Average Distance to all Individuals with Threshold
AFP	Automatic Frequency Planning
AIS	Artificial Immune System
ALL	All to All Connected Topology
ANOVA	Analysis of Variance
APP	Antenna Positioning Problem
AX	Arithmetical Crossover
BBOB	Black-Box Optimization Benchmarking
BER	Bit Error Rate
BS	Base Station
BSTL	Base Station Transmitters Location
CA	Cultural Algorithm
CAP	Channel Assignment Problem
CEA	Coevolutionary Algorithm
C/I	Carrier-to-Interference
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
COW	Clusters of Workstations
CPU	Central Processing Unit
CV	Coefficient of Variation
DBI	Distance to Best Individual
DBI-THR	Distance to Best Individual with Threshold
DCN	Distance to Closest Neighbour
DCN-THR	Distance to Closest Neighbour with Threshold
DCN-REF	Distance to Closest Neighbour based on Reference Set
DE	Differential Evolution

DNA	Deoxyribonucleic Acid
DYN	Dynamic-mapped Island-based Model
EA	Evolutionary Algorithm
EAIPS	Evolutionary Algorithm with Increasing Population Size
EC	Evolutionary Computation
EDA	Estimation of Distribution Algorithm
ELI-M	Elitist Migration Scheme
ELI-R	Elitist Ranking Replacement Scheme
EP	Evolutionary Programming
ER	Evolutionary Robotics
ES	Evolution Strategies
FAP	Frequency Assignment Problem
FLC	Fuzzy Logic Controller
FRBS	Fuzzy Rule-based System
GA	Genetic Algorithm
GCC	GNU Compiler Collection
GECCO	Genetic and Evolutionary Computation Conference
GEN-S	Elitism-based Generational Survivor Selection
GLS	Guided Local Search
GMSK	Gaussian Minimum Shift Keying
GP	Genetic Programming
GPU	Graphics Processing Unit
GRASP	Greedy Randomised Adaptive Search Procedure
GSM	Global System for Mobile Communications
HAM-R	Hamming-based Replacement Scheme
IA	Interactive Analysis
IBEA	Indicator-Based Evolutionary Algorithm
ILS	Iterated Local Search
IX	Interference-based Crossover
JSP	Job-Shop Scheduling Problem
LAN	Local Area Network
LS	Local Search
MA	Memetic Algorithm
METCO	Metaheuristic-based Extensible Tool for Cooperative Optimisation
μ GA ²	Micro GA-MOEA
MIFAP	Minimum Interference Frequency Assignment Problem
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data

MOEA	Multi-Objective Evolutionary Algorithm
MOFAP	Minimum Order Frequency Assignment Problem
MOGA	Multi-Objective Optimisation Genetic Algorithm
MOP	Multi-objective Optimisation Problem
MSFAP	Minimum Span Frequency Assignment Problem
MPI	Message Passing Interface
NM	Neighbour-based Mutation
NOW	Networks of Workstations
NPGA	Niched Pareto Genetic Algorithm
NPGA2	Niched Pareto Genetic Algorithm 2
NSGA	Non-Dominated Sorting Genetic Algorithm
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NUMA	Non-Uniform Memory Access
OpenMP	Open Multi-Processing
OPX	One Point Crossover
PAES	Pareto Archived Evolution Strategy
PBX- α	Parent-centric Blend Crossover
PM	Polynomial Mutation
POSIX	Portable Operative System Interface
PSO	Particle Swarm Optimisation
PVM	Parallel Virtual Machine
QoS	Quality of Service
REVAC	Relevance Estimation and Value Calibration
RING	Unidirectional Ring Topology
RLD	Run-Length Distribution
RND	Radio Network Design
RND-M	Random Migration Scheme
RND-R	Random Replacement Scheme
R&S	Ranking and Selection
RW-S	Replace Worst Survivor Selection
SA	Simulated Annealing
SBX	Simulated Binary Crossover
SD	Standard Deviation
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SS	Scatter Search
SSX	Two-Dimensional Sub-String Crossover
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2

SPMD	Single-Program Multiple-Data
SPO	Sequential Parameter Optimisation
SS-S	Steady-State Survivor Selection
SUS	Stochastic Universal Sampling
TS	Tabu Search
TSK	Takagi-Sugeno-Kang
TSP	Travelling Salesman Problem
UM	Uniform Mutation
UMA	Uniform Memory Access
UMD	Uniform Mutation with Domain Information
UX	Uniform Crossover
VEGA	Vector Evaluated Genetic Algorithm
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
WAN	Wide Area Network

Part I

Foundations, Background, and Contributions

Introduction

This chapter presents the basic concepts and the nomenclature that will be used throughout the rest of this dissertation. Particularly, the description and formal definition of an optimisation problem in both the single-objective and the multi-objective fields are given. Moreover, a taxonomy which classifies the optimisation schemes available for dealing with optimisation problems is provided. Since in this thesis some approaches are enabled for use in parallel environments, some basic notions on parallel and distributed computing are also described. Finally, the set of research questions posed at the beginning of this work are exposed, together with the main contributions and the synopsis of the rest of this thesis.

1.1 Optimisation Problems

Optimisation is one of the most important topics for a wide range of fields in science and technology, such as computer science, operational research, and artificial intelligence [75], as well as in other areas like finance, business, and medicine [209]. Given an optimisation problem, there exist different feasible solutions, and therefore the main aim of optimisation is to find the best possible solution to a problem from among all feasible solutions. Hence, solving an optimisation problem requires finding said best solution by taking into consideration certain objectives while at the same time satisfying certain constraints.

In order to obtain a solution for an optimisation problem, several decisions involving the problem domain must be made. For example, consider a variant of the knapsack problem [225, 233] in which a set of items, each with its corresponding profit and

weight, is given. The optimisation problem consists of deciding which items from the set of candidate items will be stored into the knapsack so that the global profit is maximised, while at the same time satisfying the constraint of maximum weight that the knapsack is able to support. Different decisions are represented by a set of *decision variables*, which is usually called the *decision vector*, and decisions are carried out by assigning values to each variable belonging to the decision vector. Continuing with the example of the knapsack problem, a possible decision vector could contain a set of ones and zeros indicating whether an item is selected to be stored or not.

The quality of a decision vector or solution has to be determined using certain criteria, which are expressed as a computable function of the decision variables. Considering this function, optimisation problems can be classified as Single-objective Optimisation Problems or Multi-objective Optimisation Problems (MOPs). In single-objective optimisation problems a scalar function is defined, while in the multi-objective field a vector or multiple objective function is applied. In the knapsack example, if a unique knapsack is considered, a scalar function is defined, and therefore a single-objective variant of the knapsack problem is addressed. This function might compute, for instance, the sum of the profits of the stored items. Thus, for a given feasible solution which satisfies the maximum weight constraint of the knapsack, the higher the value of this function, the better the quality of the solution.

1.1.1 Single-objective Optimisation Problems

The *single-objective optimisation problem*, as described in Definition 1 [70], can be handled using a wide variety of optimisation schemes.

Definition 1 *A general single-objective optimisation problem is defined as minimising—or maximising— $f(x)$ subject to $g_i(x) \leq 0$, $i = \{1, \dots, m\}$ and $h_j(x) = 0$, $j = \{1, \dots, p\}$ $x \in \Omega$.*

A solution or decision vector $x = (x_1, \dots, x_n)$, which is an n -dimensional vector belonging to some universe Ω , minimises—or maximises—the scalar $f(x)$. It is important to note that x can be a vector of continuous or discrete decision variables, whereas the scalar function f can also be continuous or discrete. Moreover, $g_i(x) \leq 0$ and $h_j(x) = 0$ are inequality and equality constraints that must be satisfied while optimising—minimising or maximising— $f(x)$. Hence, Ω contains all feasible solutions x that satisfy an evaluation of $f(x)$ and its constraints.

The process of finding the global optimum—or global optima—of any function is referred to as *Global Optimisation*. From now on, and without loss of generality, a single-objective optimisation problem to be minimised will be considered. Taking this into account, the global minimum of a single-objective problem [16] is presented in Definition 2.

Definition 2 *Given a function $f : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq \emptyset$ for $x^* \in \Omega$ the value $f^* \triangleq f(x^*) > -\infty$ is called a **global minimum** if and only if*

$$\forall x \in \Omega : f(x^*) \leq f(x) \quad (1.1)$$

Thus, x^* is called the *global minimum solution*, f is denoted as the *objective function*, and Ω is the set that determines the *feasible region*. Note that the inequality in Equation 1.1 indicates that several global minimum solutions could exist for a single-objective optimisation problem. The aim of obtaining the global minimum solution—or solutions—is therefore known as the *global optimisation problem*.

1.1.2 Multi-objective Optimisation Problems

A Multi-objective Optimisation Problem—also called *multi-criteria optimisation*, *multi-performance*, or *vector optimisation problem*—can be defined [272] as the problem of finding “a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria that are usually in conflict with one another. Hence, the term ‘optimise’ means finding a solution that would give the values of all the objective functions acceptable to the decision maker”.

In order to give the formal definition of a MOP, the Definition 1 shown in Section 1.1.1 for single-objective problems must be expanded to consider several objective functions instead of dealing with a unique objective function. Thus, when a MOP is optimised, a set of solutions must be found by using the concepts established by the *Pareto Optimality Theory* [103]. All the objective functions must be optimised simultaneously, and this fact could involve minimising every objective function, maximising every objective function or any possible combination thereof. The mathematical formulation of a MOP used in this dissertation [69] is shown in Definition 3.

Definition 3 A general **MOP** is defined as minimising—or maximising— $F(x) = (f_1(x), \dots, f_k(x))$ subject to $g_i(x) \leq 0$, $i = \{1, \dots, m\}$ and $h_j(x) = 0$, $j = \{1, \dots, p\}$ $x \in \Omega$.

A solution or decision vector $x = (x_1, \dots, x_n)$, which is an n -dimensional vector belonging to some universe Ω , minimises—or maximises—the components of a vector $F(x)$. Note that $g_i(x) \leq 0$ and $h_j(x) = 0$ are inequality and equality constraints that must be satisfied while optimising—minimising or maximising— $F(x)$. Moreover, Ω contains all feasible solutions x that satisfy an evaluation of $F(x)$ and its constraints.

It can therefore be observed that a MOP consists of a set of k objectives corresponding to the k objective functions, $m + p$ constraints on the objective functions, and n decision variables. The k objective functions can be continuous or discrete and linear or non-linear. In addition, every decision variable of a decision vector x can also be continuous or discrete. Finally, it is important to remark that the function F maps a vector of decision variables $x = (x_1, \dots, x_n)$ to an output objective vector $u = (u_1, \dots, u_k)$, where $u_i = f_i(x)$. In other words, the function F establishes a mapping between the decision space and the objective space.

Since for a MOP several objective functions are defined, the notions of optimality have to be altered. The most frequently used notions of optimality were first proposed by Francis Ysidro Edgeworth [101], and afterwards generalised by Vilfredo Pareto [276]. Although some authors use the term *Edgeworth-Pareto optimality* [318], the most widely accepted term is *Pareto optimality* [69]. Pareto optimality constitutes by itself the origin of research in multi-objective optimisation. In fact, the main aim of multi-objective optimisation is to identify the set of Pareto optimal solutions.

In order to define a Pareto optimal solution, the concept of *Pareto dominance* must first be discussed. Pareto dominance is formalised in Definitions 4 and 5. From now on, and without loss of generality, a MOP where every objective function is minimised will be considered.

Definition 4 An objective vector $u = (u_1, \dots, u_k)$ **dominates** another objective vector $v = (v_1, \dots, v_k)$ (this is denoted by $u \preceq v$) if and only if $\forall i \in \{1, \dots, k\} : u_i \leq v_i \wedge \exists i \in \{1, \dots, k\} : u_i < v_i$.

In words, an objective vector u dominates another vector v if and only if u performs better than v in at least one component and, at the same time, u performs as well as v in the remaining components.

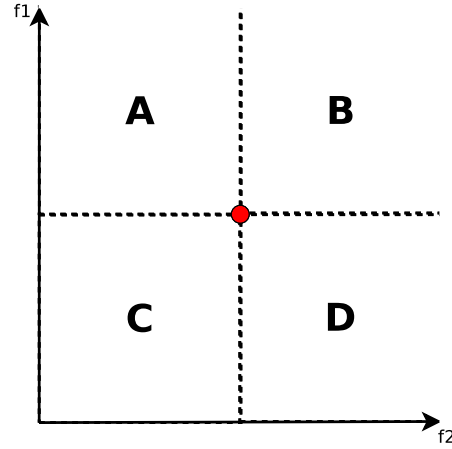


Figure 1.1: Graphical representation of the Pareto dominance concept

Definition 5 An objective vector $u = (u_1, \dots, u_k)$ **strictly dominates** another objective vector $v = (v_1, \dots, v_k)$ (this is denoted by $u \prec v$) if and only if $\forall i \in \{1, \dots, k\} : u_i < v_i$

In order to distinguish between both kinds of dominances, some authors refer to the 4 Definition of dominance as *weak dominance*. In this dissertation, the term dominance always refers to weak dominance.

Figure 1.1 describes the concept of Pareto dominance by representing the objective space. Note that $k = 2$ objective functions are defined for this example. The point represents the corresponding objective vector—the values of the objective functions—of a certain solution. If minimisation is considered for both objective functions, three possible cases arise:

- Area B includes all objective vectors dominated by the objective vector represented by the point, which has a better—lower—value assigned for at least one objective.
- In contrast, area C includes all the objective vectors that dominate the objective vector represented by the point, which has a worse—higher—value assigned for at least one objective.
- Finally, the remaining areas—A and D—include objective vectors that do not dominate and are not dominated by the objective vector represented by the point, which has a better value assigned for one objective and a worse value for the other objective.

It is worth mentioning that the aforementioned three cases change depending on the combinations of minimisation and maximisation selected for each of the two objective functions.

Definition 6 *A solution $x \in \Omega$ is **Pareto optimal** with respect to Ω if and only if $\nexists y \in \Omega: v = F(y) \preceq u = F(x)$*

A Pareto optimal solution x is optimal in the sense that no other solution in the feasible region is able to outperform x when all objectives are considered, i.e. there does not exist any solution y in the feasible region that performs better than x in at least one objective, and performs as well as x for the remaining objectives. Pareto optimal solutions are also known as *non-inferior*, *admissible*, *efficient*, or *non-dominated* solutions, and their corresponding objective vectors are usually called *non-dominated vectors* [69]. It is common for practitioners to refer to the set of solutions obtained by an optimisation scheme as the set of non-dominated solutions. This means that the optimisation scheme has not been able to find any solution that dominates them. However, if the feasible region were better explored, a solution that dominates any of those solutions could probably be found.

The set that includes all Pareto optimal solutions belonging to the feasible region is called the *Pareto optimal set*, whereas the *Pareto front*—also called the *Pareto optimal front* or the *Pareto frontier*—includes every objective vector obtained from every Pareto optimal solution that belongs to the Pareto optimal set. The Pareto front is obtained by applying the function F to every solution in the Pareto optimal set, i.e. the Pareto front is the image in the objective space of the Pareto optimal set. This transformation is used to identify the Pareto optimal solutions. The formal definitions of the Pareto optimal set and the Pareto front are as follows.

Definition 7 *For a given MOP the **Pareto optimal set**, \mathcal{P}^* , is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \nexists y \in \Omega : v = F(y) \preceq u = F(x)\} \quad (1.2)$$

Definition 8 *For a given MOP and a Pareto optimal set, \mathcal{P}^* , the **Pareto front**, \mathcal{PF}^* , is defined as:*

$$\mathcal{PF}^* := \{u = F(x) \mid x \in \mathcal{P}^*\} \quad (1.3)$$

The definition of a global optimum in the multi-objective field is not as trivial as in the case of single-objective optimisation. This is due to the fact that the best compromise solution, which is selected from \mathcal{P}^* , depends on the particular features of the MOP being solved and on the preferences of the human decision maker. Hence, there is no universally accepted definition for the MOP global optimisation problem. However, a MOP global minimum can be formally described, as Definition 9 shows [69].

Definition 9 *Given a function $F : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^k$, $\Omega \neq \emptyset$, $k \geq 2$, the set $\mathcal{PF}^* \triangleq \{F(x^*) = (f_1(x^*) > -\infty, \dots, f_k(x^*) > -\infty) \mid x^* \in \mathcal{P}^*\}$ is called the **global minimum** if and only if*

$$\forall x \in \Omega : F(x^*) \preceq F(x) \quad (1.4)$$

In fact, every solution from \mathcal{P}^* satisfies Equation 1.4 because they are Pareto optimal solutions. Thus, \mathcal{P}^* is called the *global minimum solution set*. Moreover, F is denoted as the multiple objective function, and Ω is the set that determines the feasible region. Finally, the problem of determining the global minimum solution set is called the *MOP global optimisation problem*.

1.2 Optimisation Schemes

A wide range of optimisation schemes has been designed for dealing with optimisation problems. These optimisation schemes can be classified using different taxonomies. The classification proposed in [326] is considered and exposed herein. It differentiates between *exact algorithms* and *approximate algorithms*. In order to use this classification, some notions regarding complexity theory are first given in this section. Among the approximate algorithms, *meta-heuristics* emerge as general optimisation techniques and yield solutions of acceptable quality in a reasonable execution time frame when solving a wide range of complex optimisation problems. Meta-heuristics, in turn, can be divided into *trajectory-based methods* and *population-based schemes*. Trajectory-based methods only have to consider a unique solution, whereas population-based approaches have to maintain a set of candidate solutions during the entire optimisation procedure. Since this dissertation focuses on population-based meta-heuristics, two of the main issues that arise during their

use—*premature convergence* and *parameter setting*—are described. Finally, a few notions concerning measuring the performance of optimisation schemes are presented at the end of this section.

1.2.1 Complexity Theory

Optimisation problems can be *decidable* or *undecidable*. This latter category—also known as uncomputable problems—includes problems for which an algorithm that solves them will never exist, even having an unlimited amount of computational resources and time [322]. An example of an undecidable problem is the *halting problem* [332]. In this dissertation only decidable problems are considered.

Algorithm Complexity

Algorithms need time and space—memory—as resources for solving an optimisation problem. An algorithm’s complexity can prove useful when analysing its limitations and predicting its resource requirements with a view to terminating its execution. If the time frame is considered, the complexity of an algorithm is the number of steps required to solve a problem whose size is equal to n , and is usually defined considering the worst-case analysis. Knowing the exact number of steps is not necessary, but an asymptotic bound of this number is required. The O -notation [39] (Definition 10) is one of the most frequently used when analysing algorithms. It is based on the asymptotic analysis and it can be used to compute the time and/or space complexity of an algorithm. The asymptotic analysis of algorithms allows characterising the growth rate of their complexity as a function of the problem size. It is important to note that there exist two other well-known notations for analysing algorithms. They are the Ω -notation and the Θ -notation [39].

Definition 10 *If we consider the O -notation, an algorithm has a complexity $f(n) = O(g(n))$ if there exist positive constants n_0 and c such that $\forall n > n_0, f(n) \leq c \cdot g(n)$*

In other words, the function $g(n)$ is an upper bound for the function $f(n)$, i.e. $f(n)$ grows asymptotically no higher than $g(n)$.

Definition 11 *An algorithm is a **polynomial-time algorithm** if its time complexity is $O(p(n))$, where $p(n)$ is a polynomial function of n .*

Hence, a polynomial-time algorithm has a polynomial time complexity of $O(n^k)$, where k is the degree of the upper bound polynomial function.

Definition 12 *An algorithm is an **exponential-time algorithm** if its time complexity is $O(c^n)$, where $c > 1$ is a real constant.*

Problem Complexity

A problem is *tractable* if there exists a polynomial-time algorithm that solves it. In contrast, a problem is *intractable* if no polynomial-time algorithm exists that solves it. Complexity theory deals with decision problems—whose solutions are “yes” or “no”—and not with optimisation problems. However, any optimisation problem can always be reduced to a decision problem. One of the main objectives of computational theory is to categorise problems into complexity classes. A complexity class includes every problem that can be solved using a given amount of computational resources. Thus, problems can be classified into two main complexity classes: P and NP .

Definition 13 *The **complexity class P** contains all decision problems which can be solved by a deterministic algorithm in polynomial time.*

Deterministic algorithms solve the problem without using any random component to carry out their decisions, i.e. given the same input, the same output is always obtained. A deterministic algorithm is polynomial for a decision problem A if its complexity, assuming the worst-case, is bounded by a polynomial function $p(n)$ where n is the input size of certain instance α . Examples of problems belonging to the complexity class P are minimum spanning tree, shortest path problems, or maximum flow networks, among others [74].

Definition 14 *The **complexity class NP** contains all decision problems which can be solved by a non-deterministic algorithm in polynomial time.*

Non-deterministic algorithms—also called *stochastic algorithms*—have at least one fork in which the decision is usually made considering randomness and probabilities. Hence, given the same input, the same output is not always obtained.

We should note that $P \subseteq NP$. However, whether or not $P \subset NP$ is an open research question [72]. In other words, it has been shown that for every problem belonging to the class P there exists a non-deterministic algorithm that can solve it

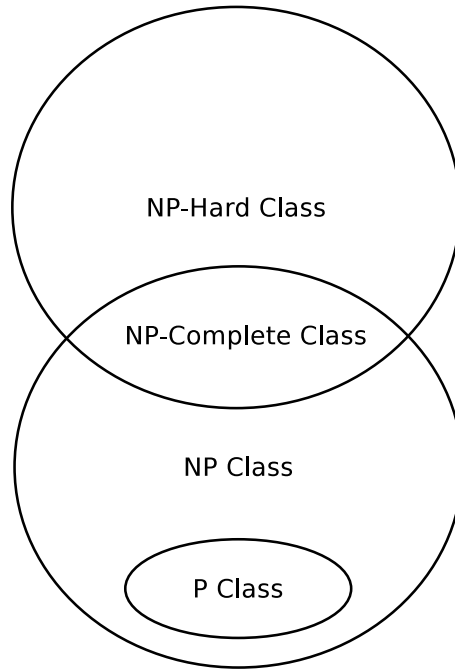


Figure 1.2: Relationship between different complexity classes ($P \neq NP$)

in polynomial time. However, it is not known whether for every problem belonging to the class NP there exists a deterministic algorithm that solves it in polynomial time.

Definition 15 *A decision problem A is **reduced polynomially** to a decision problem B if, for all input instances α for A , an input instance β for B can be built in polynomial time, such that the answer to the instance α is “yes” if and only if the answer to the instance β is “yes”.*

Definition 16 *A decision problem $A \in NP$ is **NP-complete** if all other problems that belong to the class NP can be reduced polynomially to A .*

A direct claim of this definition is that if a deterministic polynomial-time algorithm exists to solve an NP -complete problem, then all problems of class NP might be solved in polynomial time. NP -complete problems are the hardest NP problems to solve.

Definition 17 *A problem A is **NP-hard** if and only if there exists an NP -complete problem B that can be reduced polynomially to A .*

Informally, NP -hard problems are at least as hard as NP -complete problems. However, it is important to note that NP -hard problems do not have to be included in the class NP , since not all of them are decision problems—they can be optimisation problems. Moreover, some NP -hard problems are not included in the NP class, despite being decision problems. For instance, the halting problem is an NP -hard problem. It is also a decision problem, but since it is undecidable, it is outside the class NP . Finally, if an optimisation problem has an associated NP -complete decision problem, said optimisation problem is NP -hard. Figure 1.2 shows the relationships among the classes P , NP , NP -complete and NP -hard, considering $P \neq NP$.

NP -hard problems usually require exponential-time algorithms—unless $P = NP$ —in order to obtain the optimal solutions. Most real complex applications are categorised as NP -hard optimisation problems. Examples of NP -hard problems are scheduling problems, routing and covering problems, or knapsack and cutting problems, among others [74]. This dissertation considers NP -hard optimisation problems.

Depending on a problem’s complexity, it can be solved by *exact algorithms* or by *approximate algorithms*. Exact approaches are able to obtain solutions whose optimality is ensured. However, for NP -complete problems exact algorithms cannot be applied—unless $P = NP$ —since they become non-polynomial-time algorithms, and therefore they are not able to provide optimal solutions in a reasonable time. In contrast, approximate or *heuristic* schemes are able to generate high-quality solutions, even optimal ones, in a reasonable time. However, using approximate algorithms does not guarantee that the optimal solution will be found. Figure 1.3 shows a possible taxonomy [326] which classifies optimisation methods into exact and approximate schemes.

1.2.2 Exact Algorithms

Exact methods can be classified into the following groups: *dynamic programming*, *branch and bound methods*, *constraint programming*, and *A^* search algorithms*. All these approaches can be categorised as *enumerative methods* [270], where the search for solutions involves exploring the whole feasible region and dividing the optimisation problem into smaller sub-problems by using the *divide and conquer strategy* [74]. When these sub-problems are solved, their solutions are combined in order to obtain the complete solution of the initial problem.

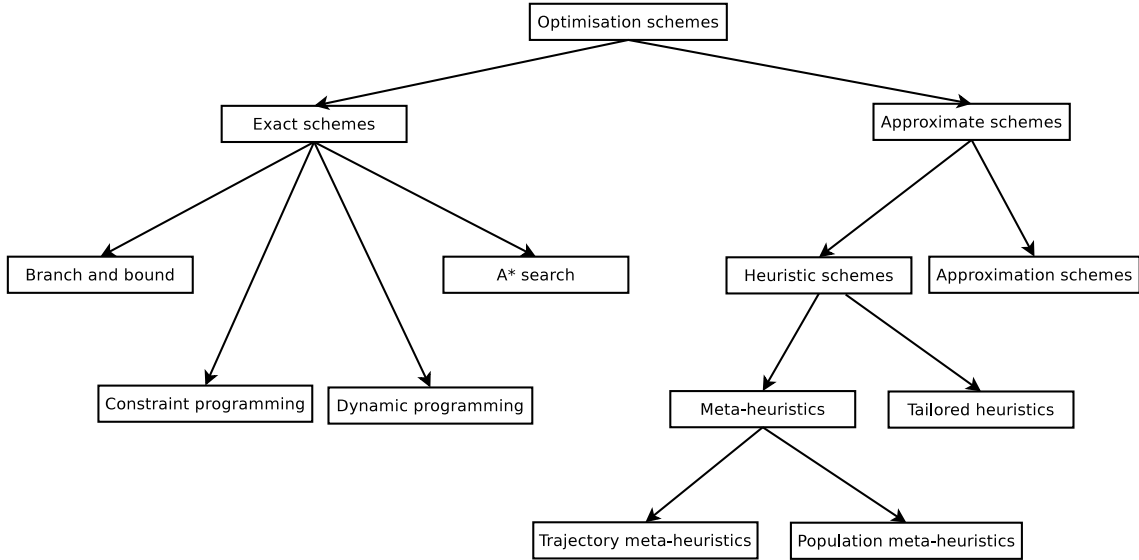


Figure 1.3: Classification of optimisation methods

Dynamic Programming

Dynamic programming [30] is based on dividing a problem into smaller sub-problems that are then easier to solve. The optimal solution to the initial problem is thus provided by obtaining the optimal solutions for each of the sub-problems, which result from a sequence of partial decisions. In dynamic programming, the solution to every solved sub-problem is stored in memory. In this way, solutions that have been previously computed do not have to be recalculated. In addition, the procedure does not enumerate the whole search space, since the partial decision sequences that do not lead to the optimal solution are pruned.

Branch and Bound and A* Search

Branch and bound [196] and A* search [153] algorithms are based on an implicit enumeration of all feasible solutions for a certain optimisation problem. In order to explore the feasible region, a search tree with the following characteristics is dynamically constructed:

- The root represents both the optimisation problem to be solved and, the whole search space.
- The leaves represent the problem's candidate solutions.

- The remaining nodes represent sub-problems.

In branch and bound algorithms, a bounding function is used to prune some of the sub-trees. This function ensures that the search areas represented by these sub-trees do not have to be searched in order to obtain the optimal solution to the problem.

Constraint Programming

Constraint programming [289] models optimisation problems as a set of variables linked by a set of constraints. The variables take their values from a finite domain of integers, with the constraints possibly having mathematical or symbolic forms. Constraint programming is based on alternating *propagation* with *search* methods to find a feasible solution. A propagation algorithm reduces—from variable domains—the values that do not involve a feasible solution. Afterwards, a search algorithm based on a search tree is executed in order to remove possible inconsistent values in the variable domains. This search algorithm performs a branching step to divide the current problem into sub-problems. Branching can instantiate a variable to a feasible value or can add a new constraint. Thus, the problem of optimising an objective function can be reduced to one of solving certain types of feasibility problems.

1.2.3 Approximate Algorithms

Approximate methods can be divided into two sub-classes (Figure 1.3): *approximation algorithms* and *heuristic algorithms*. Approximation algorithms provide demonstrable solution quality and provable run-time bounds. In contrast, heuristics usually find reasonably good solutions in a reasonable time. Heuristics can therefore be applied to large and difficult instances of a wide range of complex problems. Heuristics can in turn be categorised into *tailored heuristics* and *meta-heuristics*. Tailored heuristics are algorithms specifically designed to solve a particular problem and/or instance, whereas meta-heuristics are general methods applicable for solving a wide variety of optimisation problems. Meta-heuristics can be used as approaches that guide the design of underlying heuristics in order to solve particular optimisation problems.

Approximation Algorithms

Approximation algorithms [158] guarantee that the solutions obtained are enough close to the global optimum. An ϵ -*approximation algorithm* [335] is able to generate an approximate solution s that is not less than ϵ times the optimum solution g . Generally, ϵ is called the *approximation factor* and it is used to establish the relative performance of the algorithm. This approximation factor can be a constant or a function which depends on the size of the instance.

Approximation algorithms can be analysed to gain more knowledge regarding the difficulty of problems, as a consequence of which more efficient heuristics can be designed. Nevertheless, approximation algorithms are tailored approaches which depend on the problem at hand. In addition, these algorithms are not very useful for complex applications because the approximated solutions are usually far from the global optimum.

Heuristic Algorithms

Definition 18 “A **heuristic** is a criterion, method, or principle for deciding which of several alternative courses of action promises to be the most effective in order to achieve some goal” [278].

In order to design a heuristic, two main requirements have to be considered. Firstly, a heuristic has to be simple and require a low consumption of computational resources and time. Secondly, it has to make correct decisions instead of selecting bad choices. Heuristics are usually tailor-made approaches that rely on information on the problem or instance being solved to carry out their decisions. Consequently, their design and implementation might become a difficult task. Moreover, once a tailored heuristic is designed, it usually cannot be applied to other optimisation problems and/or instances of the same problem.

In general, there are two main ways to apply heuristics: they can be used as stand-alone optimisation techniques; or, they can be integrated together with other approaches. In the second case, heuristics can help to improve the behaviour of other methods, such as exact algorithms, by recommending the next set of candidate solutions to be explored by the exact algorithm, for instance. In contrast, other approaches can be used to improve the behaviour of heuristics, like restarting, which as its name implies, restarts the heuristic when a certain situation is detected—stagnation over a certain number of iterations, for example. Finally, it is important

to note that the combination of different heuristics might improve the solutions obtained. However, this task is not easy and it might yield improper results [139].

Definition 19 “**Meta-heuristics** are top-level general strategies which guide other low-level heuristics to search for feasible solutions in difficult domains” [70].

In recent decades, meta-heuristics have been applied to a wide range of fields, such as engineering design, aerodynamics, telecommunications, machine learning, data mining, system modelling, signal processing, and planning and scheduling problems, among others [326]. Meta-heuristics are general purpose strategies that, in general, do not make use of problem-dependent information. However, they can consider problem specific knowledge in the form of low-level heuristics controlled by the top-level strategy. The success of meta-heuristics stems from their ability to deal with large instances of complex problems for which there are no applicable exact approaches. Hence, the main aim of a meta-heuristic is to obtain solutions with “acceptable” quality—close to the global optimum—in a reasonable time. Meta-heuristics must therefore be designed with the aim of exploring the solution space in a very efficient manner. However, unlike exact or approximation algorithms, meta-heuristics are not able to guarantee that global optima or even bounded solutions will be obtained. Another drawback, which has become a significant research topic [287, 360], is to design effective stopping criteria for meta-heuristics. Ideally, the execution would finish when convergence is detected, though actually detecting this situation is an arduous task. The most common stopping criterion is based on using an amount of resources that is fixed before the execution starts. In this dissertation, two different stopping criteria are considered: execution time and number of evaluations carried out on the objective functions of the problem being solved.

One of the main issues that arise when designing meta-heuristics is striking the right balance between the exploration of the whole search space—or *diversification*—and the exploitation of its most promising regions—or *intensification* [140]. Promising regions are determined by the best solutions found at any given moment of the search procedure. Diversification tries to ensure that unexplored regions of the search space are visited so that the search process is not bound to a reduced number of regions. Hence, meta-heuristics try to generate solutions that differ from those previously obtained. In contrast, intensification allows for a more exhaustive exploration of the search space and thus aims to improve the quality of the best solutions found by examining their neighbours.

A wide variety of meta-heuristics has been proposed [37], with the following being among the most important: Ant Colony Optimisation (ACO) [97], Artificial

Bee Colony (ABC) [181], Artificial Immune System (AIS) [31], Cultural Algorithms (CAs) [286], Coevolutionary Algorithms (CEAs) [157], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [152], Differential Evolution (DE) [319], Estimation of Distribution Algorithms (EDAs) [24], Evolutionary Programming (EP) [121], Evolution Strategies (ES) [295], Genetic Algorithms (GAs) [160], Guided Local Search (GLS) [341], Genetic Programming (GP) [190], Greedy Randomised Adaptive Search Procedure (GRASP) [115], Iterated Local Search (ILS) [235], Particle Swarm Optimisation (PSO) [183], Simulated Annealing (SA) [186], Scatter Search (SS) [137], Tabu Search (TS) [138], and Variable Neighbourhood Search (VNS) [250].

It is important to note that this dissertation focuses on the study of a family of meta-heuristics that includes some of the aforementioned approaches. This family is called Evolutionary Algorithms (EAs) and it comprises a set of population-based meta-heuristics inspired on biological evolution. The family of EAs mainly includes GAs, ES, EP, and GP, although EDAs, DE, CEAs, and CAs are also categorised as EAs. Finally, Memetic Algorithms (MAs) are hybrid meta-heuristics that combine a population-based approach with a Local Search (LS) procedure.

A large number of taxonomies for classifying meta-heuristics have been devised [61]. In keeping with the taxonomy described in [326], five different criteria can be used to classify meta-heuristics. The first one classifies them using *nature-inspired* methods and *non-nature-inspired* approaches. Most meta-heuristics are inspired by natural processes, such as EAs, which are inspired by biological evolution, or ACO, ABC and PSO, which are inspired by swarm intelligence.

Meta-heuristics can also be grouped into *deterministic* and *stochastic* methods. Deterministic meta-heuristics, like LS or TS ¹, make their decisions considering deterministic rules, i.e. given the same input instance for an optimisation problem, the same output is achieved. In contrast, stochastic meta-heuristics, like SA or EAs, use randomness when making some decisions, meaning that if the same input instance of an optimisation problem, is given, the same output is not always obtained. It is important to note that this feature significantly determines the way in which the performance of meta-heuristics is measured.

Another criterion classifies meta-heuristics depending on whether they do not use information dynamically extracted from the search process, such as LS or GRASP, or they do, like TS for example, which is based on using short-term and long-term memories in order to avoid visiting already explored areas of the search space.

¹In essence, these meta-heuristics do not rely on chance [99], though every meta-heuristic might include some kind of stochastic procedure to carry out its decisions.

The most frequently used classification [37] groups meta-heuristics into *trajectory-based* approaches and *population-based* methods. Trajectory-based schemes, like SA, have to deal with a unique solution during the whole optimisation process, while population-based meta-heuristics, such as PSO or EAs, have to maintain a set of solutions. In trajectory-based algorithms the term trajectory refers to the “path” established by the single solution in the search space during the optimisation procedure. Consequently, trajectory-based meta-heuristics are also called *single-solution* meta-heuristics. We should mention that trajectory-based meta-heuristics are more oriented toward intensification, while population-based meta-heuristics are more explorative methods.

Finally, meta-heuristics can be grouped depending on whether they are *iterative* or *greedy* techniques. Iterative meta-heuristics—EAs and ILS, for example—start with a unique solution or a set of solutions and iteratively modify them by using certain operators until the optimisation procedure ends. Greedy meta-heuristics such as GRASP, on the other hand, build solutions from scratch, assigning a value to each decision variable at each step of the optimisation procedure until complete solutions are obtained.

1.2.4 Premature Convergence

Meta-heuristics have shown great promise to obtain solutions to difficult and complex applications in a wide range of fields. However, they exhibit a tendency to converge towards local optima for some problems, with the likelihood of this occurrence depending on the shape of the fitness landscape [56]. Several methods have been designed with the aim of dealing with local optima stagnation [139]. Some of the simplest techniques rely on restarting the approach when stagnation is detected [230]. In other cases, a component that inserts randomness or noise into the search is used [66]. Maintaining some memory, as TS does, in order to avoid exploring the same areas several times is also a typical approach [140]. Finally, population-based strategies intrinsically try to maintain the diversity of a solution set. By recombining these solutions, a wider area of the decision space might be explored.

In the particular case of population-based meta-heuristics, like EAs, *premature convergence* is one of the most frequent drawbacks to be addressed. It appears when every member of the population is in a sub-optimal region of the search space, and therefore the optimisation scheme is not able to generate new individuals that outperform their corresponding ancestors, i.e. there exists a loss of diversity caused

by the use of a finite population size. This phenomenon is also known as *genetic drift* [108], and it is the main reason for the appearance of premature convergence. Several methods have been devised for dealing with premature convergence. Most of them preserve the diversity in a set of solutions [337]. Some of the most frequently used are the following:

- Increase the population size to avoid *genetic drift* [108].
- Apply mating restrictions such as *incest prevention* [306], i.e., keep very similar individuals from mating. This is also known as *speciation*.
- Perform *cataclysmic mutation* [111]—highly disruptive mutations—when diversity has been lost.
- Perform selection applying *fitness sharing* [267]. In this case, highly similar individuals are clustered and penalised by sharing the obtained fitness values among the members of the group that lie in the same niche (i.e., those that are very close to each other either in the decision space or the objective space).
- Apply *crowding-based selection* where each offspring replaces similar individuals in the parent population [227].
- Use complex population structures, such as the *island-based model* [19] or the *cellular approaches* [263].

Diversity can help the optimisation procedure in two ways. Firstly, there exists a relationship between diversity and the diversification and intensification capabilities of EAs [337]. Among other advantages, as previously stated, a proper balance between diversification and intensification might allow the search space to be explored more efficiently. Secondly, maintaining proper diversity might allow combining different building blocks in crossover operations [175].

When dealing with single-objective optimisation problems, one of the diversity preservation strategies that has gained some popularity in recent years lies in using multi-objective methods to solve them [297]. *Multi-objectivisation* [188] refers to the reformulation of originally single-objective problems into multi-objective ones, the goal being to open up monotonically increasing paths to the global optimum that are not available under the single-objective formulation. Note that, when using multi-objectivisation, a multi-objective optimisation scheme has to be applied since a MOP has to be solved. Because multi-objectivisation changes the fitness landscape, it can often make it easier to find optimal solutions by avoiding local optima [151], although it has been noted that in some instances it can also produce a harder problem [43]. An alternative to this scheme—which has also been

referred to as multi-objectivisation in other works [255]—is based on considering diversity as an auxiliary objective function [2, 45, 330]. However, there is an important difference between these new kinds of approaches and the original definition of multi-objectivisation, since in the new proposal the calculation of the auxiliary objectives depends on the other individuals in the population. That is why these new approaches are called *diversity-based multi-objective schemes* so as to differentiate them from multi-objectivisation methods.

The application of a diversity-based multi-objective scheme to a single-objective optimisation problem requires defining a set of at least two objective functions. The first one is the original objective associated with the single-objective problem being solved. The remaining objectives—most of the proposals consider only one additional objective—are measures of the diversity introduced by an individual itself with respect to the population. Since the use of diversity-based auxiliary objectives has yielded high-quality results in both benchmark problems [330] and real-world applications [130], diversity-based multi-objective schemes offer a promising approach for avoiding premature convergence. Finally, let us note that this dissertation will focus on the application of EAs. Multi-Objective Evolutionary Algorithms (MOEAs) are the adaptation of EAs for dealing with MOPs, and are one of the most commonly applied multi-objective meta-heuristics. Thus, this dissertation consider the application of *diversity-based MOEAs* to single-objective optimisation problems.

1.2.5 Parameter Setting

In addition to avoiding the problem of premature convergence, another arduous task involves setting the parameters of a meta-heuristic. The problems of parameter setting and premature convergence are closely related. If the parameter values of a population-based meta-heuristic are chosen poorly, the balance between diversification and intensification might become disproportionate, producing a loss of diversity and, as a result, premature convergence. To configure a meta-heuristic, several components and/or parameters must be specified. For instance, EAs have different components [108], such as the mutation and crossover operators, which must be defined. In addition, some parameters like the mutation and crossover rates—the probabilities used to apply the aforementioned operators—must be also fixed. In general, the performance of any meta-heuristic, and consequently the quality of the solutions obtained, are highly dependent on these components and parameters. As a result, it is vital that the parameters be properly set. In order to completely configure a meta-heuristic, two types of information are required [26, 239]:

- *Symbolic*—also referred to as *qualitative*, *categoric* or *structure* parameters—like the crossover and mutation operators of an EA.
- *Numeric*—also referred to as *quantitative* or *behavioural* parameters—such as the crossover and mutation rates of an EA.

For both kinds of parameters, the different elements of the domain are known as parameter values, and a parameter is instantiated by assigning it a value. The main difference between the two types of parameters lies in the size and structure of their respective domains. Symbolic parameters, like the crossover operator of an EA, have a finite domain in which order is not established and a distance metric is not defined. In contrast, numeric parameters, such as the mutation rate of an EA, have an infinite domain in which a distance metric and an order can be defined for the values. Thus, optimisation and search methods can readily be used to look for the appropriate values of numeric parameters. However, in the case of symbolic parameters, as noted above, distance metrics cannot be applied between two values, and therefore optimisation schemes are not able to profit from the definition of these types of metrics for setting said parameters.

Parameter setting strategies are commonly divided into two categories: *parameter tuning* and *parameter control*. In parameter tuning [107]—also called *offline* or *endogenous* setting—the objective is to identify the best set of parameters for a given meta-heuristic. Once a suitable parameter set is identified, an algorithm is executed using the selected parameter values, which remain fixed for the full run. Traditionally, parameter tuning has involved performing different executions of the same meta-heuristic using different parameterisations. Afterwards, the parameterisation that provides the best performance, i.e. the best solutions for the optimisation problem in question, is selected. The main disadvantages of parameter tuning based on this systematic approach are the following [106]:

- Since the parameters interact, systematically testing all of their possible combinations is practically impossible.
- The amount of computational resources and time invested in the process of parameter tuning is huge, even if the parameters are independently optimised regardless of their interactions.
- For a given problem, the selected parameter values are not necessarily optimal for every stage of the optimisation procedure, even if the previous effort made to tune them was significant.

In past decades, research into parameter tuning mainly focused on looking for a general set of optimal values for the parameters of a specific algorithm, which allowed promising results to be obtained across several optimisation problems [88, 143]. However, it is now generally accepted by the research community that a single set of parameters is unlikely to be optimal across a range of problems and that an algorithm has to be specifically configured in order to successfully solve a given optimisation problem [19]. In fact, the *No Free Lunch* theorem [351] provided evidence from a theoretical point of view that a given optimisation method is not appropriate for all optimisation problems. Hence, this has given rise to a significant field of research in automated parameter tuning to enable an algorithm to be suitably tuned for the optimisation problem at hand. Tuning methods have been applied to a large number of meta-heuristics, including AISs [3], GAs [251] GLS [6], MAs [353] or SA [3], for instance. Moreover, a wide variety of parameter tuning approaches used in different fields have been proposed [107], including the *sampling* [260] and *racing* [231] methods. Regardless of the tuning method chosen, it is crucial to recognise that parameters usually interact in highly non-linear ways, meaning they should ideally be tuned simultaneously rather than independently [106, 105]. However, simultaneously tuning several parameters requires a huge amount of experimentation and is likely to be computationally infeasible. As a result, parameter tuning is often performed considering the parameters independently, though this is likely to lead to a sub-optimal set of parameters for the reasons given above. Moreover, several studies have concluded that the use of a static set of parameters during a complete run seems to be inappropriate [13, 14, 15, 84, 106, 105, 156]. In fact, it has been empirically and theoretically demonstrated that different values of parameters might be optimal at different stages of the optimisation process [13, 316]. As a result, it seems more appropriate to apply strategies that allow the parameter values to adapt or change during the course of a run.

This is precisely the goal of parameter control [106]—also known as *online* or *exogenous* setting—which is based on designing a control strategy that dynamically selects the most suitable parameter values to use at every stage of the search process. A wide range of parameter control methods has been proposed in the literature [106]. They have been successfully applied to different meta-heuristics, such as ES [191], DE [280], GAs [117], GRASP [221] or PSOs [355, 359]. Most of them are tailor-made methods, however, and depend on the specific meta-heuristic and its parameters. Thus, it would be desirable to design generic parameter control schemes that can be directly applied to different meta-heuristics and/or parameters. *Fuzzy Logic Controllers (FLCs)* and *Hyper-heuristics* can be used as generic parameter control approaches. In fact, novel proposals for both approaches are analysed in

this dissertation.

Fuzzy Logic Controllers

In recent years, our knowledge regarding the performance of meta-heuristics has significantly increased due to the large number of empirical analyses conducted on a wide range of applications in different areas. It would be desirable to profit from this human knowledge by encapsulating it within an algorithm to automate the task of improving the behaviour and performance of meta-heuristics. However, this sort of knowledge is usually incomplete, imprecise, and/or not well organised. Consequently, the application of *fuzzy logic-based methods* would appear to offer a promising approach for handling this kind of knowledge. One application of fuzzy logic is the design of FLCs.

FLCs let us define control approaches to which human knowledge can be incorporated intuitively. By using an FLC to control the parameters of a meta-heuristic, we can use any combination of performance measures and current parameter values as the input to the controller to compute new parameter values, which can be absolute or relative values with respect to the current values. If every component of an FLC is designed bearing in mind its general application—for example, selecting the proper performance measures—then an FLC can be used as a way to adapt different parameters belonging to different meta-heuristics. It is worth noting that FLCs have been successfully applied to adaptively adjust the parameters of different meta-heuristics, including ACO [63], GAs [210], or PSO [241], among others.

Hyper-heuristics

A hyper-heuristic can be viewed as a method that iteratively chooses between a set of candidate low-level heuristics or meta-heuristics in order to solve an optimisation problem [50]. The motivation behind using a hyper-heuristic is two-fold. Firstly, the hyper-heuristics has no knowledge of the problem domain being solved—the same hyper-heuristic can be used in a new domain by swapping in a new set of low-level heuristics or meta-heuristics. Secondly, it ought to be possible to improve the performance of any single heuristic or meta-heuristic in the low-level candidate set by properly combining them during the course of the search [46].

Hyper-heuristics are closely related to parameter control schemes [309]. If the low-level approaches represent different configurations of the same heuristic or meta-heuristic, then the hyper-heuristic might select the most suitable configuration

continuously during the search procedure. It is important to remark that hyper-heuristics have been successfully applied as parameter control techniques in different meta-heuristics for several problem domains [193, 285].

1.2.6 Performance Measurement

After applying any optimisation scheme to a particular optimisation problem, its performance must be measured in order to check whether the optimisation method is suitable for the optimisation problem at hand. In the case of exact algorithms, since they guarantee that the optimal solutions will be obtained, the most frequently used metric to measure their performance is the time invested in obtaining said optimal solutions. However, in the case of approximate algorithms, and particularly when meta-heuristics are applied, other indicators have to be considered to measure their performance. This is because meta-heuristics do not ensure that the optimal solutions will be obtained. Moreover, a large number of meta-heuristics are stochastic approaches. Considering the classification carried out in [25], performance metrics can be classified based on the following criteria: *quality of the solutions*, *computational effort*, and *robustness*.

As concerns the *quality of the solutions*, performance metrics are generally based on calculating the distance or the error between the solutions obtained and the optimal ones. However, for a large number of optimisation problems the optimal solutions are not known. For these cases, other solutions can be used as the reference set. The most frequently used are lower/upper bound solutions, the best-known solutions or solutions defined by the human decision maker considering any kind of requirement for the problem at hand. This dissertation deals with problems for which the optimal solutions are known, while it also considers other problems for which the best-known solutions are used as the reference set so as to make comparisons based on the quality of the solutions.

From the point of view of the *computational effort*, the metrics are closely related to the stopping criterion selected for the meta-heuristic at hand. Among the most widely applied metrics, it is important to mention the time invested in achieving a certain stopping criterion. However, this metric depends on the hardware and software present in the machine where the optimisation scheme is executed. As a result, another frequently used metric measures the number of function evaluations needed to reach a certain stopping criterion. The main drawback of this indicator is that it is not suitable for optimisation problems whose objective functions are not computationally expensive. Since stochastic meta-heuristics are considered in

this dissertation, *Run-Length Distributions (RLDs)* [116] are used as the metric for comparisons based on the computational effort. RLDs—also called *Run-Time Distributions* or *Time-to-Target Plots*—show the relationship between the success rate and the run time or the number of function evaluations. The success rate is defined as the probability of achieving a certain preset quality level within the given run time or number of function evaluations. Hence, different optimisation schemes are compared based on the run time or the number of function evaluations invested by each approach to achieve some quality level and some success rate, both of which are predefined. Finally, it is important to note that the use of RLDs has been suggested as a suitable performance metric for making comparisons based on the computational effort [162].

The *robustness* of a meta-heuristic is another possible criterion for grouping performance metrics. The term robustness can be defined in several different ways. Firstly, a meta-heuristic is robust if the variability of the results does not significantly increase even if its parameters are modified. Another criterion defines a meta-heuristic as robust if it performs reasonably well among a large set of problems and/or instances with the same parameterisation. Lastly, if a stochastic meta-heuristic is applied, a low deviation from the average in the results obtained by the stochastic approach over several runs indicates that it is robust. In this dissertation, the three above definitions of robustness are used to a greater or lesser extent. Since most of the meta-heuristics applied herein are stochastic methods, well-known statistical indicators, such as the Standard Deviation (SD) or the Coefficient of Variation (CV) are used to measure the robustness of the proposals.

Finally, in order to determine whether the results obtained by different stochastic methods present statistically significant differences, an *statistical analysis* is mandatory. For this reason, in this dissertation the following statistical procedure [92, 302] is applied in an effort to assign statistical confidence to the results. First, a *Shapiro-Wilk test* is performed to check whether the values of the results follow a normal (Gaussian) distribution or not. If so, the *Levene test* checks for the homogeneity of the variances. If samples have equal variances, an *Analysis of Variance (ANOVA)* is done; otherwise, a *Welch test* is performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis test* is used. In every case, a significance level of 5% is considered.

1.3 Parallel and Distributed Computing

The increase in computational power has contributed to some of the greatest advances in different areas of science and technology. This increase in computational power, however, also means that the number of problems that can be solved has also grown, which in turn requires much more computational power. Examples of this kind of challenging problem can be found in the fields of climate modelling, energy research, medicine, data analysis, and many more. These and other future problems will be solved only if the performance of processors continues to improve.

In recent decades the performance of processors has increased exponentially. As expressed by *Moore's Law* [252], the number of transistors that could be integrated onto a circuit doubled every two years, i.e. a growth rate equal to 50% per year. This fact meant that users and programmers only had to wait until the next generation of processors to obtain better performance from their applications. However, things have changed, and nowadays the growth rate has decreased to 20% per year, approximately. Transistor speed, and consequently processing speed as a whole, grows as a consequence of a reduction in transistor size. This increase in speed requires a higher consumption of power by the processor, the majority of which is dissipated in the form of heat. The main drawback is that the amount of heat which must be dissipated is greater than the ability of the existing devices responsible for carrying out this task [277]. Moreover, the process of transistor miniaturisation is also reaching its limits. These are the two main reasons why the speed of processors cannot be increased any further. As a result, processor manufacturers have started to use parallelism as a way to improve the performance of processors. Instead of integrating a larger number of circuits in the next generation of processors and thereby increase their speed, the tendency now is to include several complete processors or *cores* on a single chip, thus providing *multi-core architectures*. From now on, and for the sake of simplicity, the traditional monolithic processor will be referred to as a "*single-core processor*".

Hence, a challenging task for programmers has arisen, since the sequential applications that have traditionally been designed for single-core processors have to be adapted to deal with multi-core architectures if their performance is to be improved. Ideally, a set of tools would be available for converting sequential programs into parallel programs. Although some research has been done in this area [94], the process of transforming common sequential programming constructs into efficient parallel programming constructs is quite difficult [275], and even more so if complex sequential programs are involved. Consequently, programmers must be able to write parallel programs directly instead of writing sequential programs and parallelising

them by using some kind of automatic tool.

In order to write parallel programs the processing work must be divided among the available cores. The work can be distributed in the form of either tasks or data. In *task parallelism* the problem being solved is divided into tasks, and every task is assigned to a core. In contrast, in *data parallelism* the data of the problem being solved is distributed among the various cores in blocks, and every core executes the same program on its corresponding block of data. The most common way to write parallel programs is to use explicit parallel constructs, available as extensions to languages like Fortran or C++. Another option might be to use a higher-level language which, while facilitating the development task, will also reduce the application's performance. This section aims to provide some basic information on parallel and distributed computing. Firstly, the most widely used parallel architectures are presented using a taxonomy that adheres to different criteria. Afterwards, the most important parallel programming models are described. Finally, some common metrics used to evaluate the performance of parallel approaches are defined.

1.3.1 Parallel Architectures

In parallel computing one of the most frequently used taxonomies for classifying parallel computer architectures is the one proposed by Flynn [119]. In this taxonomy two different criteria are considered: the number of *instruction streams* and the number of *data streams*. Hence, four possible groups emerge from the combination of these two criteria:

- *Single Instruction Single Data (SISD)*. This group includes systems that execute a single instruction at a time and fetch or store a single data item at a time. Traditionally, this class has represented machines with a single-core processor executing a sequential program. The SISD architecture is disappearing, however, since most processors are currently based on multi-core architectures. For instance, the classical Von Neumann architecture [275] is a SISD system.
- *Single Instruction Multiple Data (SIMD)*. This group contains parallel systems where a single instruction stream is executed on multiple streams of data, i.e. the same instruction is applied to different pieces of data. SIMD architectures are well-suited, for example, for tasks like parallelising iterative programming sentences involving large data sets. Data is divided among the processors and each one executes the same instruction on its corresponding block of data. Thus, SIMD architectures are suited to exploiting parallelism in data. *Vector processors* [109] are an example of SIMD systems. *Graphics Processing Units*

(GPUs) [262] are not pure SIMD systems, even though they make use of some elements of SIMD architectures [185].

- *Multiple Instruction Single Data (MISD)*. Systems categorised in this group are able to execute multiple instruction streams on a single stream of data, although no practical implementation of this kind of architecture exists.
- *Multiple Instruction Multiple Data (MIMD)*. This group, which represents the most general model of a parallel architecture, contains systems that are able to simultaneously execute multiple instruction streams on multiple data streams. Thus, MIMD architectures usually consist of a set of processing units or cores that act independently, i.e. they are asynchronous, unlike SIMD systems, which are synchronous systems. For example, GPUs are regarded as MIMD machines nowadays. Finally, it is worth mentioning that *shared-memory systems* and *distributed-memory systems* are the main types of systems based on the MIMD architecture.

Shared-memory Systems

In shared-memory systems there are several independent processors—single-core or multi-core—connected to the main memory via an interconnection network such as, a bus or a crossbar. Hence, every processor is able to access every position available in the memory. In this kind of parallel system, communications among processors are handled implicitly by accessing shared data structures. Shared-memory systems are easy to program, but they have poor scalability. When the number of processors increases in shared-memory systems, the number of accesses to the communication network also grows, meaning a higher memory bandwidth is required. Other common problems in shared-memory parallel systems are *cache coherence* and *false sharing* [275]. Figure 1.4 shows the general architecture of a shared-memory system.

When a shared-memory system consists of multi-core processors, two possible ways exist to access memory. On the one hand, the interconnection network can connect every multi-core processor directly to the main memory. As a result, every processor is able to access every memory address with the same access time. In this cases, the shared-memory system is called a *Uniform Memory Access (UMA)* system (Figure 1.5). On the other hand, the interconnection network of a shared-memory system can connect every multi-core processor to a certain block of the main memory. Thus, every multi-core processor is able to access another processor's memory block by using the built-in hardware specifically designed for this purpose. In this kind of system the access time varies depending on whether a processor is accessing a

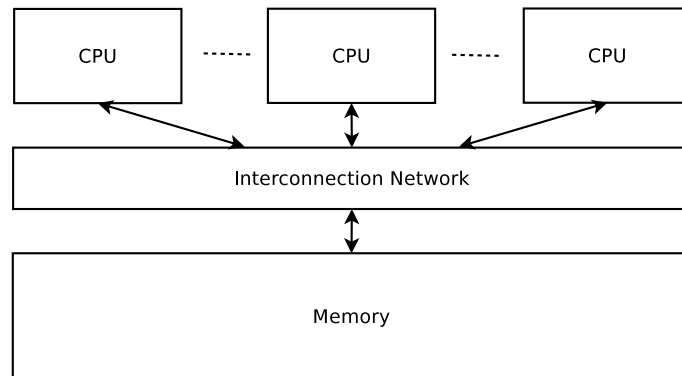
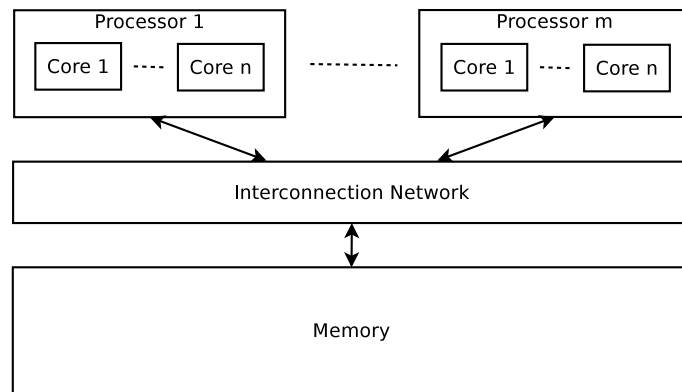


Figure 1.4: General architecture of a shared-memory system

Figure 1.5: Architecture of a uniform memory access system with m processors and n cores each

memory position located in its corresponding memory block or the memory block of another processor. In this cases, the shared-memory system is called a *Non-Uniform Memory Access (NUMA)* system (Figure 1.6). UMA systems are even easier to program than NUMA systems. This is because the programmer does not have to worry about access time concerns. However, since the scalability of NUMA systems is usually better than that of UMA systems, the amount of memory is consequently larger in NUMA systems than in UMA systems.

Distributed-memory Systems

In pure distributed-memory systems, there are several single-core processors and a private memory associated with each. Moreover, different processors are connected

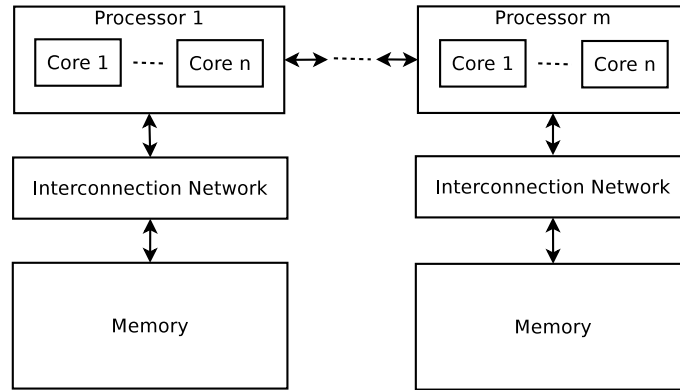


Figure 1.6: Architecture of a non-uniform memory access system with m processors and n cores each

by a communication network with a certain topology, such as hypercubes or toroidal meshes, among others. Communications among processors are carried out by sending explicit messages or by making usage of special functions which allow a given processor to access the private memory of another. Distributed-memory systems are harder to program than shared-memory systems, since the programmer has to explicitly distribute data and work among processors. However, a distributed-memory system is much more scalable in terms of the number of processors than a shared-memory system. Consequently, distributed-memory systems are better suited than shared-memory systems to problems with huge data transfer and/or computational requirements. Figure 1.7 shows the general architecture of a distributed-memory system.

Clusters of Workstations (COW), or simply *clusters*, are one of the most popular distributed-memory systems. A cluster consists of a set of machines or nodes interconnected by a communication network. The main aim of a cluster is to obtain a good ratio between its cost and its performance. The first clusters were based on the architecture known as “*Beowulf*”. It consists of a set of nodes—built from cheap standard components—commonly interconnected by an Ethernet network at 10 Mb/s. Nowadays, the nodes of a cluster are interconnected by faster communication networks like Gigabit Ethernet or Infiniband.

Finally, it is worth mentioning that every node in a distributed-memory system might be a shared-memory machine, with one or several multi-core processors. These systems are called *hybrid distributed-shared memory systems* in order to differentiate them from pure distributed-memory systems. Hybrid machines are the most widely used nowadays due to their flexibility and to their applicability to a wide range of

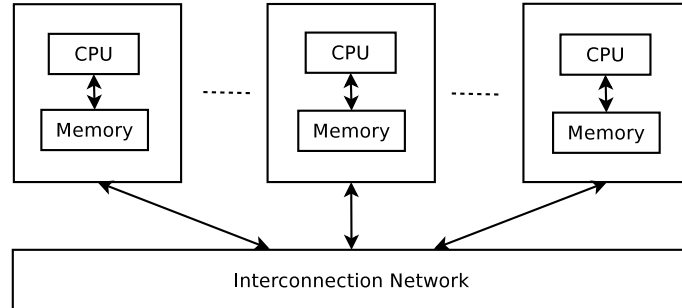


Figure 1.7: General architecture of a distributed-memory system

problems.

Other Criteria for Grouping Parallel Systems

Due to the large number of parallel systems that have emerged in recent decades, each with its own features, the taxonomy proposed by Flynn is not appropriate [334]. This is the reason why other criteria should be used to classify parallel systems [326]:

- *Homogeneous/Heterogeneous parallel architectures.* Parallel systems can be classified based on the homogeneity of their components, i.e. processors, interconnection networks, software, etc. For example, a cluster in which all the nodes use the same architecture is a homogeneous parallel system. In contrast, an example of a heterogeneous parallel system might be a laptop, with its own multi-core processor and a GPU added on. Currently, much more powerful machines and much faster networks have emerged, leading to the appearance of heterogeneous *Networks of Workstations (NOW)*. The main differences between a NOW and a COW lie in the cost of the communications and in the workload of the machines, which are higher in the former.
- *Shared/non-shared parallel architectures.* In a non-shared system the resources are dedicated to executing a single application belonging to a single user. In contrast, in a shared system resources are shared by different users and/or applications. For instance, a NOW is usually a shared parallel system, while a COW is generally a non-shared parallel system.
- *Strongly/weakly coupled parallel architectures.* This criterion classifies parallel architectures depending on the distance between the processing units. In a

strongly-coupled parallel system, the processing units are interconnected by a Local Area Network (LAN), while in a weakly-coupled parallel system, the processing units are usually interconnected by a Wide Area Network (WAN), and consequently the communications cost is higher than in strongly-coupled systems. A COW is, in general, a strongly-coupled machine, whereas a *computational grid* is a weakly-coupled system. A computational grid is a scalable system that consists of a set of dynamic heterogeneous resources distributed in different geographic locations across multiple domains and administrated by different organisations [124].

- *Volatile/non-volatile parallel architectures.* In volatile systems, the availability of the resources is dynamic since they have a high probability of failure. Volatile parallel systems have to deal with issues such as *dynamic resource discovery* and *fault tolerance*. Examples of volatile systems might be a large NOW or some computational grids.

1.3.2 Parallel Programming Models

As was stated in the previous section, ideally there would be tools for automatically converting sequential programs into efficient parallel programs. However, such tools do not exist, and programmers have to write parallel code directly by using parallel programming models. Parallel programming models can be grouped by considering the architecture of the underlying parallel system as a criterion for classifying them. As a result, since the most important taxonomy divides parallel architectures into shared-memory systems and distributed-memory systems, the most frequently used models can be classified as *shared-memory programming models* and *distributed-memory programming models* [275]. Before explaining the classification of these models, it is important to explain some of the terminology used in this section. When a parallel program is executed on a shared-memory system, the program generally starts its execution in the form of a process that forks a certain number of *threads*—light-weight processes—which are run simultaneously. In contrast, when a parallel program is run on a distributed-memory system, the program is usually executed as a set of different processes that are executed simultaneously.

In shared-memory programs all threads can access every memory address asynchronously, and the main problem facing programmers is the appearance of *race conditions*, which can be avoided by the use of *mutual exclusive locks*, *semaphores* or *monitors*, among other mechanisms [275]. The main shared-memory programming models are *multi-threading* and the use of *compiler directives*. In multi-threading,

a process forks into several threads that share the same memory address space. In single-core architectures, only one thread can be executed at a time, while in multi-core architectures every thread can be executed on one core. The most widely used library which allows multi-threading to be implemented is *Portable Operating System Interface (POSIX) Threads* [52]. The use of compiler directives can be viewed as another shared-memory parallel programming model, the most representative example of which is *Open Multi-Processing (OpenMP)* [65, 271, 275]. OpenMP is a specification that consists of a set of routines, environment variables, and compiler directives in the form of *pragmas* that can be included in programs written in Fortran, C, or C++. Pragmas are included in the program source code and indicate the parallel regions that the compiler has to consider. There are several compilers that implement the OpenMP specification, such as *GNU Compiler Collection (GCC)* [328].

In distributed-memory systems, processors can only access their private memory. The most prominent distributed-memory programming model is *message passing*. In this programming model, communications among processes involves the sending and receiving of messages synchronously or asynchronously. The most important specification based on the message passing model is *Message Passing Interface (MPI)* [244, 275], whose most widely spread implementations are *MPICH* [11] and *Open MPI* [329]. Another important library is *Parallel Virtual Machine (PVM)* [134]. Libraries based on the MPI specification are very powerful tools for developing parallel programs. However, since it is not a high-level programming library, the developer has to specify a large number of details. In fact, MPI has been called “*the assembly language of parallel programming*” [275].

Finally, it is important to note that applications for hybrid systems—a cluster of multi-core nodes, for instance—may be programmed by using a shared-memory library for intra-node communication, such as OpenMP, and a distributed-memory library like MPI for internode communication. This is usually done when the approach at hand requires the highest performance. However, managing two different libraries places even more demands on the programmer, meaning that only one library, for example MPI, might be used for both intra-node and internode communications. Similarly, OpenMP could be used in a pure distributed-memory system thanks to the use of libraries that are able to emulate a shared memory.

Algorithmic Skeletons

Another criterion that can be used to classify parallel programming models relies on the *problem decomposition* to yield different *parallel programming paradigms* or

algorithmic skeletons [350]. In this section, the term process refers to a process or a thread indistinctly. In the *task-farming* paradigm two different types of processes are defined: a *master process* and a set of *worker processes*. Firstly, the master is responsible for decomposing the problem into several sub-tasks, which are distributed among workers. Then, every worker executes its corresponding sub-task and when it finalises, sends the results to the master. Finally, the master gathers all the results provided by the workers and combines them to create the final solution to the problem.

The *Single-Program Multiple-Data (SPMD)* paradigm is one of the most widely used. Data is divided into different blocks and every block is assigned to a process. Then, every process executes the same program but on its corresponding block of data. In this paradigm it is usual for data dependencies among different processes to appear, i.e. a process cannot continue with its execution until another process ends and sends its results. This might require the use of communication and synchronisation mechanisms.

The *pipeline* paradigm is based on decomposing the tasks of the parallel program into several sub-tasks that are executed at different stages. The efficiency of this paradigm depends on the ability to properly balance the load across the stages of the pipeline. The communication pattern is usually very simple because data flows only between adjacent stages.

Programs based on the *parallel divide and conquer* paradigm use an extension of the well-known sequential divide and conquer strategy. A problem is recursively reduced to a set of smaller sub-problems. When the sub-problems are small enough, they are solved and the results are subsequently merged until a complete solution is obtained for the initial problem. Since each sub-problem is independent, it might be assigned to different processes. It is important to note that, although this paradigm seems to exploit task parallelism, it in fact exploits data parallelism since the code used to solve every sub-problem is the same but the associated data are not.

In the *speculative parallelism* paradigm some processes execute some pieces of code by speculating on the results that other processes might obtain. In some cases their assumptions are invalid, and the computed data must be discarded. However, when their assumptions turn out to be correct, the completed computation might improve the overall performance. This paradigm is usually applied when other paradigms are not suitable.

Sometimes, parallel programs need to combine various features from different paradigms. In these cases, they are said to follow a *hybrid paradigm*. For instance,

there are cases where data and task parallelism are simultaneously applied. This is closely related to nested algorithmic skeletons. An example could be a pipeline consisting of different algorithmic skeletons, one for each stage.

1.3.3 Performance Measurement in Parallel Applications

The main aim of a parallel program is to improve performance. As a result, indicators that measure this performance are required [275, 194]. Among the most important are the *speedup* and the *efficiency* indicators. *Scalability* is also a desirable feature of parallel programs. This section first presents the definition of these two metrics. Then, an observation made by Amdahl [8] concerning the maximum theoretical speedup that a parallel program can provide is described. Finally, an extension of the metrics exposed herein to evaluate the performance of parallel meta-heuristics is explained.

Speedup, Efficiency, and Scalability

Given a certain sequential program and its equivalent parallel version, the ideal case would be to have $T_{par} = T_{seq}/p$, where T_{par} is the time invested by the parallel program, T_{seq} the time invested by the sequential program, and p the number of cores—one thread/process per core—used to run the parallel version. This would mean that our parallel approach is able to run p times faster than the sequential version. When the ideal case arise, it is said that a *linear speedup* is obtained. Sometimes T_{seq} is referred to as the time needed by the fastest sequential program to complete an execution on the fastest processor available. In a large number of cases, however, the fastest sequential program is not known, and therefore the sequential program is the one on which the parallel program is based. Furthermore, said sequential program is executed on a single processor of the parallel machine.

This is unlikely to yield a linear speedup due to the overhead introduced by the use of multiple processes/threads. For instance, in a shared-memory program there will probably exist race conditions among different threads, meaning that some mutual exclusion mechanism will be needed to resolve them. The use of mutual exclusion mechanisms introduces certain overhead in the run time of the parallel program. Another example might be the intensive usage of the communication network by the different processes in a distributed-memory program. Moreover, the larger the number of processes/threads, the higher the overhead. It is important to note that sequential programs do not have to deal with this overhead. As a result, obtaining

the ideal equality $S = p$ is unusual if the *speedup* of a parallel program is defined as shown in Equation 1.5.

$$S = \frac{T_{seq}}{T_{par}} \quad (1.5)$$

Note that as p grows, S should become a smaller and smaller fraction of the ideal linear speedup. In other words, the ratio S/p will likely become smaller and smaller as p increases. This ratio is commonly called the *efficiency* of a parallel program—Equation 1.6.

$$E = \frac{S}{p} = \frac{\left(\frac{T_{seq}}{T_{par}}\right)}{p} = \frac{T_{seq}}{p \cdot T_{par}} \quad (1.6)$$

It is clear that E , S , and T_{par} depend on the number of processes/threads p . However, it is also necessary to bear in mind that E , S , T_{seq} , and T_{par} depend on the size of the problem being solved. Many parallel programs are built by distributing the work of the sequential program among different processes/threads, and adding the parallel overhead due to synchronisation and/or communication. Hence, if $T_{overhead}$ is the time resulting from the parallel overhead, the execution time of the parallel program can be defined as shown in Equation 1.7.

$$T_{par} = \frac{T_{seq}}{p} + T_{overhead} \quad (1.7)$$

Note that when the problem size increases, $T_{overhead}$ does not grow as fast as T_{seq} . This is because when the problem size grows, every process/thread has to perform a larger amount of work, whereas the relative amount of time invested in coordinating processes/threads should be smaller. Consequently, it is quite common for the speedup and the efficiency of a parallel program to increase when the size of the problem being solved grows.

Ideally, a parallel program is *scalable* if the efficiency remains constant when the number of processes/threads and the problem size increase at the same rate. However, it is also scalable if the efficiency is unchanged as the processes/threads increase without a corresponding increase in the problem size. The first definition is also known as *weak scalability*, whereas the second one is called *strong scalability*. Scalability analyses can be used to check whether a parallel program is scalable or

not. In scalability analyses several executions are carried out by altering the number of processes/threads and/or the problem size. In this dissertation, scalability analyses are performed in order to evaluate the performance of parallel approaches.

Amdahl's Law

Amdahl's law says that a parallel program is able to provide a maximum speedup, which is frequently very limited, regardless of the number of cores available. Thus, let us consider a sequential program where a fraction r cannot be parallelised. This fraction is also called the “*inherently serial*” fraction of a sequential program. Amdahl's law therefore indicates that the parallel version of the program will never provide a speedup better than $1/r$, i.e. $S \leq 1/r$, regardless of the number of cores used. For instance, suppose that a sequential program has a inherently serial fraction $r = 1 \cdot 10^{-2}$. This means that the associated parallel program will not be able to provide a speedup better than 100 even if thousands of cores are available. Amdahl's law, however, does not take into account aspects such as the size of the problem being solved. For a large number of cases, as the problem size increases the inherently serial fraction of the program decreases, and consequently a larger number of cores can be used in order to improve the speedup, as stated by *Gustafson's law* [145].

Performance Evaluation in Parallel Meta-heuristics

The metrics presented in previous sections, as well as Amdahl's law, are suited to exact and deterministic algorithms. However, the use of parallel approximate algorithms, like parallel meta-heuristics, which are usually stochastic approaches, might be able to provide *super-linear speedups*, i.e. $S > p$, for some problems. This is because some parallel meta-heuristics might be able to explore the search space more efficiently than the corresponding sequential meta-heuristics, thus yielding similar or even better solutions than the sequential methods in less time. In contrast, it is possible for parallel meta-heuristics to suffer from stagnation and thus exhibit a worse behaviour than sequential meta-heuristics, which would yield unsatisfactory speedups. This is why measuring the performance of parallel meta-heuristics is an arduous task.

In this dissertation, speedup and efficiency indicators are computed by applying the following procedure. Firstly, RLDs—Section 1.2.6—are calculated for both sequential and parallel approaches by pre-specifying a certain quality level. It is important to recall that RLDs provide an indication of the relationship between success rates

and the run time or the number of function evaluations. Then, for different success rates, the time or the number of function evaluations invested by both sequential and parallel schemes are obtained. Finally, Equations 1.5 and 1.6 are applied to calculate the speedup and the efficiency, respectively, of the parallel schemes considering the different success rates. Once the speedup and efficiency metrics are calculated, scalability analyses can be carried out. Additionally, the procedure presented in Section 1.2.6 is also used to assign statistical confidence to the results.

1.4 Research Questions

“The field of Evolutionary Computation (EC) has grown dramatically and matured into a well-established discipline” [178]. The main reason is that EAs have been applied to a wide range of complex applications in numerous fields of science and technology, providing high-quality results and thus demonstrating their high performance and general applicability. EAs have awoken the interest of many researchers, which has given rise to a large number of still-open research questions [212]. Since the majority of these questions are general, they can be extended to other meta-heuristics. Some of these questions, if answered, would help these optimisation schemes to provide even better results. Furthermore, their application might be easier to put into practice. Most of these open research topics are directly or indirectly related to the problems of premature convergence and parameter setting in EAs.

Due to diversity loss in the population, EAs usually converge towards local optima. It is this loss of diversity that is the main reason for the appearance of premature convergence in EAs. Several methods have been designed in an effort to deal with the problem of premature convergence. One of the most novel proposals is based on applying multi-objective schemes to single-objective optimisation problems, with diversity-based MOEAs being one of the most promising schemes. In these methods, a metric of the diversity introduced by each individual is used as an auxiliary objective, while the original objective function of the optimisation problem being solved is maintained. Some diversity metrics define their own parameters, thus incrementing the number of parameters of the optimisation scheme as a whole. Another drawback of EAs, which is closely related to the problem of premature convergence, is finding their appropriate parameter setting. If the parameter values of an EA are chosen poorly, a loss of diversity might result, giving rise to the appearance of premature convergence. Hence, the performance of an EA, and consequently the quality of the resulting solutions, is highly dependent on these parameters. As a result, it is

essential that the parameters of an EA be properly determined. Finally, even if the parameters were suitably chosen and premature convergence was mitigated, there are optimisation problems for which the time required to obtain high-quality solutions is prohibitive. For this reason, it would be desirable to enable the use of EAs in parallel environments so as to speedup the rate at which high-quality results are obtained for these types of complex applications.

The main aim of this research work is to shed some light on these topics, which are analysed together in this dissertation. Thus, the main research questions to be answered in this thesis are the following:

- Can general diversity-based approaches be designed that avoid the premature convergence of an EA or is it necessary to make use of problem-dependent information in order to deal with this problem?
- Is it possible to provide better performance if these diversity-based methods incorporate the use of parameters?
- Do these extra parameters hinder the setting of an EA?
- Is it desirable to dynamically alter the values for the parameters of an EA during its run instead of prefixing them before the run starts?
- Can general control approaches be designed that simultaneously adapt the numeric and symbolic parameters of an EA?
- Are these control methods applicable to different parameters of an EA?
- How can parameter control strategies and diversity-based methods be enabled for use in parallel environments?
- Are all these techniques suitable for solving complex real-world applications?

1.5 Main Contributions

This thesis mainly deals with the problems of premature convergence and parameter setting in EAs. The main contributions to these fields are the following:

- A set of novel diversity-based objectives are proposed to address the problem of diversity preservation, and consequently the problem of premature convergence in EAs. Additionally, a new diversity-based survivor selection operator is also introduced. Some of the novel diversity-based approaches have parameters

which must be set. This was done by applying the various parameter control methods proposed in this dissertation.

- The different diversity-based objectives, as well as the diversity-based survivor selection operator, are integrated together with several MOEAs to comprise a set of novel diversity-based MOEAs. Very promising results are obtained by applying the proposed diversity-based MOEAs to different single-objective optimisation problems, thus demonstrating the validity of the proposals by mitigating the problem of premature convergence.
- Several novel parameter control schemes based on the use of FLCs are presented in order to address the problem of parameter setting in EAs. The novelty of these schemes lies in the definition of different fuzzy rule bases and on a score function which allows the most promising set of rules to be enabled at every instant during the optimisation process. They have been designed taking into consideration the generality on their application, meaning they can be used to adapt different numeric—discrete and continuous—parameters belonging to different meta-heuristics. In this dissertation, they are used to successfully control different numeric parameters of the diversity-based MOEAs, including the parameters defined for the novel diversity-based objectives.
- Both sequential and parallel hyper-heuristics are also applied herein as parameter control approaches. The parallel hyper-heuristic is based on the use of an island-based model, a well-known parallel EA. In order to analyse their behaviour as parameter control schemes, they are used to adapt different parameters of the diversity-based MOEAs. We demonstrate that both sequential and parallel hyper-heuristics provide high-quality results. Additionally, the scalability and robustness analyses performed on the parallel hyper-heuristic prove its good performance.
- An extensive experimental evaluation to compare the control schemes based on FLCs and hyper-heuristics is carried out. High-quality solutions are obtained for several single-objective optimisation problems by the application of both types of strategies to different diversity-based MOEAs. Moreover, the advantages of dynamically adjusting the parameters of an algorithm during its run instead of executing it with preset values, i.e. the benefits of parameter control over parameter tuning, are also shown.
- A novel hybrid parameter control scheme based on FLCs and hyper-heuristics is proposed. This hybrid control method is able to simultaneously adapt numeric and symbolic parameters. The validity of this proposal is demonstrated

by using it to adapt numeric and symbolic parameters of a diversity-based MOEA applied to a set of well-known benchmark functions.

- Different optimisation problems are considered not only to validate the different aforementioned proposals, but also to show that the resolution of this set of problems is important from a practical point of view. Both benchmark functions and real-world complex applications are addressed, namely:
 - The F1–F19 benchmark functions [220] are a set of scalable continuous optimisation problems, which combine different properties involving the modality, the separability, and the ease of optimisation dimension by dimension, i.e. whether the objective value can be optimised by independently adjusting each variable or not. The proper behaviour of the different proposed control schemes and diversity-based MOEAs is illustrated with this set of functions.
 - The Antenna Positioning Problem (APP) [5, 348] is an optimisation problem that arises in the engineering of mobile telecommunication networks. The APP is defined as the problem of identifying the infrastructure required to establish a wireless network. In this case, high-quality results are reported by using the novel diversity-based MOEAs.
 - The Frequency Assignment Problem (FAP) [222] is an optimisation problem of great importance in the telecommunications field. It is one of the key issues in the design of Global System for Mobile Communications (GSM) networks. The main aim of the FAP is to assign the set of frequencies that must be used in the different antennas of the network in order to minimise the loss of signal quality. For this problem a tailored local search method is defined. The currently best published frequency maps for two real cities are improved upon by combining the various control strategies proposed with the diversity-based MOEAs.
 - The Two-Dimensional Packing Problem (2DPP) is a variant of a packing problem proposed in the Genetic and Evolutionary Computation Conference (GECCO) 2008 competition. The problem definition proposed is different from the traditional definition of a packing problem. However, this problem is complex enough to measure the performance of an optimisation scheme. For this problem, a tailor-made local search method is also defined. As in the case of the previous optimisation problem, the currently best published results for several instances are improved upon combining the different control techniques proposed with

the diversity-based MOEAs.

- Metaheuristic-based Extensible Tool for Cooperative Optimisation (METCO) [206] is a framework that supports the implementation and execution of sequential and parallel meta-heuristics. Its functionality can be extended by defining new plugins. Since in this thesis some of the above algorithmic proposals and optimisation problems are integrated as novel plugins into METCO, the contribution to the improvement of this tool is significant.

1.6 Synopsis

The contents of this dissertation are grouped into five main parts:

- *Part I: Foundations, Background, and Contributions.* The aim of this part is to establish the basic notions and nomenclature used over the course of this dissertation, as well as to provide some background into the main topics discussed, and to enumerate the contributions of the research work contained herein. Particularly, Chapter 2 describes the set of EAs that have been applied in this thesis. Then, Chapter 3 and Chapter 4 describe the state of the art on the application of multi-objective schemes as single-objective optimisers, and on parameter setting in EAs.
- *Part II: Algorithmic Proposals.* The contents of this part focus on the novel schemes that are proposed in this work. Firstly, Chapter 5 introduces the novelties of the diversity-based MOEAs designed for single-objective optimisation. Lastly, Chapter 6 exposes the design of the novel parameter control strategies based on FLCs and hyper-heuristics, including the hybrid parameter control method based on these two approaches.
- *Part III: Validation of the Algorithmic Proposals: Benchmark Problems and Complex Applications.* This part presents the experimental evaluation conducted on the algorithmic proposals so as to analyse their behaviour with both benchmark problems and complex applications. Chapter 7 describes the experimental evaluation with the benchmark functions, whereas Chapters 8, 9, and 10 respectively do the same for the APP, the FAP, and the 2DPP.
- *Part IV: Conclusions and Future Lines of Research.* The aim of this part is to share the main conclusions extracted from this research. Moreover, the most promising lines of future work are discussed.

- *Part V: Appendices.* This last part contains an appendix with the set of publications that emerged from the different topics considered in this dissertation, as well as an appendix with the complete set of rule bases defined for the FLCs proposed herein.

Evolutionary Algorithms

This chapter focuses on the description of the family of meta-heuristics that are applied throughout this dissertation: Evolutionary Algorithms (EAs). This chapter starts with a brief historical introduction to these types of algorithms. Then, the common concepts and features shared by all EAs are discussed. Afterwards, the different approaches used herein for both the single-objective and the multi-objective fields are presented, including the definition of MAs, which are considered hybrid meta-heuristics based on the combination of a population-based meta-heuristic, such as EAs, and a LS procedure. Finally, one of the most frequently used models for defining parallel EAs, the island-based model, is explained.

2.1 A Brief Historical Introduction

Early work on EC was based on the application of evolutionary systems such as optimisation methods, feedback-control approaches, or automatic tools to generate computer programs. In this dissertation, evolutionary system refers to Darwinian evolutionary systems, and consists of the following features [179]:

- There exists a population of individuals which compete for limited resources.
- Due to the birth and death of individuals, the population changes dynamically.
- The ability of an individual to reproduce and survive is called fitness.
- The fact that offspring are similar, but not identical, to their parents is known as variational inheritance.

One of the first studies that influenced the emergence of EC was provided by Sewall Wright in the 1930s [352]. Wright explained evolution as a process where the genetic information of species is continuously changing by a trial and error mechanism. The effect of this procedure is to optimise the adaptation of species to their environment. During the same period, Walter Cannon also noted that natural evolution was a learning process that proceeds by random trial and error [58]. Nevertheless, it was not until the late 1950s that Hans Bremermann proposed the relationship between evolution and mathematical optimisation [40, 41]. Hence, Bremermann can be considered as one of the first practitioners of EC in the field of optimisation. Nowadays, optimisation continues to be the largest area of application of EC. Other early works were based on the usage of evolutionary systems as feedback-control approaches, which altered their responses dynamically depending on the feedback obtained from the system being controlled. From this point of view, an example was the usage of an evolutionary system to evolve the control circuits of a robot [126]. Finally, evolutionary systems were also used in an attempt to generate computer programs automatically. An example of this type of application was the design of a “*Learning Machine*” that evolved sets of machine language instructions [125]. All this early research work was the origin of several sub-areas of EC, like EP and GAs. A deeper insight into the origins of EC can be found in [120].

In the 1960’s, the availability of “cheap” computers boosted the idea of implementing evolutionary systems as computational algorithms to solve complex problems, thus triggering a revolution in the field of EC. Among the most important contributions were the ideas put forth by Rechenberg and Schwefel, Fogel, and Holland:

- Rechenberg and Schwefel formulated some ideas on how evolutionary processes could be used to deal with real-valued optimisation problems [283]. ES were born from these ideas.
- Fogel’s research work was based on applying the ideas of evolutionary processes to the field of artificial intelligence [122]. His first works were based on representing intelligent agents as finite state machines, which were evolved to even more intelligent agents. His ideas allowed EP methods to emerge.
- Holland started to apply evolutionary processes so as to implement problem-independent robust adaptive systems, whose decisions were carried out by taking into consideration the feedback received from interacting with a particular environment [159]. These ideas set the stage for the modern-day GAs.

Afterwards, the aim of most research activities during the 1970’s was twofold. Firstly, to characterise the behaviour of the three aforementioned approaches, and

Table 2.1: Implementation details for the canonical evolutionary algorithms

	EP	ES	GAs
Parent pop. size	$\mu = N$	$\mu = 1$	$\mu = N$
Offspring pop. size	$\lambda = N$	λ	$\lambda = N$
Representation	Finite State Machines	Real-valued vectors	Binary string
Crossover	-	-	One-Point
Mutation	Add/Delete states/arcs	Adaptive Gaussian	Bit Flip
Parent Selection	Deterministic	-	Random
Survivor Selection	N fittest	$(1 + \lambda)$	Generational

secondly, to better understand the way they could be used to solve problems. Decisions on how best to implement these approaches were also investigated, such as how to manage the population of individuals, or how to select the parents in order to obtain the offspring, among other topics. As a result, three canonical EAs—EP, ES, and GAs—resulted from all the research work performed during this decade. The implementation differences among these canonical EAs are shown in Table 2.1.

In the case of EP, starting from a population of N parents, N offspring were generated with each iteration of the algorithm. The fittest N individuals among the parents and offspring were selected as the parent population for the next iteration of the algorithm. The selection of the parents was deterministic, and each parent evolved in order to produce a single offspring. Since the individuals were represented by finite state machines, the offspring were obtained by the usage of a mutation operator that was responsible for adding/deleting states/arcs.

With regard to ES, individuals were represented by a vector of real variables, because this approach originated with the aim of optimising real-valued functions. The original proposal was named $(1 + \lambda)$ -ES. In each iteration of the algorithm, λ offspring was created starting from a single parent, and the fittest individual between the parent and the offspring was selected as the unique parent for the next iteration. In order to obtain the offspring, an adaptive mutation operator based on a Gaussian distribution with a mean equal to zero and a standard deviation equal to σ — $G(0, \sigma)$ —was applied. So as to deal with this adaptive mutation operator, the representation consisted of D real variables of the problem and D real values for the standard deviation σ . Each value σ_i , $i = 1, \dots, D$ was used in order to perturbate the variable i through a Gaussian distribution $G(0, \sigma_i)$, and since these values were incorporated into the representation of the individual, they changed during the optimisation procedure. To modify these values, the famous “ $1/5$ rule” proposed by Rechenberg [284] was used. This is why these EAs are called Evolution Strategies,

since the strategy parameters are evolved together with the individuals.

In GAs, the individuals were represented by means of a binary string. Variation operators—crossover and mutation—were applied to obtain N offspring starting from N parents, and all offspring survived as the parent population for the next iteration of the algorithm, i.e. no parents were selected to survive. This is known as a *generational* survivor selection method. To generate the offspring, parents were randomly selected, but proportionately to their fitness.

While the canonical EAs emerged from the 1970's, the 1980's were focused on applying this set of algorithms to more complex problems, and on defining new algorithmic proposals based on them. In the particular case of ES, different problems arose when dealing with high-dimensional and multi-modal problems. Consequently, several variants of the original ES algorithm were proposed, like the $(\mu + \lambda)$ -ES, which generates λ offspring starting from μ parents, and selects the best μ from $(\mu + \lambda)$ individuals as the parent population for the next iteration. Another proposal was the (μ, λ) -ES, in which μ parents produce λ offspring, and the best μ individuals to survive are selected from the λ offspring, i.e. no parents survive. In the case of GAs, something similar happened, since they were widely applied to different optimisation problems. The most common difficulties that emerged from the application of GAs involved the convergence to global optima and stagnation in sub-optimal regions. As a result, different solutions were proposed, such as the use of *tournament selection* as a novel parent selection mechanism [141], or the use of *elitism* in the survivor selection methods [88]. Another important topic of research was the modification of the internal representation of individuals for GAs, by experimenting with other binary representations, or by changing the original representation based on a binary string for a real-valued representation. Moreover, some research topics remained open, such as the application of GAs to other fields, like classifier systems or neural networks, the definition of non-linear representations of variable size in order to address other types of problems with GAs, or even the parallelisation of GAs. Finally, a large amount of tailor-made EAs, which incorporated problem-dependent information in the form of representations or operators specifically designed for the problem at hand, were also proposed during this decade.

Until the 1990's the research on ES, EP, and GAs had been carried out independently. However, in the late 1980's and early 1990's a significant number of conferences on this topic were held that allowed different research groups to exchange their ideas and points of view. As a consequence, an agreement on the term Evolutionary Computation was reached, and the first journal on the subject, also called *Evolutionary Computation*, emerged. Due to this exchange of ideas, different modifications

Table 2.2: Evolutionary process versus resolution of an optimisation problem

Evolutionary process	Resolution of an optimisation problem
Population	Set of solutions
Individual	Solution
Fitness	Objective function
Environment	Optimisation problem
Genotype, Phenotype, Chromosome	Internal representation of a solution
Gene	Decision variable
Allele	Value of a decision variable
Locus	Position of a decision variable

at the implementation level were investigated for the canonical EAs. In the case of ES, different crossover operators that improved the performance of this EA were proposed [20]. With regard to GAs, the use of problem-dependent representations and operators improved the behaviour of these types of algorithms [247]. Finally, with the incorporation of internal representations and operators from the field of ES, the application of EP was extended to other areas such as optimisation. As a result, several novel EAs, like GP [190], were born. Moreover, many of the fundamental assumptions and underlying theories were revised in order to strengthen and generalise the basis for the different paradigms based on EAs. Finally, by the late 1990's the field of EC had become a mature scientific discipline.

2.2 Basic Concepts for Evolutionary Algorithms

In the previous section some features of EAs were introduced indirectly. This section provides a detailed discussion of the basic terms and concepts that different variants of these types of approaches share. EAs are a family of population-based meta-heuristics inspired by natural evolution. They have been successfully applied to a large number of complex applications in different scientific and technological fields. In natural evolution, a given environment is filled with a population of individuals that compete in order to survive and reproduce. Hence, since the environment can only contain a limited number of individuals, survivor selection mechanisms are required to keep the population from growing uncontrollably. Natural selection promotes the most competitive individuals, i.e. those that adapt better to the environment. This phenomenon is also known as *the survival of the fittest*. Some of these survivors will be able to reproduce with the aim of obtaining even fitter individuals.

Algorithm 1 Generic pseudocode for an evolutionary algorithm

- 1: **Initialisation.** Generate the initial parent population.
 - 2: **Evaluation.** Evaluate all individuals in the initial parent population by applying the objective function in order to assign a fitness value to every individual.
 - 3: **while** (not stopping criterion) **do**
 - 4: **Parent selection.** Select the individuals from the parent population to build the mating pool.
 - 5: **Variation.** Apply the variation operators to the mating pool so as to create the offspring population.
 - 6: **Evaluation.** Evaluate the generated offspring via the objective function so as to assign a fitness value to every offspring.
 - 7: **Survivor selection.** Select individuals from among the parents and offspring to survive as the new parent population for the next generation.
 - 8: **end while**
-

Continuing with metaphor between an evolutionary process and the resolution of an optimisation problem—Table 2.2—a *structure* or an *individual* is an encoded solution to some problem. The codification of an individual or its internal representation is known as the *genotype*, which is decoded to obtain the *phenotype*, or decoded solution. If a *direct encoding* of the individuals is used, the genotype and the phenotype are similar. A genotype is usually composed of one or more *chromosomes*, and every chromosome in turn consists of several independent *genes*, which take on certain values or *alleles*. A *locus* identifies the position of a gene within the chromosome, and a set of chromosomes is called a *population*. Generally, an EA requires both an objective function and a fitness function. The objective function is located in the problem domain, and defines the optimality condition of the EA. In contrast, the fitness function is situated in the algorithm domain, and measures if a particular solution satisfies said optimality condition. Both functions, however, are usually identical [70], at least in the single-objective case. Hence, an objective function assigns a fitness value to every individual. This fitness value measures the ability of the corresponding individual to survive and reproduce in the *environment*, i.e. whether the solution is appropriate for the optimisation problem at hand. We should note that in this dissertation a direct encoding of the individuals is always used. In addition, genotypes consist of a unique chromosome. Consequently, the terms genotype and chromosome refer to the internal representation of an individual.

As was stated in previous sections, several EAs exist. However, they share the same generic framework. Algorithm 1 shows the generic pseudocode of an EA. During the initialisation stage—line 1—individuals are generated to fill the initial parent population. This initial parent population is evaluated—line 2—through the

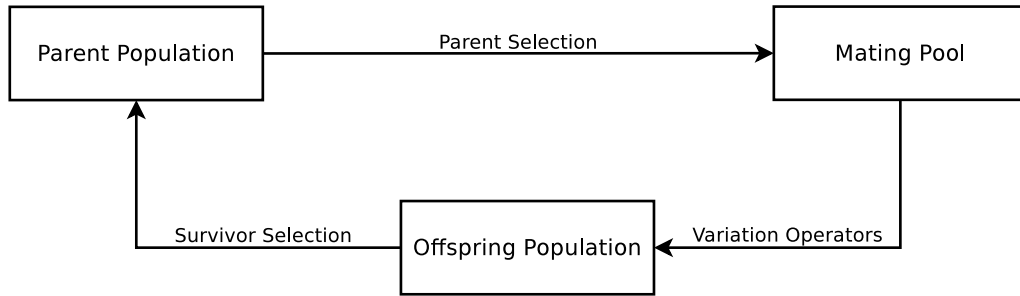


Figure 2.1: Flow chart representing an evolutionary algorithm generation

application of the objective function so as to assign a fitness value to every individual. Then, at each iteration or *generation* of the EA, a set of steps is repeated. Firstly, the parents that comprise the *mating pool*—line 4—are selected via the *parent selection* or *mating selection* mechanism. Then, the *variation operators* are applied to the mating pool to generate the offspring population—line 5. Particularly important among the different variation operators are the *recombination* operator—or *crossover* operator—and the *mutation* operator. Once the offspring are obtained, they have to be evaluated—line 6—by means of the objective function in order to assign them a fitness value. Finally, a *replacement* or *survivor selection* operator is applied—line 7—in order to determine the set of individuals from among the parents and the offspring that are going to survive as the parent population for the next generation. These four steps are repeated until a stopping criterion—line 3—is satisfied. A flow chart representing a generation of an EA is shown in Figure 2.1.

From the above content, it can be observed that different components, such as the survivor selection strategy or the variation operators, among others, have to be specified in order to completely design an EA. Additionally, some of these components incorporate the use of parameters—mutation and crossover rates, or the population size, for instance—and consequently they must be also fixed in order to execute the EA as designed. In particular, decisions regarding the following components must be made during the design of an EA:

- *Individual's representation, genotype, or chromosome.* The structure which is going to be used in order to represent a solution to the problem being solved is a common component not only among different EAs, but also for all meta-heuristics. In the case of EAs, the method for decoding the genotype into the phenotype must also be considered at this point. In this dissertation, three main representations are used depending on the problem at hand: binary string, vector of discrete values—integer values, for instance—and vector of

real values. Additionally, since a direct encoding is always used, it is not necessary to specify a method to transform genotypes into phenotypes.

- *Population initialisation.* This component is also shared by all population based meta-heuristics, and therefore by all EAs. In this thesis, the initial population is always filled with randomly generated individuals.
- *Parent selection mechanism.* This is a component that must be selected exclusively for EAs. This method is responsible for selecting the individuals from the parent population for the purpose of reproducing. It is important to remark that in this research work, the well-known *Binary Tournament* operator [108] is always used as the parent selection strategy. Its operation is detailed in Section 2.2.1.
- *Variation operators.* As in the previous case, the variation operators are components that are specifically taken into account during the design of an EA. The objective of the variation operators is to generate offspring starting from the mating pool. The most widely applied variation approaches are the crossover and mutation operators. The different crossover and mutation operators that are used throughout this thesis are described in Sections 2.2.2 and 2.2.3.
- *Survivor selection strategy.* This is another specific component whose proper selection must be determined during the design of an EA. The survivor selection mechanism is responsible for choosing, from among the current parents and offspring, the individuals which survive for the next generation. Section 2.2.4 is devoted to describing the particular survivor selection strategies addressed herein.
- *Stopping criterion.* This component is also common to all meta-heuristics. As was stated in the previous chapter, in this thesis, two main stopping criteria are considered: execution time and number of evaluations carried out involving the objective functions of the problem being solved.

2.2.1 Parent Selection Mechanisms

The main aim of the parent selection strategy is to select the individuals from the parent population that are going to reproduce in order to generate the offspring. Usually, these strategies are based on the fitness assigned to every individual. Hence, the better the fitness of an individual, the greater its likelihood of being selected. This is known as the *selection pressure*, and it is used to guide the search procedure

towards better individuals. The higher the selection pressure, the higher the probability that the fittest individuals survive. Nevertheless, some individuals with a poor fitness should be considered for their selection, because they might contribute with their genetic material to guide the search process to unexplored areas where the global optimum could be found. The fitness can be assigned in two different ways:

- *Direct fitness assignment.* The fitness values are directly associated with individuals.
- *Rank-based fitness assignment.* Each individual in the population is assigned a rank, with this rank generally depending on its fitness value. For instance, suppose a list in which individuals are sorted in descending order depending on their fitness values. Hence, an individual is associated with its corresponding rank in this list.

There exist different types of parent selection mechanisms. Some of them are based on a direct fitness assignment, while other approaches are based on a rank-based fitness assignment in order to carry out the selection. The most important ones are *Fitness Proportional selection*, also called *Roulette Wheel selection*, *Stochastic Universal Sampling (SUS)*, *Rank-based selection*, and *Tournament selection* [108].

In this dissertation, a tournament selection strategy, which selects one individual from the parent population, is always applied. If n parents have to be selected, the operator has to be applied n times. Tournament selection consists of two main steps. Firstly, k individuals are randomly selected from the current parent population using a uniform distribution. This random selection can be performed with replacement or without replacement. If no replacement is considered, the k selected individuals are discarded from subsequent tournaments, whereas with replacement, the k selected individuals might be randomly selected again in future tournaments. In the second step, a probability p is used to determine the winner of the tournament from among the k possible candidates. The value of p represents the probability that the fittest individual from among the k possible candidates will win the tournament. If a deterministic tournament is carried out— $p = 1$ —the fittest individual, i.e. the one with the best fitness, is always selected as the winner of the tournament. However, stochastic variants of this selection operator can be defined by letting $p < 1$. Particularly, in this dissertation a deterministic *binary tournament*— $k = 2$ —is always applied as the parent selection strategy [108]. Whether replacement is used or not depends on the specific EA. Figure 2.2 shows its operation. Each circle represents an individual and the corresponding number refers to its fitness value. In this example, the higher the fitness value, the fitter the individual.

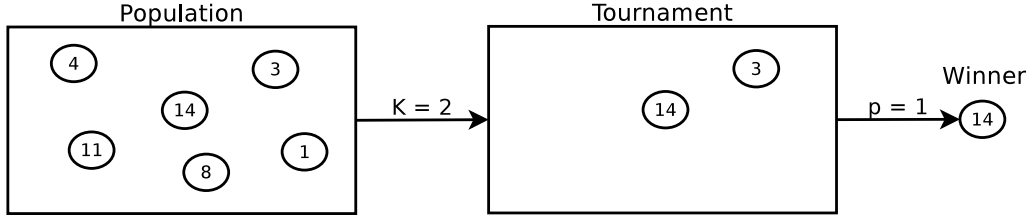


Figure 2.2: Parent selection based on a deterministic binary tournament

One of the main benefits of tournament selection is that it does not require any global knowledge about the population, since it is based on a direct fitness assignment. As a result, it is conceptually simple and easy to implement. Additionally, the selection pressure can be controlled by means of the tournament size k . The larger the tournament size, the higher the number of randomly picked individuals, and therefore the lower the probability that an individual with a poor fitness will be selected. Thus, as the tournament size increases, the selection pressure grows.

2.2.2 Crossover Operators

Recombination or crossover operators are responsible for generating new offspring by the inheritance of features belonging to different parents. Crossover operators are usually binary, i.e. one or more offspring are obtained from two parents. However, there exist variants of n -ary crossovers, in which n parents are used to produce the offspring. Since crossover operators are applied to the chromosome of an individual, the internal representation selected mainly determines the way in which a particular crossover operator is designed. The most important aspects that a crossover operator has to address are [326]:

- *Inheritance.* The crossover operator should be able to promote the inheritance of features that belong to both parents.
- *Validity.* The crossover should be able to produce valid offspring, i.e. solutions that belong to the feasible region of the optimisation problem in question.

In this dissertation different binary crossover operators are applied with a probability $p_c \in [0, 1]$, also called *crossover rate*. These operators are as follows:

- *Uniform Crossover (UX)* [323]. This crossover works by treating every gene independently and randomly selecting from which of the two parents a particular gene is inherited. To do so, first a vector $R = (r_1, \dots, r_D)$ with D random

values $r_i \in [0, 1]$ is generated using a uniform distribution, where D is the number of genes in a chromosome. Second, if r_i is below a given threshold—in general, 0.5—the gene i of the offspring is inherited from the first parent; otherwise it is inherited from the second parent. This second step is performed for every gene of the offspring. In order to generate the second offspring, an inverse mapping is used. Figure 2.3 shows the above process. This crossover operator is suitable for binary string, real-valued, and discrete chromosomes.

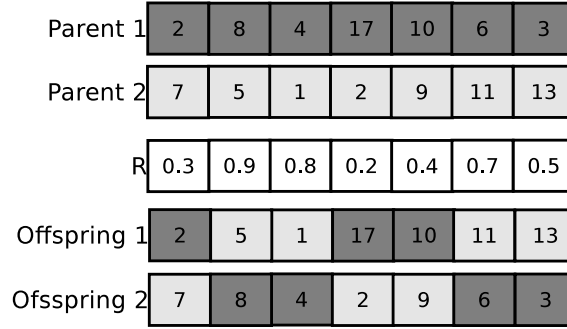


Figure 2.3: Operation of the Uniform Crossover

- *One Point Crossover (OPX)* [160]. First, this operator randomly chooses a locus from the chromosome using a uniform distribution. Both parents are then split at this point, thus yielding two offspring by the exchange of the parents' tails. This operation is shown in Figure 2.4. It can be used with binary string, real-valued, and discrete chromosomes. However, its main drawback is the disruptive effect that it can have.

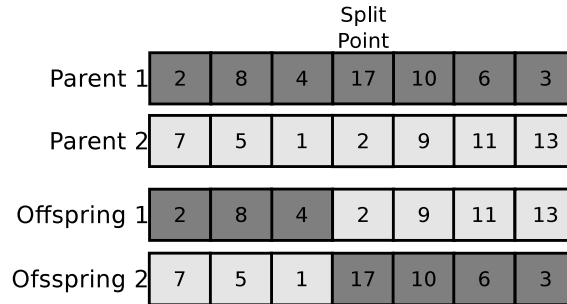


Figure 2.4: Operation of the One Point Crossover

- *Simulated Binary Crossover (SBX)* [89]. This operator is classified as a *parent-centric* crossover. With parent-centric crossover operators, offspring are generated closer to their parents. Particularly, this operator simulates the opera-

tion of the OPX operator when it is applied to binary string representations. However, the SBX operator was specifically designed to deal with real-coded chromosomes. It is based on a probability distribution function that can be adapted to the problem at hand by means of the distribution index η :

$$P(\beta) = \begin{cases} 0.5 \cdot (\eta + 1) \cdot \beta^\eta & \text{if } \beta \leq 1 \\ 0.5 \cdot (\eta + 1) \cdot \frac{1}{\beta^{\eta+2}} & \text{if } \beta > 1 \end{cases} \quad (2.1)$$

A value for β_i is obtained so that the area under the probability curve is equal to a randomly generated number μ_i in the range $[0, 1]$ using a uniform distribution. Hence, β_i can be calculated as follows:

$$\beta_i = \begin{cases} (2 \cdot \mu_i)^{\frac{1}{\eta+1}} & \text{if } \mu_i \leq 0.5 \\ \left[\frac{1}{2 \cdot (1 - \mu_i)} \right]^{\frac{1}{\eta+1}} & \text{if } \mu_i > 0.5 \end{cases} \quad (2.2)$$

Finally, if $X = (x_1, \dots, x_D)$ and $Y = (y_1, \dots, y_D)$ are the parents, $V = (v_1, \dots, v_D)$ and $Z = (z_1, \dots, z_D)$ are the offspring, and D is the number of genes in a chromosome, the values of the genes v_i and z_i are calculated as follows:

$$\begin{aligned} v_i &= 0.5 \cdot [(1 + \beta_i) \cdot x_i + (1 - \beta_i) \cdot y_i] \\ z_i &= 0.5 \cdot [(1 - \beta_i) \cdot x_i + (1 + \beta_i) \cdot y_i] \end{aligned} \quad (2.3)$$

Note that v_i and z_i might be outside the range $[a_i, b_i]$, which delimits the possible values that can be assigned to the gene situated at position i . In this case, v_i and z_i have to be confined to this range. The above process is repeated for every gene assuming a probability equal to 0.5. Therefore, there will be cases in which the genes of both offspring directly inherit the alleles of the parents' genes without performing the aforementioned procedure. Finally, we note that in this dissertation a distribution index $\eta = 5$ is always used.

- *Arithmetical Crossover (AX)* [248]. This operator is classified as a *mean-centric* crossover, where offspring are created closer to the centroid of their parents. The AX operator was designed to work with real-valued chromosomes. Its operation is as follows. First, a value w in the range $[0, 1]$ is randomly generated by using a uniform distribution. Then, if $X = (x_1, \dots, x_D)$ and $Y = (y_1, \dots, y_D)$ are the parents, and D is the number of genes in the

chromosomes, the offspring $V = (v_1, \dots, v_D)$ and $Z = (z_1, \dots, z_D)$ are produced by applying Equation 2.4.

$$\begin{aligned} v_i &= w \cdot x_i + (1 - w) \cdot y_i \\ z_i &= w \cdot y_i + (1 - w) \cdot x_i \end{aligned} \tag{2.4}$$

- *Parent-centric Blend Crossover (PBX- α)* [219]. As its name indicates, this operator is classified as a parent-centric operator, and therefore it is suited to real-valued chromosomes. Although it was proposed to yield a unique offspring, in this research work it is modified so as to produce two offspring. The original version produces an offspring in the following manner. If D is the number of genes of a chromosome, $X = (x_1, \dots, x_D)$ is the female parent, and $Y = (y_1, \dots, y_D)$ is the male parent, this operator generates the offspring $Z = (z_1, \dots, z_D)$ where z_i is a randomly selected value, using a uniform distribution, from the range $[l_i, u_i]$ with $l_i = \max\{a_i, x_i - I \cdot \alpha\}$, $u_i = \min\{b_i, x_i + I \cdot \alpha\}$, and $I = |x_i - y_i|$. Moreover, a_i and b_i respectively are the lower and upper bounds which delimit the possible values that can be assigned to the gene located at position i . It is important to note that the value of the parameter α allows the spread associated with the probability distribution to be altered, which usually takes values from the range $[0.5, 1]$. In this dissertation a value $\alpha = 0.5$ is always applied. Finally, in order to generate two offspring instead of producing a single one as the original proposal does, the second offspring is generated by exchanging the roles of the parents, i.e. X becomes the male parent and Y becomes the female parent.
- *Two-Dimensional Sub-String Crossover (SSX)* [161]. This operator is suitable for two-dimensional chromosomes based on real, discrete, or binary values. Its operation is as follows. First, a locus within the two-dimensional chromosome is selected as the division point. Second, the operator randomly decides to carry out a vertical or a horizontal recombination. Hence, the two-dimensional chromosome is transformed into a one-dimensional chromosome by sorting the genes by rows or by columns. Lastly, the OPX operator is applied considering the one-dimensional transformed chromosome and the selected locus. The operation of this crossover operator is graphically shown in Figure 2.5. Note that the offspring $H1$ and $H2$ are generated through a horizontal recombination, whereas the offspring $V1$ and $V2$ are produced by the application of a vertical recombination. For both cases, the locus $(3, 2)$ has been selected as the division point. This crossover operator might have a disruptive effect.

Parent1			
1	3	9	8
5	4	7	2
6	12	11	10

Parent2			
4	6	11	9
10	1	5	3
2	12	7	8

H1			
1	3	9	8
5	4	5	3
2	12	7	8

H2			
4	6	11	9
10	1	7	2
6	12	11	10

V1			
1	3	9	9
5	4	5	3
6	12	7	8

V2			
4	6	11	8
10	1	7	2
2	12	11	10

Figure 2.5: Operation of the Two-Dimensional Sub-String Crossover

Finally, crossover operators which have been specifically designed to deal with some of the optimisation problems addressed herein are also applied. They will be described in the corresponding chapters on optimisation problems.

2.2.3 Mutation Operators

Besides the crossover operator, another widely applied variation approach can be used in the form of a mutation operator. The main difference with crossover operators is that mutation operators are unary, i.e. they produce a single offspring starting from a unique parent. In general, p_m refers to the mutation probability applied to every gene in a chromosome, and it is also called *mutation rate*. p_m might also refer to the mutation probability for a unique gene, however. Mutation operators are used to effect small changes in the selected individuals. This is why they are usually applied with low values for p_m . Otherwise, mutation operators might produce a highly disruptive effect. One of the most common practices is to assign the value $1/D$ to the probability p_m , D being the number of genes in a chromosome. In this way, a single gene is mutated on average. The most important aspects which must be considered during the design of a mutation operator are [326]:

- *Ergodicity*. A mutation operator should be able to produce every solution from the search space.
- *Validity*. The offspring generated should be a valid solution belonging to the feasible region of the search space.

- *Locality*. Locality refers to the effect produced on the phenotype when the genotype is altered. If small changes are performed at the genotype level, small changes should be produced at the phenotype level. This is also known as strong locality. In contrast, if a small change is produced in the genotype that results in a large modification in the phenotype, then the locality is weak. Strong locality is a desirable feature in a mutation operator. In cases where a direct encoding is used, locality is ensured if small changes are carried out by the mutation operator.

In this thesis the following mutation operators are applied:

- *Bit flip mutation* [108]. This mutation operator, which is suitable for binary string chromosomes, allows an offspring to be obtained by flipping the alleles of the parent's genes. Every gene in the parent is mutated with a probability p_m . In cases where the gene of the parent takes the value zero, the corresponding gene of the offspring will be assigned a one, and vice-versa. An example is graphically shown in Figure 2.6. Only two bits are mutated due to p_m .

Parent	1	0	0	1	0	1	1
Offspring	1	0	1	1	0	0	1

Figure 2.6: Operation of the bit flip mutation

- *Uniform Mutation (UM)* [108]. The UM operator was specifically designed to deal with real-valued representations. Given a gene x_i of the parent $X = (x_1, \dots, x_D)$, where D is the number of genes in the chromosomes, it is mutated to obtain the gene z_i of the offspring $Z = (z_1, \dots, z_D)$ as follows. In the first place, a value $\mu_i \in [0, 1]$ is randomly generated by means of a uniform distribution. Afterwards, the mutated gene z_i of the offspring is given by:

$$z_i = \mu_i \cdot (b_i - a_i) + a_i \quad (2.5)$$

In Equation 2.5, a_i and b_i respectively are the minimum and maximum values that can be assigned to the gene i . It is important to note that two different variants of the UM operator are applied in this dissertation. The first one mutates every parent gene with probability p_m , while the second version mutates a unique gene with probability p_m .

- *Polynomial Mutation (PM)* [90]. As in the case of the UM operator, the PM operator is also appropriate for real-valued chromosomes. Given a parent

$X = (x_1, \dots, x_D)$ where D is the number of genes in a chromosome, the gene x_i is mutated to obtain the gene z_i belonging to the offspring $Z = (z_1, \dots, z_D)$ as follows:

$$z_i = x_i + (b_i - a_i) \cdot \delta_i \quad (2.6)$$

In the above Equation, a_i and b_i are the lower and upper bounds that determine the possible values that can be assigned to the gene i . Moreover, the value of δ_i can be calculated from the polynomial probability distribution:

$$P(\delta) = 0.5 \cdot (\eta + 1) \cdot (1 - |\delta|)^\eta \quad (2.7)$$

$$\delta_i = \begin{cases} (2 \cdot \mu_i)^{\frac{1}{\eta+1}} - 1 & \text{if } \mu_i < 0.5 \\ 1 - [2 \cdot (1 - \mu_i)]^{\frac{1}{\eta+1}} & \text{if } \mu_i \geq 0.5 \end{cases} \quad (2.8)$$

In Equations 2.7 and 2.8, μ_i is a random number that is generated in the range $[0, 1]$ using a uniform distribution, and η denotes the distribution index. We should note that two different versions of the PM operator are applied herein. The first variant mutates every parent gene with probability p_m , whereas the second variant mutates a unique parent gene with probability p_m . Finally, a distribution index $\eta = 20$ is always used with both versions.

Other mutation operators that have been specifically designed to deal with some of the optimisation problems addressed here are also applied. They will be described in the corresponding chapters devoted to optimisation problems.

2.2.4 Survivor Selection Methods

As its name indicates the survivor selection scheme is responsible for choosing the individuals from among the parents and the offspring that will form a part of the parent population for the next generation. The different survivor selection methods that are applied in this research work are the following:

- *Steady-State Survivor Selection (SS-S)*. In the case of this survivor selection strategy, a single offspring is generated in every generation. If no individual from the parent population is worse than the new offspring, the latter is

discarded. In contrast, if several individuals belonging to the parent population are worse than the new offspring, the worst one is replaced by the new offspring.

- *Elitism-based Generational Survivor Selection (GEN-S)*. In this dissertation a variant of the survivor selection scheme incorporated into the GA proposed by Holland [159] is used. In every generation, $N - 1$ offspring are generated starting from a parent population with N members. Then, all parents, except the fittest one, are discarded and replaced by the new offspring.
- *Replace Worst Survivor Selection (RW-S)*. This survivor selection mechanism selects the N fittest individuals from among N parents and N new offspring produced in every generation. This survivor selection approach was the one used in the EP algorithm proposed by Fogel [122]. If ES-like nomenclature is used, then we note that the RW-S scheme is a $(\mu + \lambda)$ selection method, where $\mu = \lambda = N$.

When considering these three schemes, we see that all of them allow *overlapping-generation* EAs to be implemented. This means that survivors are selected from among parents and offspring, instead of only choosing survivors from the offspring population, as is the case with *non-overlapping-generation* EAs [179]. Examples of non-overlapping EAs would be the traditional GA based on a pure generational survivor selection operator or the (μ, λ) -ES, among others. The selection pressure in an overlapping-generation EA is much higher than in the non-overlapping-generation version of the same EA. This is because in the former EA, the number of possible candidates to be selected is usually larger than in the latter EA. Additionally, as the evolutionary procedure progresses, the individuals become increasingly competitive in overlapping-generation EAs.

2.3 Single-objective Evolutionary Algorithms

This section describes the EAs, and particularly, the GAs that are applied as single-objective optimisers throughout this dissertation.

2.3.1 Genetic Algorithms

As was mentioned in Section 2.1, GAs originated as problem-independent adaptive systems [159]. In early versions of GAs, individuals were represented by binary

string chromosomes. This kind of representation was independent from the problem at hand. Additionally, problem-independent crossover and mutation operators were applied in order to obtain the offspring in every generation, with probabilities p_c and p_m , respectively. The most frequently used crossover operator was the OPX, while the mutation operator was based on a bit flip. Parents were selected by a fitness proportional selection method, and only offspring survived for the next generation, since a generational selection strategy was used.

However, some components that were originally proposed for other EAs have been incorporated into GAs over the years, due to a convergence of ideas in the different sub-areas belonging to EC. As a result, for instance, it is not unusual to use a real-valued chromosome to represent the individuals nowadays. Consequently, crossover and mutation operators specifically designed to deal with this type of chromosome are also applied. Furthermore, other types of parent and survivor selection schemes are usually incorporated into GAs.

In this dissertation, GAs are based on some of the different components described in the last section. Their operation is shown in Algorithm 2. During the initialisation stage—line 1—the initial parent population is randomly generated. This initial population is then evaluated by using the objective function—line 2—so as to assign a fitness value to every individual. Afterwards, as long as the stopping criterion of the GA is not satisfied—line 3—the following steps are carried out in every generation. Firstly, M offspring must be generated—lines 4–9. To do so, three different steps have to be carried out for as long as the offspring population is not filled with M new individuals—line 5. During the first step—line 6—a deterministic binary tournament selection with replacement is used as the parent selection strategy in order to choose two parents from the current parent population. It is important to note that both parents might be the same individual. In the second step—line 7—the crossover operator is applied with probability p_c to both parents in order to generate two new offspring. If the crossover operator is not used, the two parents become the two new offspring. Afterwards, in the third and last step—line 8—the mutation operator is applied with probability p_m to both offspring. Therefore, some offspring will be produced by the application of the crossover and mutation operators, whereas other offspring will be generated only by the application of the mutation operator. Once the offspring population is generated, if M is odd, the worst individual from the offspring population is discarded, since $M + 1$ offspring have been produced—lines 10–13. Then, the offspring population is evaluated—line 14—by means of the objective function to assign a fitness value to every offspring. Finally, the survivor selection mechanism is applied to parents and offspring—line 15—in order to determine the parent population for the next generation.

Algorithm 2 Pseudocode of the different single-objective genetic algorithms

- 1: **Initialisation.** Create the initial parent population by filling it with N randomly generated individuals.
 - 2: **Evaluation.** Evaluate all individuals in the initial parent population by applying the objective function in order to assign a fitness value to every individual.
 - 3: **while** (not stopping criterion) **do**
 - 4: **Offspring population generation:**
 - 5: **while** (offspring population is not filled with M individuals) **do**
 - 6: **Parent selection.** Apply deterministic binary tournament selection with replacement on the current parent population in order to select two parents.
 - 7: **Recombination.** Apply the crossover operator with probability p_c to both parents in order to produce two offspring. If crossover operator is not applied, both parents become the two new offspring.
 - 8: **Mutation.** Apply the mutation operator with probability p_m to both generated offspring.
 - 9: **end while**
 - 10: **Offspring population truncation:**
 - 11: **if** (M is odd) **then**
 - 12: Discard the worst individual from the offspring population.
 - 13: **end if**
 - 14: **Evaluation.** Evaluate the M generated offspring by means of the objective function so as to assign a fitness value to every offspring.
 - 15: **Survivor selection.** Select individuals from among N parents and M offspring that will constitute the parent population for the next generation.
 - 16: **end while**
-

Three different variants of the aforementioned GA are proposed in this research work. Each of them is based on one of the three survivor selection strategies exposed in Section 2.2.4. The choice for the remaining components, i.e. the parent population size N , the internal representation of individuals, the crossover and mutation operators, and their rates of application— p_c and p_m —depends on the optimisation problem in question. Additionally, the stopping criterion must also be determined. The offspring population size M depends on the parent population size N and/or on the survivor selection method of the GA. For example, if the survivor selection method SS-S is taken into account, the offspring population size is fixed to $M = 1$, and with every generation a unique offspring is generated.

2.4 Multi-objective Evolutionary Algorithms

EAs specifically designed for solving MOPs are known as MOEAs. Since MOEAs are population-based meta-heuristics, which are able to deal with a set of solutions, their application to MOPs allows the Pareto front to be obtained in a single execution. Furthermore, MOEAs are suitable for difficult shapes of the Pareto front. Providing the exact Pareto front of a MOP is an arduous task. Nevertheless, obtaining a reasonably good approximation of the Pareto front in a reasonable time frame is possible by the application of a MOEA. The main objectives of a MOEA are the following [70]:

- Preserve a set of non-dominated vectors in the objective space and their corresponding non-dominated solutions in the decision space.
- Guide the search process towards the Pareto front in the objective space.
- Maintain the diversity of the Pareto front—objective space—and/or of Pareto optimal solutions—decision space.
- Provide a set of non-dominated solutions such that the best compromise solution can be selected depending on the features of the MOP being solved and on the preferences of the human decision maker.

A MOEA was first implemented in 1984 by Schaffer, who proposed the Vector Evaluated Genetic Algorithm (VEGA) [292] in order to solve problems in the field of machine learning. Since then, different MOEAs have been proposed [357]. The most important ones include Multi-Objective Optimisation Genetic Algorithm (MOGA) [123], Niche Pareto Genetic Algorithm (NPGA) [163], Non-Dominated Sorting Genetic Algorithm (NSGA) [317], Strength Pareto Evolutionary Algorithm (SPEA) [363], Pareto Archived Evolution Strategy (PAES) [187], Niche Pareto Genetic Algorithm 2 (NPGA2) [110], Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [91], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [362], Micro GA-MOEA (μGA^2) [331], and Indicator-Based Evolutionary Algorithm (IBEA) [361].

In general, the solutions to a MOP result, on the one hand, from an optimisation process carried out by a certain optimisation scheme, and on the other hand, from a decision process performed by the decision maker, in which the most suitable compromise solutions are selected. The latter process is also referred to as the *multi-criteria decision making process*. MOEAs are usually grouped by the definition of three different variants of the decision process [71]. Thus, the final solutions depend on the preferences of the decision maker, which can be specified before, during

or after the optimisation process. Consequently, MOEAs can be classified in the following groups [70]:

- *A priori methods.* In this case, the decision process is performed before the optimisation process. Lexicographic ordering, linear aggregating functions, and non-linear aggregating functions can be grouped into this category.
- *Progressive or interactive methods.* These types of techniques are based on carrying out the decision process at the same time as the optimisation process. This group includes progressive techniques and interactive computational steering.
- *A posteriori methods.* A posteriori approaches perform the decision process once the optimisation process finalises. The most important techniques belonging to this group are independent sampling, criterion selection, aggregation selection, ϵ -constraint methods, and Pareto sampling.

The main difference between single-objective EAs and MOEAs lies in the fitness assignment strategy. The most important fitness assignment methods for MOEAs can be classified as *scalar approaches*, *criterion-based approaches*, *indicator-based approaches*, and *dominance-based approaches* [326]. The next section provides a description of the different MOEAs applied in this research work to solve MOPs. Particularly, the well-known NSGA-II and SPEA2, which can be classified as dominance-based approaches, are considered herein.

2.4.1 Dominance-based Multi-objective Evolutionary Algorithms

Dominance-based MOEAs make use of the Pareto dominance concepts when assigning fitness. This idea was first proposed by Goldberg in 1989 [141]. *Ranking approaches*, which are based on the concepts of Pareto dominance and Pareto optimality, allow an order among different solutions to be established. The most popular dominance-based ranking methods are as follows [70]:

- *Dominance rank.* The rank associated with a particular solution is based on the number of solutions that dominate the solution being considered. MOGA and NPGA are examples of MOEAs that incorporate this dominance-based ranking method.
- *Dominance count.* In this case, the rank associated with a specific solution is based on the number of solutions dominated by the solution being considered.

For instance, the SPEA2 makes use of this type of dominance-based ranking approach.

- *Dominance depth.* This type of dominance-based ranking approach decomposes the population of individuals into different fronts. The non-dominated solutions in the population are assigned to the first front. Afterwards, the non-dominated solutions in the population without considering the solutions belonging to the first front are assigned to the second front, and so on. Thus, the rank associated with a solution is given by the depth of its front. An example of a MOEA which incorporates this kind of dominance-based ranking method is the NSGA-II.

Since the fitness assigned by the above schemes consists of a scalar value, dominance-based MOEAs follow the generic pseudocode shown in Algorithm 1. Nevertheless, instead of directly using the value of the objective function as the measure of fitness, such as in a single-objective EAs, one of the aforementioned dominance-based ranking methods is incorporated into the MOEA as the fitness assignment strategy. Hence, the different components exposed in previous sections, like the crossover, mutation, and selection operators, can be applied together with these MOEAs.

Non-dominated Sorting Genetic Algorithm II

The NSGA-II [91] is one of the most widely used MOEAs. One of its most important features is that it uses a fast non-dominated sorting approach with reduced computational complexity. In addition, it applies a selection operator which combines previous populations with newly generated ones to ensure elitism in the approach. The fast non-dominated approach, as well as the selection operator, require defining a partial order for the individuals. The *crowded comparison operator* (\geq_n) is used to establish such an order. This operator assigns two different attributes to every individual i in the parent population: the *non-domination rank* ($rank_i$) and the *local crowding distance* ($distance_i$).

The non-domination rank makes use of the Pareto dominance concept. The procedure to calculate it is as follows. First, the set of non-dominated individuals in the parent population is assigned to the first rank. Then the process is repeated considering only the individuals that do not have a rank assigned. The rank assigned in each step is increased by one. The process ends when every individual in the parent population has its corresponding rank established.

The local crowding distance is used to estimate the density of solutions surrounding

Algorithm 3 Pseudocode of the Non-Dominated Sorting Genetic Algorithm II

-
- 1: **Initialisation.** Randomly generate the initial parent population P_0 with N individuals. Assign $t = 0$.
 - 2: **Evaluation.** Evaluate all the individuals in the initial parent population by calculating the objective functions.
 - 3: **while** (not stopping criterion) **do**
 - 4: **Fitness assignment.** Calculate the fitness values of individuals in P_t . Use the non-domination rank in the first generation, and the crowded comparison operator in remaining generations.
 - 5: **Parent selection.** Perform deterministic binary tournament selection with replacement on P_t in order to fill the mating pool with N parents.
 - 6: **Variation.** Apply the crossover and mutation operators with probabilities p_c and p_m , respectively, to the individuals of the mating pool in order to create the offspring population CP with $M = N$ new individuals.
 - 7: **Evaluation.** Evaluate every offspring in CP by computing the objective functions.
 - 8: **Survivor selection.** Select the N fittest individuals from among N parents and M offspring by using the crowded comparison operator so as to constitute P_{t+1} .
 - 9: $t = t + 1$
 - 10: **end while**
-

a particular individual. First, the size of the largest cuboid enclosing the individual i without including any other individual that belongs to its rank is calculated. Then, the crowding distance is given by the mean side-length of the cuboid. It is important to note that the local crowding distance of the boundary individuals in every rank is assigned an infinite value. Finally, the partial order given by the crowded comparison operator \geq_n is as follows:

$$i \geq_n j \text{ if } \begin{cases} (rank_i < rank_j) \\ or \\ ((rank_i = rank_j) \text{ and } (distance_i > distance_j)) \end{cases} \quad (2.9)$$

The pseudocode of the NSGA-II is shown in Algorithm 3. During the initialisation stage—line 1—the initial parent population is filled with N randomly generated individuals, which are evaluated by calculating the objective functions—line 2. Then, as long as the stopping criterion of the NSGA-II is not satisfied—line 3—several steps are carried out in every generation. Firstly, the fitness is computed and assigned to every individual in the parent population—line 4. In the first generation, the non-domination rank is used, while the crowded comparison operator is applied in remaining generations. Secondly, N parents are selected from the parent population so as to fill the mating pool—line 5—by performing deterministic binary tournament

ment selection with replacement. In order to select parents, the fitness assigned to every individual in the previous step is considered. Then, a crossover operator and a mutation operator are applied to the mating pool—line 6—with the aim of building the offspring population, which consists of $M = N$ individuals. Crossover and mutation operators are applied with probabilities p_c and p_m , respectively. Afterwards, the offspring population—line 7—is evaluated by calculating the objective functions. Lastly, the fittest N individuals from among N parents and M offspring are selected—line 8—to create the parent population for the next generation. The crowded comparison operator is applied to select the fittest survivors.

The parent population size N , the internal representation of individuals, the crossover and mutation operators, as well as their rates of application, p_c and p_m , depend on the MOP in question. Furthermore, the stopping criterion must also be specified.

Strength Pareto Evolutionary Algorithm 2

The SPEA2 [362] is another well-known MOEA. One of its main features is that it makes use of an external archive to store the fittest individuals found. Furthermore, it establishes an order among the individuals using a fine-grained fitness assignment strategy. The fitness value of each individual—which must be minimised—is calculated as the sum of its raw fitness plus a density estimate. In order to calculate the raw fitness, the strength $strength_i$ of each individual i is calculated as the number of individuals that it dominates considering both the parent population P_t and the archive \overline{P}_t :

$$strength_i = |\{j | j \in P_t \cup \overline{P}_t \wedge i \preceq j\}| \quad (2.10)$$

Therefore, the raw fitness raw_i of each individual i is calculated as follows:

$$raw_i = \sum_{j \in P_t \cup \overline{P}_t, j \preceq i} strength_j \quad (2.11)$$

It is important to note that a minimisation problem arises when defining both equations above. In addition, note that the raw fitness of an individual is determined by the strengths of its dominators in both the parent population and the archive. Consequently, the lower the raw fitness the fitter the individual. A high raw fitness

Algorithm 4 Pseudocode of the Strength Pareto Evolutionary Algorithm 2

-
- 1: **Initialisation.** Generate the initial parent population P_0 with N individuals, and create the empty archive \bar{P}_0 . Assign $t = 0$.
 - 2: **while** (not stopping criterion) **do**
 - 3: **Evaluation.** Evaluate all individuals in the parent population by calculating the objective functions.
 - 4: **Fitness assignment.** Calculate the fitness values of individuals in P_t and \bar{P}_t . For each individual i , calculate the raw fitness raw_i and the density estimate $density_i$.
 - 5: **Environmental Selection.** Copy non-dominated individuals which belong to P_t and \bar{P}_t to \bar{P}_{t+1} . If $|\bar{P}_{t+1}| > \bar{N}$ reduce \bar{P}_{t+1} by means of the truncation operator. Otherwise, if $|\bar{P}_{t+1}| < \bar{N}$, fill \bar{P}_{t+1} with dominated individuals belonging to P_t and \bar{P}_t , considering their fitness.
 - 6: **Parent selection.** Perform deterministic binary tournament selection with replacement on \bar{P}_{t+1} to fill the mating pool with N parents.
 - 7: **Variation.** Apply crossover and mutation operators, with probabilities p_c and p_m , to the mating pool so as to obtain $M = N$ offspring.
 - 8: **Survivor selection.** Set P_{t+1} to the offspring population.
 - 9: $t = t + 1$
 - 10: **end while**
-

implies that the corresponding individual is dominated by a large number of individuals, which in turn also dominate many individuals. In contrast, a raw fitness equal to zero implies that the individual is not dominated by any other individual.

The raw fitness is not appropriate when most individuals do not dominate one other. As a result, density information is incorporated to discriminate among individuals with identical raw fitness values. The density estimator is an adaptation of the k -th nearest neighbour method [305]. For each individual i , the distances—considering the objective space—to each individual j from the parent population and the archive are calculated and stored in a list. Then, this list is sorted by increasing order, and the k -th item gives the desired distance, which is designated σ_i^k . Finally, $k = \sqrt{N + \bar{N}}$ is usually applied, where N is the parent population size and \bar{N} the archive size. The density estimate for the individual i — $density_i$ —is therefore given by:

$$density_i = \frac{1}{\sigma_i^k + 2} \quad (2.12)$$

Algorithm 4 shows the pseudocode for the SPEA2. During the initialisation stage—line 1—the initial parent population is filled with N randomly generated individuals, and an empty archive is also created. Then, for as long as the stopping criterion of

the SPEA2 is not satisfied—line 2—several steps are repeated for every generation of the algorithm. Firstly, the objective functions are computed to evaluate the individuals in the parent population—line 3. Secondly, the fitness assignment is carried out—line 4. To do so, the raw fitness and the density estimate are calculated for every individual in the current parent population and the archive. Then, the non-dominated individuals in the parent population and the archive are used to build an updated archive taking into account the fitness calculated in the previous step. If the number of non-dominated individuals in this updated archive is greater than the archive size, a truncation operator is applied. The truncation operator is based on the k -th nearest neighbour method. If, on the other hand, the number of non-dominated individuals in the updated archive is lower than the archive size, dominated individuals from the parent population and the old archive are used to fill the updated archive. Moreover, the updated archive becomes the current archive. The above procedure is known as the *environmental selection*—line 5. Then, deterministic binary tournament selection with replacement is performed on the archive—line 6—so as to create the mating pool with N parents. Afterwards, the crossover and mutation operators are applied to the mating pool with probabilities p_c and p_m in order to yield $M = N$ offspring—line 7. Finally, the offspring population—line 8—is used as the parent population for the next generation. We should note that when the stopping criterion of the SPEA2 is satisfied, the non-dominated individuals in the current archive are returned as the best found solutions.

The parent population size N , the archive size \overline{N} , the internal representation of individuals, the crossover and mutation operators, as well as their rates of application, p_c and p_m , depend on the MOP in question. Additionally, the stopping criterion must also be specified.

2.5 Memetic Algorithms

A MA consists of the combination of a population-based approach with a Local Search (LS) procedure [253]. The term “memetic”, which comes from the word “meme”, was first introduced by Richard Dawkins [85]. A meme is a “unit of imitation” in cultural transmission. Consequently, a MA tries to mimic cultural evolution rather than biological evolution. The main difference is that memes are typically adapted by the individual who transmits them, while genes are transmitted intact, as is the case with EAs. MAs are of great value because they execute several orders of magnitude faster than traditional EAs for certain problem domains [131, 274]. Additionally, these types of algorithms have been successfully applied in both

the single-objective [177, 266] and the multi-objective [127, 308, 343] fields.

In the literature, the local search procedure is also known as the *individual learning strategy* [200], which can be based on sophisticated approaches, such as TS [345] or SA [269], or on other, “simpler” techniques, like hill climbing methods [57] or other search procedures. The individual learning strategy can be classified into two main groups as per the taxonomy proposed by Whitley [349]. On the one hand, in *Lamarckian learning* the locally improved individual is placed back into the population to compete for reproduction. Hence, the result of the improvements made during the learning stage is reflected in the individual’s genotype. On the other hand, in *Baldwinian learning* only the objective functions of the individuals are updated and the locally improved individual is not placed back into the population. As a result, MAs are also referred to in the literature as *Lamarckian EAs* or *Baldwinian EAs*. It is worth mentioning that both types of individual learning procedures have been successfully applied to different optimisation problems [208, 356]. In this thesis, Lamarckian learning is always applied in the form of stochastic hill climbing approaches, which were specifically designed for the optimisation problem at hand.

Apart from the type of individual learning procedure they incorporate, MAs can be classified according to other characteristics. For instance, MAs can be grouped by different generations [265]. *First generation MAs* comprise the traditional approaches, which combine a population-based scheme with an individual learning strategy. *Second generation MAs* contain some types of hyper-heuristics, among other schemes. In this case, the aim of the hyper-heuristic is to constantly select the most promising individual learning strategy or meme from among a set of possible candidates during the optimisation process. In order to make its decisions, the hyper-heuristic considers historical information on the behaviour of the different candidate individual learning strategies. Finally, *third generation MAs* include approaches that are able to dynamically produce the pool of memes during the optimisation process, instead of previously specifying it as in the case of second generation MAs. It is important to note that first generation MAs are applied in this dissertation. Hyper-heuristics are also applied herein, but they cannot be classified as second generation MAs, since they do not select from a set of candidate individual learning procedures, instead choosing from a set of different configurations of certain meta-heuristics.

Algorithm 5 shows a generic pseudocode for a first-generation MA that combines an EA with an individual learning process. During the initialisation stage—line 1—the initial parent population is created. Then, the parent population is evaluated—line 2—through the computation of the objective function so as to assign a fitness value

Algorithm 5 Generic pseudocode for a memetic algorithm

- 1: **Initialisation.** Generate the initial parent population.
 - 2: **Evaluation.** Evaluate all individuals in the initial parent population by applying the objective function in order to assign a fitness value to every individual.
 - 3: **while** (not stopping criterion) **do**
 - 4: **Parent selection.** Select the individuals from the parent population to build the mating pool.
 - 5: **Variation.** Apply the variation operators to the mating pool in order to generate the offspring population.
 - 6: **Learning process.** Carry out the individual learning process in the offspring population with probability p_l .
 - 7: **Evaluation.** Evaluate the generated offspring via the objective function so as to assign a fitness value to every offspring.
 - 8: **Survivor selection.** Select individuals from among the parents and offspring as the new parent population for the next generation.
 - 9: **end while**
-

to every individual. Afterwards, in every generation of the MA, a set of steps is carried out until a stopping criterion is satisfied—line 3. Firstly, individuals are selected from the parent population—line 4—to create the mating pool. Secondly, the variation operators are applied to the mating pool—line 5—so as to produce the offspring population. The main difference between a MA with respect to an EA—Algorithm 1—is the addition of a step to carry out the learning process—line 6. Usually, this step makes use of information that depends on the optimisation problem at hand, and therefore it is specifically designed to deal with such a problem. However, general approaches are also considered as individual learning strategies. Note that the learning process is applied to the offspring population with probability p_l . Once the individual learning strategy is applied, the offspring are evaluated—line 7—by calculating the objective function in order to assign them a fitness value. Lastly, the survivors are selected—line 8—from among the parents and offspring to create the parent population for the next generation of the MA.

In [168], the effect of the probability p_l was studied considering a set of benchmark problems. This work concluded that the performance of a MA might be significantly improved by dynamically adjusting the probability p_l . However, there are other references in which the individual learning strategy is successfully applied in every generation of the MA [131]. In this dissertation, the individual learning procedure is always applied in every generation of the MAs in question. The main reason for selecting $p_l = 1$ is that the offspring obtained after the variation operators are applied might not attain a minimum level of quality. Moreover, the individual

learning procedures applied herein were designed bearing in mind their efficiency, and therefore they are able to improve the quality of the individuals in a reasonable time frame.

We should note that these individual learning procedures are combined with the NSGA-II and the SPEA2—Section 2.4.1—so as to create the multi-objective MAs applied in this research work. Furthermore, other single-objective MAs are also taken into account, but they were specifically designed to deal with certain problems. As a result, these MAs, as well as the individual learning strategies, will be introduced in the chapters devoted to optimisation problems.

2.6 Parallel Evolutionary Algorithms

As was stated in previous sections, meta-heuristics, and EAs in particular, yield high-quality, if not optimal, solutions, in a reasonable time frame. However, optimisation problems and real-world applications are becoming increasingly complex and a vast amount of computational resources and time are required for existing optimisation schemes, even EAs, to find a solution. Additionally, the emergence of multi-core architectures and faster communication networks has expanded the interest in parallel and distributed computing. As a result, researchers have focused on increasing the effectiveness and efficiency of EAs by parallelising them.

The main objective of a parallel EA is to explore a broader area of the search space in order to find better solutions in less, or at least the same time as that invested by a sequential version of the EA, i.e. to improve the quality of the solutions, as well as to speed up the search process. Furthermore, increasing the robustness of an EA, in terms of its ability to successfully solve different optimisation problems and/or instances, or in terms of its sensitivity to modifications of its parameters, is another aim of this parallelisation. Finally, another goal of parallel EAs is to solve large-scale optimisation problems for which a sequential EA is not able to provide solutions in a reasonable time.

Several classifications and surveys have been proposed in the literature for parallel EAs [59, 70, 82, 173, 189, 326]. Considering the classification proposed by Talbi [326], the parallelisation of an EA can be performed at three different levels:

- *Solution level.* The parallelisation is carried out on a particular component of the EA, such as a variation operator or the evaluation of the objective functions. The parallelisation at this level speeds up the search process. However,

the behaviour of the EA is not modified, and therefore, the quality of the solutions is not improved.

- *Iteration level.* At this level, every generation of an EA is parallelised, thus speeding up the search process. As in the case of parallel implementations at the solution level, the behaviour of the EA is not altered.
- *Algorithmic level.* In this case, a set of different EAs are executed in parallel. If the different EAs are independent, the quality of the solutions will be the same as in the case of separately executing the sequential versions of the EAs. In contrast, if some type of collaboration scheme is enabled among different EAs, an improvement in the quality of the solutions might be produced.

Parallel EAs are normally useful for optimisation problems in which the cost of evaluating the objective functions is very expensive from the standpoint of the computational resources invested [70]. For this reason, it is desirable to speed up the search process by parallelising the computation of the objective functions. In keeping with the above classification, this parallelisation is performed at the solution level. Different techniques have been developed to address this issue. One of the most widely applied methods is based on decomposing the objective function and distributing the different components among several cores. In this case, every core is responsible for executing one of the components so as to evaluate one of the objective functions of a single individual. Therefore, the above procedure has to be repeated with every objective function—if a MOP is considered—for every individual in the whole population. Another frequently used approach involves decomposing the population of individuals and distributing the sub-populations among the different cores. Thus, every core is responsible for computing the objective functions of the individuals in its sub-population. A third possibility, which is suitable only for MOPs, consists of assigning each of the objective functions to a particular core. Hence, the evaluation of an individual is also carried out by several cores, with every core being responsible for evaluating one of the objective functions. The above process has to be repeated for every individual in the population. The objective function is not the unique component of an EA that can be parallelised at the solution level. For instance, variation operators might also be parallelised. However, if the time invested by a variation operator is compared to the time invested by the evaluation of computationally expensive objective functions, the parallelisation of the variation operators is not worth the effort. Something similar happens with parent and survivor selection schemes.

Considering the parallelisation at the iteration level and recalling the generic pseudocode of an EA—Algorithm 1—we can see that the parent selection, the variation

stage, the evaluation of individuals, and the survivor selection must be executed sequentially. Each step requires the completion of the preceding one before it can start. That is why it is not possible to implement an efficient parallel EA in which each of the aforementioned steps is executed by one core. Since there are dependencies among different tasks, cores increase their idle time, and consequently the performance of the parallel EA diminishes. Nevertheless, another parallel implementation at the iteration level considers the decomposition of the population and the distribution of sub-populations among several cores. Thus, each core performs different operations on the corresponding sub-population.

Parallel implementations at the solution and iteration levels do not modify the behaviour of the EA; they only speed up the search procedure. However, it is also desirable to improve the quality of the solutions, and therefore the only possibility lies in parallelising at the algorithmic level, where different EAs are simultaneously executed by different cores, and where the results obtained by the EAs are combined using some type of collaboration scheme so as to explore the search space more efficiently [338]. Parallelisation at the algorithmic level is considered in this dissertation.

Another famous taxonomy for parallel EAs was the one proposed by Coello [70]. Four major parallel computational paradigms are identified:

- *Master-worker paradigm.* In this paradigm, the evaluations of the objective functions are distributed among a set of worker cores, while a master core is responsible for executing the variation and selection operators, among other tasks. The evaluations can be distributed using one of the methods mentioned in a previous paragraph. Models that follow this paradigm speed up the search process, but they do not improve the quality of the solutions. Finally, it is important to note that the parallelism is implemented at the solution and iteration levels in the master-worker paradigm.
- *Island paradigm.* The approaches belonging to this paradigm are also called *distributed*, *multiple-population*, *multiple-deme* or *coarse-grained models*. In this case, the population is decomposed into a number of independent and separate sub-populations or *demes*, i.e. there exist small, separate, and simultaneously executing EAs—one per core or island. Thus, each island evolves in isolation for the majority of the execution, but occasionally, some individuals can be migrated between neighbouring islands based on some selection or fitness criteria. In this type of paradigm, the parallelism is located at the algorithmic level. In addition, since some implementations make use of collaborative schemes, the search space can be explored more efficiently and

consequently better solutions might be found.

- *Diffusion paradigm.* This paradigm is also known as the *fine-grained or cellular paradigm*. Models belonging to this group deal with one conceptual population in which every core holds only between one and a few individuals. In diffusion models a topological structure must be established between the different cores, which defines a neighbourhood. Variation and selection operators are applied exclusively to this neighbourhood. As in the case of the master-worker paradigm, the parallelism is located at the solution and iteration levels.
- *Hierarchical or hybrid paradigm.* This type of paradigm can be seen as a combination of the three above paradigms. For instance, Cantú-Paz proposed three different island-based hybrid models [59]: each island contains a diffusion parallel EA, each island comprises a master-worker parallel EA, and each island consists of an island parallel EA.

2.6.1 Island-based Model

In this research work, parallel EAs are implemented starting from the island-based model. The island-based model, when compared to the other parallel approaches, offers two main benefits: it maps easily onto the parallel architectures—thanks to its distributed and coarse-grained structure—and it extends the search area—due to multiplicity of islands—thus reducing the problem of local optima stagnation. Coello provided a classification for island-based models that considers MOEAs [70]. This classification is also suitable for single-objective EAs. Four basic island-based schemes exist according to this classification: all islands execute identical EAs with the same parameterisation (*homogeneous*), all islands execute different EAs and/or parameterisations (*heterogeneous*), each island evaluates different objective function sub-sets, and each island represents a different region of the genotype or phenotype domains. The first two variants are usually known as *standard island-based models*. In both the population located on every island represents solutions to the same problem. In the third variant, which is only suitable for MOPs, every island focuses on some of the objective functions considered. The last variant isolates each core to solve specific, non-overlapping regions of the genotype/phenotype domain space. In these four types of island-based models, populations are evolved in isolation on every island. However, the islands are occasionally able to exchange individuals. This exchange of individuals is called *migration*.

The standard island-based models, in which every island is associated with the complete search space, seem to be more efficient and easier to implement [338], although

they do not guarantee optimality. Generally, the benefit with the implementation of standard island-based models lies in the fact that in these models every island executes a sequential version of an EA.

In island-based models the configuration of the migration stage is particularly important, since it allows for cooperation among different islands. Therefore, the *migration policy* must be carefully designed in order to guarantee the best performance of the parallel EA. Some of the most common migration stages were analysed in [60]. In order to configure the migration stage, it is necessary to establish the *migration topology*—where to migrate the individuals—and the *migration rate*—how many individuals to migrate and how often. In synchronous schemes migrations are periodically performed every fixed number of generations. In asynchronous models a probability of migration has to be established. Then, when a generation finishes, a migration is performed based on this probability. In this dissertation, asynchronous migrations are taken into consideration. Furthermore, individuals that are going to be migrated and those that are going to be replaced must be selected. This selection is performed by using the *migration scheme* and the *replacement scheme*, respectively.

With regard to the migration topologies, the following are considered in this research work:

- *All to All Connected Topology (ALL)*. In this topology every island connects with and sends its individuals to every other island.
- *Unidirectional Ring Topology (RING)*. In this case, every island connects to exactly two other islands, constituting a logical ring. If there exist n_p islands, labelled from 0 to $n_p - 1$, every island γ sends its individuals to the island $(\gamma + 1) \bmod n_p$, and receives individuals from the island $(\gamma + n_p - 1) \bmod n_p$.

Several migration and replacement schemes have been proposed for both the single-objective [60] and the multi-objective [338] fields. Particularly, the following approaches are considered in this thesis:

- *Elitist Migration Scheme (ELI-M)*. An individual from an island's population is migrated when its fitness is better than the fitness of any member of its previous generation. Note that the migration rate is implicitly defined for this migration scheme.
- *Random Migration Scheme (RND-M)*. The individual that is migrated is randomly selected.
- *Elitist Ranking Replacement Scheme (ELI-R)*. This replacement scheme ranks

all Pareto fronts and replaces an individual randomly selected from the worst ranked front with the immigrant. This scheme was specifically designed for the multi-objective field and it provides a high selection pressure.

- *Random Replacement Scheme (RND-R)*. The individual that is replaced is randomly selected.
- *Hamming-based Replacement Scheme (HAM-R)*. This replacement scheme first checks whether or not the immigrant has a better fitness than all the individuals on the destination island. If so, the immigrant replaces the individual with the lowest Hamming distance to it, considering the decision space. If not, the immigrant is discarded. This replacement scheme was proposed for binary representations of the individuals, and its aim is to maintain the diversity of the population, since it is based on selecting individuals that are quite similar to the immigrants.

Background in Multi-objective Evolutionary Algorithms for Single-objective Optimisation

As was stated in the previous chapter, one of the goals of most MOEAs is to maintain a proper diversity of individuals in both the objective and decision spaces so as to minimise the premature convergence problem. Due to this implicit feature that the majority of MOEAs share, their application to solving single-objective optimisation problems might be helpful [2]. Three different types of mechanisms are seen to exist for solving single-objective optimisation problems by means of MOEAs [297].

- Methods that transform a constrained single-objective problem into an unconstrained MOP [246]. On the one hand, the constrained single-objective problem can be transformed into an unconstrained bi-objective problem where the second objective is a measure of the constraint violations. On the other hand, the constrained single-objective problem can also be transformed into an unconstrained MOP with N objectives. In this last case, the original objective and the constraints are treated as separate objective functions. Hence, $N - 1$ is the number of constraints. In addition to the above classification, another taxonomy [297] considers an additional dimension that is used to group this type of mechanism into *feasible-compliant methods*, which prefer a feasible solution over an unfeasible solution, and *non-feasible-compliant methods*, which treat feasible and unfeasible solutions equally.
- Methods that consider diversity in the definition of auxiliary objective functions [188].
- Methods known as *multi-objectivisation*, which transform a single-objective problem into a multi-objective one by altering its fitness landscape [45].

This chapter is devoted to providing a background for the second and third types of methods, i.e. diversity-based MOEAs—Section 3.1—and multi-objectivisation—Section 3.2—as techniques for solving single-objective problems, since these mechanisms are applied throughout this thesis.

3.1 Diversity-based Multi-objective Evolutionary Algorithms

The application of MOEAs to guarantee the appropriate diversity when tackling single-objective optimisation problems is a promising idea. Multi-objective schemes try to optimise several objective functions simultaneously; consequently, the use of diversity measures to define auxiliary objective functions might provide a suitable balance between the exploration and exploitation abilities of a MOEA. Such auxiliary objective functions are also called *diversity-based objectives* in the literature [297]. It is important to remark that what is required to define the diversity-based objective is not an indicator that measures the diversity of the whole population, but rather one that measures the amount of diversity introduced into the population by an individual itself. Furthermore, since diversity-based objectives are used together with the original objective of the single-objective problem in question, one of the main advantages of using this kind of approach is that the Pareto front always contains an objective vector whose value for the original objective is optimal.

In keeping with the taxonomy provided by Segura et al. [297], diversity-based objectives can be grouped depending on the diversity measures used to define them:

- *Encoding-independent measures*. These types of measures can be applied regardless of the chromosome used to represent individuals.
- *Genotypic and phenotypic measures*. These kinds of measures take into account the values of the genes either in the genotypic or the phenotypic space.
- *Behavioural measures*. This group of measures considers the behaviour of the individuals.

We should note that diversity-based objectives can be used not only to deal with single-objective problems, but also with MOPs. For instance, a bi-objective problem might be solved by the application of a MOEA that simultaneously optimises three objective functions—two objective functions belonging to the original definition of the bi-objective problem, and an additional diversity-based objective [87].

Additionally, several diversity-based objectives could be defined and simultaneously optimised together with the original objectives. Nevertheless, this last approach is rarely applied since the resulting MOP might have many objectives. In this dissertation, encoding-independent and genotypic measures are taken into consideration when defining diversity-based objective functions. They are incorporated into some of the multi-objective optimisers described in Chapter 2. Hence, since diversity-based MOEAs are used to solve single-objective optimisation problems, two different objective functions are simultaneously optimised herein: the original objective function corresponding to the single-objective optimisation problem at hand, and an additional diversity-based objective.

3.1.1 Encoding-independent Measures

The codification of the individuals is not considered in the design of this type of diversity measure. As a result, they are not direct measures of the diversity. Nevertheless, they can indirectly maintain the diversity of a population of individuals. Different encoding-independent measures have been proposed with the aim of defining diversity-based objective functions. In this thesis, the encoding-independent diversity-based objectives proposed by Abbass and Deb [2] are considered:

- *Random.* A random value is assigned as the diversity-based objective. Smaller random values may be assigned to some low-quality individuals, thus giving them a chance to survive. This diversity-based objective must be minimised.
- *Inversion.* In this particular case, the optimisation direction of the original objective function is inverted and used as the diversity-based objective. As a result, both objective functions will assign the same value to an objective vector. However, one of the objectives will be maximised, while the other objective will be minimised. The main feature of this diversity-based objective is that it significantly decreases the selection pressure. In fact, every member of the population is a non-dominated solution.
- *Time stamp.* This diversity-based objective is calculated as a time stamp for every individual. Every individual in the initial population is assigned a different time stamp represented by a counter which is increased every time a new individual is generated. Afterwards, starting with the second population, all newly generated individuals are assigned the same time stamp, which is set as the population size plus the generation index. As in the case of the first approach, this diversity-based objective must be minimised.

3.1.2 Genotypic and Phenotypic Measures

This type of diversity measure is designed by considering differences among individuals at the genotypic or phenotypic domains. The most frequently used genotypic and phenotypic diversity measures are based on the calculation of distance metrics [337]. To do this, the values of the genes have to be used in order to calculate the distance metrics, and different approaches can be taken into consideration—Hamming distance, Euclidean distance or edit distance, among others. Thus, these diversity measures are also known as direct diversity measures. A large number of diversity measures have been proposed for both the genotypic [76, 337] and the phenotypic [77, 337] spaces.

One of the first diversity-based objective functions to use a direct measure of diversity was based on a distance metric suitable for tree representations [87]. In this case, the diversity-based objective of an individual was calculated as its mean distance to the remaining individuals in the population. This diversity-based objective was incorporated into a multi-objective GP algorithm and optimised together with two other objective functions. Another example is a diversity-based objective specifically designed for working with a multi-objective version of a Vehicle Routing Problem (VRP) [130], in which the distance between two individuals is calculated as the number of shared edges. As in the previous approach, the mean distance to the remaining individuals in the population is used as the diversity-based objective. In [300], a bi-objective formulation of a problem aimed at optimising compliant mechanisms—flexible elastic structures—is addressed by the definition of a diversity-based objective. In this case, individuals are encoded using a binary string. The original objective belonging to the single-objective definition of the problem consists of minimising the weight of the structure. The diversity-based objective involves maximising the Hamming distance between the individual at hand and the reference design. The structure produced after the optimisation of the single-objective variant of the problem, i.e. after minimising the weight of the compliant mechanism, is considered as the reference design. Lastly, other diversity-based objectives have been specifically designed to deal with real-valued encodings of the individuals [45, 330]. The following diversity-based objectives belonging to this class are applied herein:

- *Average Distance to all Individuals (ADI)*. This diversity-based objective proposed by Toffolo and Benini [330] is calculated as the mean Euclidean distance in the genotypic space to the remaining individuals in the population. It has to be maximised.

- *Distance to Closest Neighbour (DCN)*. This diversity-based objective was proposed by Bui et al. [45] considering the ideas exposed by Toffolo and Benini [330]. The diversity-objective is calculated as the Euclidean distance in the genotypic space to the closest neighbour in the population. It has to be maximised.
- *Distance to Best Individual (DBI)*. This diversity-based objective was proposed together with the DCN approach [45]. In this case, it is calculated as the Euclidean distance in the genotypic space to the best individual in the population, with this best individual being determined by its original objective value. This diversity-based objective has to be maximised.

It is important to note that among the aforementioned diversity-based objectives, the ADI and the DCN approaches take more time to calculate, because they need to compute $O(N^2)$ distances, where N is the population size. However, since in real-world applications the time invested in evaluating an individual is usually high, the time required to calculate distances is insignificant in comparison to the time required to evaluate an individual.

3.1.3 Behavioural Measures

Behavioural measures are used in fields where calculating distance metrics in the genotypic or the phenotypic spaces is an arduous task. For instance, in the field of Evolutionary Robotics (ER), these types of measures are usually applied. In ER, EAs allow neural networks to be evolved. These neural networks operate as robot controllers. Since the computation of distances among neural networks in the genotypic or phenotypic spaces is quite difficult, Mouret and Doncieux have proposed the usage of *behavioural diversity* as an alternative to calculate distances [96, 256, 258], and therefore to define diversity-based objectives. Behavioural diversity defines diversity-based objectives which evaluate the difference in terms of the behaviour of an individual with regard to the current population. The behaviour of a robot results from its interaction with a given environment. In the particular case of controlling a mobile robot [256], the currently evolved neural networks—individuals—are evaluated by using them in order to simulate the behaviour of a mobile robot in the environment. Afterwards, differences or distances among individuals are computed by means of the status of the environment at the end of the simulations. For example, if the problem consists of moving several items in the environment, a possibility might be to calculate the differences among the positions of the different items in the environment at the end of the simulations. In this way, a distance metric among

individuals might be calculated, and consequently diversity-based objectives could be defined.

Behavioural novelty [202, 203] is another frequently used concept to define diversity-based objectives in ER [255, 257]. In behavioural novelty, distance metrics are similar to those applied in behavioural diversity. However, the novelty of an individual takes into consideration the set of all previously found behaviours—including the current population—and not only the current population, as in the case of behavioural diversity. To do so, an archive is used to store all the previous novel individuals and an individual is stored in the archive if its novelty is above some minimal threshold. As a result, different variants are seen to exist so as to compute distances among individuals. Distances can be calculated by using just the members of the archive, or by using both the members of the archive and the members of the current population.

Finally, another line of research involves calculating distances as the differences of the values coming from the sensors and the actions being sent to the effectors belonging to the robot [95], rather than using behavioural diversity and/or behavioural novelty.

3.2 Multi-objectivisation

Multi-objectivisation is a technique which also considers several objective functions in order to transform a single-objective optimisation problem into a MOP [188]. Nevertheless, the objective functions are not based on diversity measures, which require information about the remaining individuals in the population so as to calculate them. Quite the opposite, the additional objective functions are calculated by only considering the chromosome of the individual at hand. Hence, this is the main difference with the diversity-based objective functions introduced in the previous section.

Research has uncovered several ways in which multi-objectivisation leads to improved performance [213]. Firstly, multi-objectivisation might allow optimisation schemes to escape from local optima through avoiding confounding interactions within the search space. This process is called the *breakage of epistasis*. It implies that multi-objectivisation allows optimisation schemes to cover regions of the search space that would have been difficult to explore in light of the single-objective definition of the problem. Secondly, multi-objectivisation might increase the recognition of important fitness improvements. Changes in the decision space might produce a significant improvement in one of the newly considered objective functions. While

such an improvement might be easily detected by multi-objective approaches, this is not the case for single-objective optimisers.

In one of the first applications of multi-objectivisation [217]—this term was not yet used—a deceptive single-objective function that consisted of the sum of two different components was multi-objectivised by considering each of the two components as two independent objective functions. This work demonstrated the benefits of using Pareto-based selection to solve a single-objective problem. However, these ideas were abandoned during a decade approximately, until the term multi-objectivisation was introduced by Knowles et al. [188]. Multi-objectivisation can be performed using two approximations: *decomposition* and *aggregation*. As its name indicates, in multi-objectivisation by decomposition the original objective is decomposed into several components. Afterwards, every component is treated as an independent objective function, all of which are optimised such that the original optimum is a Pareto optimum in the multi-objective variant. In order to achieve this feature, the most widely applied method is to consider the original objective function as a sum of several components. In contrast, in multi-objectivisation by aggregation some additional objective functions are defined and optimised together with the original objective function. Since the original objective is kept, the Pareto front of the new formulation of the problem always contains an objective vector whose value for the original objective is optimal.

The first work carried out by Knowles et al. [188] focused on multi-objectivisation by decomposition. The advantages of this type of multi-objectivisation were demonstrated by addressing some small instances of the Travelling Salesman Problem (TSP) and a benchmark function. Multi-objectivisation by decomposition might remove some local optima from the original single-objective formulation of the problem [217, 188]. Furthermore, some plateaus might be incorporated in the fitness landscape by the usage of this kind of multi-objectivisation. Plateaus are useful for destroying deceptive regions, and offer the possibility of escaping from sub-optimal areas of the search space. A deeper analysis on multi-objectivisation by decomposition concluded that Pareto selection in a decomposed problem has the unique effect of creating plateaus containing incomparable solutions [150]. Nevertheless, the incorporation of plateaus into the search space might produce a beneficial or a counter-productive effect on the whole optimisation procedure.

Theoretical research has also been carried out that takes into consideration multi-objectivisation by aggregation of objective functions. The different benefits that multi-objectivisation by aggregation might yield include avoiding the stagnation of local optima, preserving diversity and identifying promising building blocks [176].

It is worth noting that these benefits can be also achieved via multi-objectivisation by decomposition. Additional objectives are also called *helper-objectives* in the literature [176]. Helper-objectives can be defined as novel objectives [144] or as decomposed components of the original objectives [176]. In addition, they can be generated by considering problem-dependent [347] or problem-independent information [144]. The main advantage of the latter approach is its generality. The use of helper-objectives together with Pareto-based selection could lead to two different effects [44]. Firstly, comparable solutions could become incomparable, since a region of the search space with a certain search direction could be transformed into a plateau. Secondly, incomparable solutions could become comparable, transforming a plateau into a region with a clear search direction. It is important to note that the removed/added search direction might be deceptive or it could guide the search procedure towards the global optimum; consequently, multi-objectivisation by aggregation of helper-objectives may have a positive or a negative effect. Throughout this dissertation, multi-objectivisation by aggregation of helper-objectives is carried out. These helper-objectives are components of the original objective function. Since they are based on information that depends on the problem being solved, their description will be given in those sections relevant to the optimisation problems considered.

Multi-objectivisation, and particularly helper-objectives, can be applied using different guidelines. The use of dynamic helper-objectives, where the helper-objective applied changes depending on the stage of the optimisation process, was proposed in [176]. In this particular case, helper-objectives were randomly chosen. Following this approach, it was concluded that the structure of the search space might be modified, thus facilitating the avoidance of being trapped in local optima. An example of applications in which helper-objectives are dynamically selected is the generation of performance test cases for programming challenge tasks [53, 54]. In the first work, the dynamic selection is adaptive and based on reinforcement learning, whereas in the second, the dynamic selection is performed randomly. The use of dynamic selection approaches yielded better results than the selection of helper-objectives by hand.

The simultaneous application of several helper-objectives was also studied in [176]. However, applying different helper-objectives simultaneously does not offer any advantages because the selection pressure is almost completely removed from the optimisation scheme. Studies on the order in which helper-objectives have to be applied have also been carried out not only for complex applications [214], but also for benchmark problems [215]. In both cases, the results were significantly altered by applying the appropriate sequence of helper-objectives. Multi-objectivisation has

also been applied to optimise scalarizing functions [166, 170]. Scalarizing functions are used to transform a MOP into a single-objective problem [249]. Since a single-objective variant of an optimisation problem is considered, multi-objectivisation techniques can be applied in these cases. Finally, the application of the principles of multi-objectivisation to solve MOPs has also been proposed. In these schemes, the selection pressure is generally decreased significantly due to the large number of objective functions. Nevertheless, there exist some MOPs which have been successfully solved by the application of multi-objectivisation approaches [167]. In this specific application, a bi-objective problem is converted into a problem with four objective functions. The two added objectives are linear combinations of the two original objective functions.

With regard to multi-objectivisation methods applied to real-world complex problems, it is important to note the reduction of bloat in GP algorithms [36], the prediction of protein structures [132, 133, 149, 151], the optimisation of VRPs [347], or an optimisation problem which considers the design of structures [144], among others. For instance, in the case of reducing the bloat in GP, the size of the trees is considered as an additional helper-objective. In the original definition of the problem which deals with the prediction of protein structures, an energy function must be minimised. Several studies have proposed different ways of decomposing such an energy function into different components. Additionally, other typical problems, such as the TSP [172, 176, 188] or the Job-Shop Scheduling Problem (JSP) [176, 214], have also been addressed by the use of multi-objectivisation by decomposition and/or by aggregation, as have graph problems, like the shortest path problem [293] or the minimum spanning tree problem [264].

State of the Art in Parameter Setting in Evolutionary Algorithms

This chapter provides some background on parameter setting in EAs. Parameter setting strategies are usually divided into parameter tuning and parameter control schemes. Sections 4.1 and 4.2 describe the main features of tuning and control schemes, and classify the most important proposals for both types of setting methods using well-known taxonomies given in the literature. Afterwards, the combination of fuzzy systems and EAs, including the application of FLCs as methods to adapt the numeric parameters of an EA, is discussed in Section 4.3. Lastly, the usage of hyper-heuristics as parameter control approaches is detailed in Section 4.4.

4.1 Parameter Tuning

Parameter tuning can be viewed as a particular case of algorithm design [107]. One of the most challenging tasks for EA designers is given by the fact that the values selected for the parameters greatly determine the performance of the algorithm. Therefore, the proper choice of parameter values for an EA is an optimisation problem in and of itself. In parameter tuning, parameter values are fixed before the run of a particular EA starts, and do not change while the EA is running. Two different types of parameter tuning exist. In *structural tuning* symbolic parameters are tuned, while in *parametric tuning* numeric parameters are tuned.

Figure 4.1 shows the control flow—left-hand side—and the information flow—right-hand side—through a three-level hierarchy for parameter tuning. The three different layers are the *application layer*, the *algorithm layer*, and the *design layer*. Additionally, two different parts are also considered: the problem solving part and

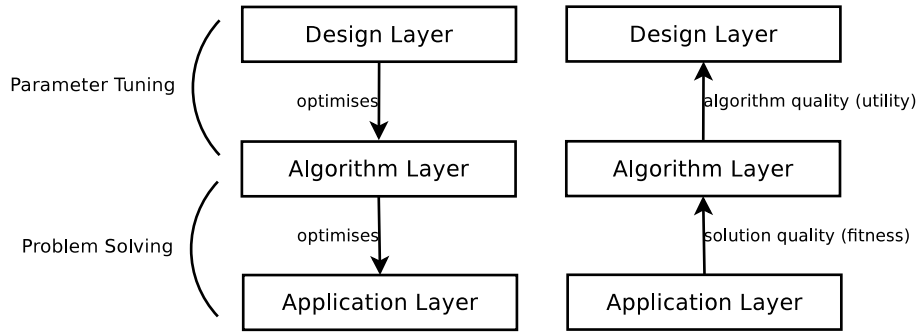


Figure 4.1: Three-layer hierarchy of parameter tuning

the parameter tuning part. On the one hand, the problem solving part consists of an optimisation problem involving the application layer, and an EA involving the algorithm layer whose task is to look for optimal solutions—with the best fitness—for the optimisation problem in question. On the other hand, the parameter tuning part consists of a tuning method involving the design layer that searches for optimal parameter values—with the best utility—for the EA located in the algorithm layer. The utility of a set of parameters allows the performance of an EA executed with said set of parameters to be measured. Hence, the problem of parameter tuning can be viewed as an optimisation problem in a search space of parameter sets given some utility function. The main difference between utility and fitness functions lies in the fact that the latter is closely related to the objective functions of the optimisation problem involving the application layer, whereas the former is related to the indicators used to measure the performance of the EA at the algorithm level.

Performance and robustness are the two main factors that determine the quality of an EA. With regard to performance, combinations of solution quality and algorithm speed metrics are the most commonly used. Solution quality is generally expressed by means of the fitness function incorporated into the EA in question. Algorithm speed is usually measured through the number of function evaluations or by the Central Processing Unit (CPU) time, for instance. Moreover, due to its stochastic nature, several runs must be performed in order to estimate the performance of an EA with sufficient statistical confidence. As a result, some of the most frequently applied metrics to measure the performance of an EA are the mean best fitness achieved at the end of the executions, the average number of evaluations or time invested to achieve a minimum fitness level, and the success rate, which indicates the percentage of successful runs. A run succeeds if a minimum fitness level is achieved within a maximum amount of evaluations or time. Other measures rely on using the median instead of the mean of the data. With respect to the robustness of an EA,

different interpretations of this term exist. However, robustness relates to the variance in the algorithm's performance over some dimension. Usually, the performance of an EA depends on its parameter values, on the problem instance being solved, or on the random seed used to provoke a stochastic behaviour. Consequently, robustness can be defined as the variance in the performance of an EA when changes are made to its parameters, when it is applied to different problems and/or instances, or when it is executed with different random seeds.

Several taxonomies have been introduced in order to group tuning methods [107]. One of the most recent classifications, which was proposed by Eiben and Smit [107], considers the different interpretations of robustness. Based on this distinction, the authors group tuning approaches in the following four classes:

- *Sampling methods.* These approaches reduce the search effort by reducing the number of parameter sets tested with respect to a full factorial design. The most commonly used sampling methods are *Latin-Square* [260] and *Taguchi Orthogonal Arrays* [324]. The output of these schemes needs to be analysed to predict which parameter values are the best and the most robust. As a result, sampling methods are not usually applied as independent tuning approaches. They are applied as initialisation methods or as a starting point for model-based methods. In general, these approaches yield low-quality parameter values due to a lack of search refinement. Another variant of sampling methods are *iterative sampling methods*. The difference with respect to the standard sampling methods is that the iterative variants are able to refine the area from which new points are sampled. Thus, iterative sampling approaches can be used as independent tuning techniques. One of the most famous iterative sampling methods is *CALIBRA* [4]. In order to determine the promising areas to be resampled a quality measure is considered, meaning that iterative sampling methods are suitable for improving the performance of an EA but not its robustness.
- *Model-based methods.* In the field of parameter tuning, *meta-models* or *surrogate models* allows building a model of the utility landscape. Firstly, different tests with different parameter sets are executed, for instance, by using a sampling method. Afterwards, the utility of each tested parameter set is calculated. Lastly, the utility information is used to generate the model via a regression method [83]. This model is applied to predict the utility of an untested parameter set. As in the case of sampling methods, low-quality parameter sets are obtained by model-based methods. *Iterative model-based methods* try to mitigate this problem through a more fine-grained exploration

of the parameter search space. Coy’s procedure [81] is one of the most basic iterative model-based methods. It consists of applying a standard model-based method followed by a local search procedure. Another widely used iterative model-based method is Sequential Parameter Optimisation (SPO) [27], where the model is continuously refined.

- *Screening methods.* These methods try to identify the best parameter set while employing a minimum number of tests. The selected parameter sets are re-tested and the whole process is repeated until no more testing is required. Hence, screening methods are either able to provide the best parameter set with less computational effort than sampling methods, or they are able to test a larger number of parameter sets by investing the same amount of computational resources. In the last case, the quality of the best parameter set found is usually higher and the information on the robustness to changes in parameter values is also more extensive. Screening methods are influenced by the field of system selection [142]. Among the most important system selection approaches are Interactive Analysis (IA) [294] and Ranking and Selection (R&S) [288]. In the field of parameter tuning, the term *racing* has been adopted to refer to screening methods. One of the most frequently applied racing schemes is *F-RACE* [33]. *Iterative screening methods* are also available. For instance *I/F-RACE* [23] is the iterative variant of F-RACE, which is based on combining a screening method with a fine-grained search.
- *Meta-EAs.* The notion of meta-EAs was introduced by Mercer and Sampson [243]. However, the meta-GA proposed by Grefenstette [143] was the first practical method, since previous proposals were computationally expensive. In a meta-EA every individual encodes a different parameter set corresponding to the EA being tuned, and individuals are evolved by the meta-EA. The (meta-)fitness value of an individual represents its utility, which is computed by executing the tuned EA with the parameter set encoded by said individual. The two main problems that arise when applying a meta-EA are the existing noise in the utility values, and the high cost of the evaluations needed to calculate them. As a result, some schemes have been proposed that generally make use of screening approaches to reduce the number of tests required. For instance, in [354] a racing method is combined with a meta-GA so as to tune symbolic and numeric parameters. Other approaches have been proposed in an effort to provide models of the utility landscape, as well as a deeper insight into the different types of robustness. One of these approaches is Relevance Estimation and Value Calibration (REVAC) [261]. It is a meta-EA whose population approximates the probability density function of the most promising

areas of the utility landscape. It is combined with racing methods to deal with the existing noise in the utility values [309]. Finally, some meta-EAs have been proposed to deal with multi-function tuning problems. They are termed multi-objective meta-EAs. In these schemes, each performance measure and fitness function is treated as a different objective. As a result, a multi-objective optimiser must be employed as the meta-EA. Hence, the Pareto front can be used to evaluate robustness in terms of changes in problem definition, as well as to measure the performance using several metrics. An example of multi-objective meta-EA is *M-FETA* [310].

4.2 Parameter Control

Parameter control schemes, unlike parameter tuning methods, allow the parameter values of a given algorithm to be modified during its execution. The ideas of parameter control were first incorporated in early work in EAs [84, 284]. For instance, Rechenberg proposed its “1/5 *success rule*” [284] to adapt the mutation parameters of ES during their run. Nevertheless, recent research has seen a marked increase in proposals for methods to achieve parameter control in EAs. Control methods have been successfully applied to a wide range of EAs such as ES [191], GAs [117], or DE [280].

In order to classify parameter control approaches, several taxonomies have been proposed [9, 105, 106, 315]. Considering the one given by Eiben et al. [106], four criteria are used. Firstly, parameter control approaches can be classified depending on the parameter or component that they change. Specific methods exist to control the representation of individuals, the variation operators, the evaluation function, or the selection operators, among other parameters and components. A second classification can be performed according to the manner in which parameter values are changed. Hence, control methods are classified in three main groups—Figure 4.2:

- *Deterministic parameter control*. Parameter values are altered by a deterministic rule without using any feedback from the search procedure.
- *Adaptive parameter control*. Parameter values are updated by a mechanism which uses some feedback from the search process. Such a mechanism is externally supplied.
- *Self-Adaptive parameter control*. Parameters are encoded into the chromosome and their values are modified by the variation operators. The main difference

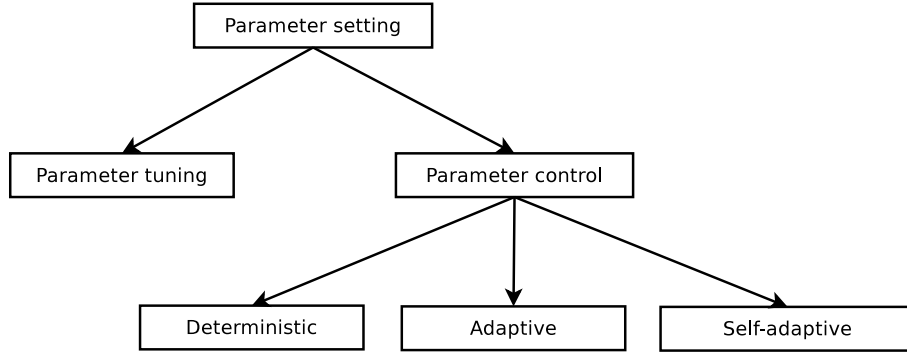


Figure 4.2: Taxonomy of parameter setting in evolutionary algorithms

between adaptation and self-adaptation lies in the fact that the mechanism that updates the parameter values in the latter approach is implicit, i.e. is given by the selection and variation operators of the EA itself. Good reviews on this type of parameter control methods are provided in [17, 245].

Some authors have introduced other terminologies to classify control approaches based on this second criterion. For instance, Angeline [9] proposed classifying control schemes into *absolute* and *empirical* methods, which refer to the *uncoupled* and *tightly-coupled* approaches given by Spears [315]. The absolute/uncoupled group refers to deterministic and adaptive methods, whereas the empirical/tightly-coupled schemes correspond to self-adaptive techniques.

Another additional criterion for classifying parameter control approaches is the information or evidence that is used to apply the change to the parameter [311, 312]. For instance, control methods might be based on information regarding the performance of the variation operators, or on a measure of the population diversity. From this point of view, there exist two main categories:

- *Absolute evidence.* The value of a parameter is altered by some rule that is fired when a predefined event happens. The main difference with respect to deterministic parameter control is that feedback from the search process is taken into account. An example might be changing the crossover or the mutation rate according to a fuzzy system whose inputs are given by different metrics calculated for the population [201].
- *Relative evidence.* Parameter values are compared considering the fitness of the offspring they produce, thus rewarding the best values. The direction and/or magnitude of the change is not specified in a deterministic way, but relative to the performance of other values. For instance, consider an EA

applying several crossover operators during a generation and assume that the sum of the crossover rates is 1. These rates are then modified taking into consideration the performance of the different crossover operators in terms of the offspring they generate. The update of these rates might be adaptive or self-adaptive.

Finally, a fourth classification can be carried out considering the scope or level that is affected by the change [9, 312, 315]. Hence, a change can affect a gene, an individual, the whole population, other components, such as the selection operators, or even the evaluation function.

Within each of these four classifications, a wide variety of approaches can be found in the literature. For instance, the *delta coding algorithm* introduced in [237] performs an adaptive adjustment of the individuals' representation based on absolute evidence. Another example is given in [192], where the *Boltzmann selection mechanism* [19] is used to dynamically alter the selection pressure in the individual learning stage of a MA. It is worth noting, however, that the majority of work on parameter control in EAs is focused on the variation operators—mutation and crossover—the population size, or combinations of all three [10, 13, 18, 84, 106, 156].

4.3 Synergy between Fuzzy Systems and Evolutionary Algorithms

Our knowledge of EAs has significantly increased in recent years due to the vast amount of theoretical and empirical studies conducted on a large number of complex problems in different fields. It would be desirable to profit from this human knowledge by encapsulating it within an algorithm so as to automate the task of improving the behaviour and performance of EAs. However, this sort of knowledge is usually incomplete, imprecise, and it is not well organised. Consequently, the application of *fuzzy systems* would seem to offer a promising approach for dealing with this kind of knowledge. Fuzzy systems can be regarded as universal approximators of real continuous functions in a compact set to arbitrary accuracy [112]. They have been applied to several fields, such as control [100], classification [169], regression [73] and general data mining problems, due to their ability to manage imprecision and uncertainty, as well as to describe the behaviour of complex systems without requiring a precise mathematical model. The most common fuzzy models consist of a set of logical fuzzy rules and are known as *Fuzzy Rule-based Systems (FRBSs)*. The two most popular FRBSs are linguistic fuzzy models, which are also referred to as

Mamdani [228] fuzzy models, and *Takagi-Sugeno-Kang (TSK)* [325] fuzzy models. The main difference between these two types of fuzzy models lies in the form that the consequents of their fuzzy rules take. More details on the two kinds of fuzzy models will be given in Section 4.3.1.

A considerable body of research on fuzzy systems and EAs already exists [112, 154, 165, 303]. EAs have been used to automatically design fuzzy systems, and particularly FRBSs, since the design process can be viewed as an optimisation problem. Hybrid approaches that combine the use of an EA to design a fuzzy system are known as *evolutionary fuzzy systems*. When using GAs, these kinds of approaches are called *genetic fuzzy systems* [154], whereas if a MOEA is used as the optimiser, the methods are then termed *multi-objective evolutionary fuzzy systems* [112].

Fuzzy Logic Controllers (FLCs) are FRBSs designed to carry out control tasks. There is no set procedure for designing an FLC. Furthermore, the fuzzy parameters corresponding to the FLC must be properly selected in light of the control objective. In order to deal with these issues, the application of EAs has been proposed for designing FLCs [174]. The design of an FLC by means of an EA could be based on [112]:

- *Learning a structure for the controller.* Learning the set of fuzzy rules, which is also referred to as the *rule base*.
- *Identifying the numeric parameters of the controller.* Tuning the parameters belonging to the membership functions and/or performing rule selection as a post-processing method.

Different EAs, including MOEAs, have been used to identify the parameters of FLCs, as well as to design their structure, taking into account different applications [62, 67, 128, 129, 146, 199, 259, 290, 301, 333]. For instance, in [62] a GA is used to design an FLC for a mobile robot. In [333], a DE algorithm is employed to systematically tune the optimal parameters of an FLC whose aim is to control a power system stabiliser. Another example is given in [301], where an ES approach is used to optimise an FLC designed to deal with the optimal antiviral therapy problem of infectious diseases. Lastly, in [129], a tuning process is combined with a rule selection process to improve the performance of an FLC to control heating, ventilating, and air conditioning systems. The approach is based on the SPEA2.

In this dissertation, the reverse of the above type of application is analysed; namely, the design of FLCs that adapt the parameters of an EA is studied to provide an adaptive control technique that utilises feedback from the search process to adapt these parameters. These hybrid systems, where an FLC is used to adapt the pa-

rameters of an EA, are also termed *fuzzy adaptive EAs*, and in the particular case of controlling GAs, *fuzzy adaptive GAs* [155]. Both Mamdani-type and TSK-type FLCs are employed herein as adaptive parameter control methods, that dynamically change the parameters of an EA. Unlike other FLCs, an error feedback signal does not directly exist in the approaches proposed throughout this research work, since their goal is to optimise as much as possible. However, these systems have to include some way of measuring performance, which makes them atypical since the desired objective is not known beforehand. Despite this fact, these kinds of schemes are also called FLCs in the literature [114, 155], which is why this term has been adopted here. The main benefit of using FLCs to adapt the parameters of an EA is that the possible values that can be assigned to certain parameters are infinite, in contrast to other techniques that can only use certain values from a finite set. The main drawback is that FLCs cannot be directly applied to control the symbolic parameters of an EA. Hence, in this thesis FLCs are applied to adapt numeric parameters of EAs.

Several methods have been proposed for controlling the parameters of an EA through FLCs. The principle behind these schemes is to use an FLC to compute new parameter values by considering any combination of performance measures and current parameter values as the input to the controllers. This idea was first proposed in [201]. In this scheme, the population size and the mutation and crossover rates were adapted considering the best, the mean, and the worst fitness values of the individuals in the population. Subsequently, a large number of variants have been proposed [135]. Some of the schemes that are more closely related to the FLCs applied in this dissertation are briefly summarised below.

A survey of several optimisation schemes, including EAs, which relied on an FLC to control their internal parameters, was presented in [135]. Some of the simplest approaches adapted the crossover and mutation rates of a GA by considering the mean fitness of the last two generations as input variables to the controller [344]. Basically, depending on the improvements obtained in the last generation, the crossover and mutation probabilities were modified so as to change the perturbation strength of the variation scheme. Using only two generations might not be enough, so in some schemes the mean fitness values of the last three generations were considered [210]. A similar idea was proposed in [211] to adapt the parameters of a DE approach. Specifically, two FLCs were used to adapt the mutation scale factor and the crossover rate. In this case, the input variables not only consisted of measures in the objective space, but the decision space was also considered.

More advanced FLCs take into account diversity metrics as inputs to the FLCs in order to carry out their decisions. For instance, a fuzzy adaptive search method for

parallel GAs based on the use of diversity measures was proposed in [226]. In [42] the frequency of the best individual, as well as the rate of duplicate individuals were used to control the mutation, crossover and surviving individual rates. In addition, an input variable that estimated the quality of the resulting fitness was used. Thus, as in some of the other schemes described, knowledge was assumed regarding the supposed optimal values. Another example is given in [207], where a diversity metric was calculated by considering the difference between the maximum and mean fitness of the population.

From the perspective of MOEAs, it is apparent that the body of research is much smaller than in the case of single-objective EAs. However, FLCs have also been used to control the parameters of different MOEAs [68, 114]. For example, in [114] an FLC is used to dynamically adapt the greediness and the perturbation scale factor belonging to the reproduction operator of a multi-objective DE approach. We should note that even though in this dissertation diversity-based MOEAs are applied, the original objective of the problem being solved is the only one considered. As a result, most of the ideas proposed for controlling multi-objective schemes with FLCs cannot be directly implemented into the approaches being proposed here.

It is also worth noting that FLCs have been successfully used to adapt different EAs applied to real-world applications, demonstrating their efficiency and reliability even for complex problems [68, 113, 291, 313, 321]. For instance, in [313] two FLCs are used to adaptively adjust the crossover and mutation rates of a GA that is applied to a power system environmental/economic dispatch. Another example is detailed in [291], where a MOEA is used to solve a complex multi-objective power market clearing problem for the competitive electricity market environment. An FLC is used to dynamically update the crossover and mutation rates.

Summarising, the feature common to most of the research described in the literature is that FLCs are used to adapt the parameters of GAs [155]. Specifically, the crossover and mutation rates, the population size, or combinations of all three [147, 155, 240, 320] are controlled. Moreover, these FLCs are usually tailor-made methods for a particular EA and/or parameters, and they only make use of a single rule base. Chapter 6 will be devoted to discussing innovations on the FLCs proposed in this research work with respect to the most common approaches present in the literature. Lastly, the next section will go into detail on the different components that an FLC consists of, as well as on the set of steps required in order to design an FLC that can be used to adaptively adjust the parameters of an EA.

4.3.1 Adapting Evolutionary Algorithms using Fuzzy Logic Controllers

FLCs are useful for complex applications that are difficult to analyse by quantitative methods, or in cases where the sources of information are interpreted qualitatively, inexactly, or uncertainly. One of the main advantages of an FLC is that it is modelled in *linguistic terms*. Hence, the human knowledge can be easily represented. As a result, using them to dynamically control the parameters of an EA seems to be a very promising idea. An FLC consists of the following components [155]:

- *Knowledge base*. This contains the human knowledge in the form of linguistic fuzzy control rules.
- *Fuzzification interface*. This transforms crisp input data into fuzzy sets.
- *Fuzzy inference engine*. This element is in charge of performing inference based on the knowledge base and the fuzzy sets given by the fuzzification interface.
- *Defuzzification interface*. Responsible for transforming fuzzy control actions to real control actions by using a defuzzification approach.

Figure 4.3 shows the general architecture of an FLC. The knowledge base has two different parts. On the one hand, a *data base*, which includes the definitions of the membership functions of the linguistic terms for each input and output linguistic variable. And on the other hand, a *rule base* constituted by the collection of fuzzy rules, which represents the human knowledge.

Input and output linguistic variables are expressed by a set of linguistic terms. For instance, a variable called *fan speed* might be represented by means of the linguistic terms *low*, *medium*, and *high*. The set of terms belonging to a given variable is a *fuzzy set* when each value of the linguistic variable belongs with degrees of membership to one or more of its linguistic terms based on their corresponding membership functions. For instance, suppose that the linguistic variable *fan speed* is modelled as shown in Figure 4.4. From left to right, triangular-shaped membership functions represent the linguistic terms *low*, *medium*, and *high*. If this linguistic variable takes the crisp value $f = 25$, then the corresponding fuzzy set will be $\bar{f} = 0.5/\text{low} + 0.5/\text{medium} + 0/\text{high}$.

A *fuzzy control rule* is a conditional statement of the form “*IF a set of conditions are satisfied THEN a set of consequences can be inferred*” [155]. The antecedent can include different conditions, while the consequent is a collection of control actions to be applied to the controlled system, and both antecedent and consequent are

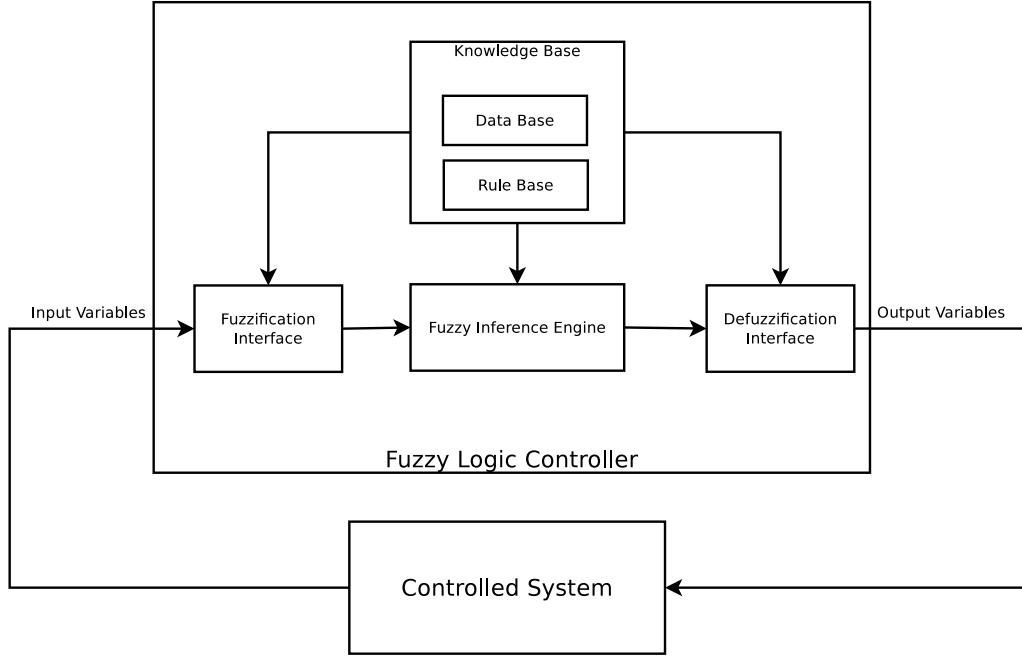


Figure 4.3: General architecture of a fuzzy logic controller

associated with fuzzy concepts, i.e. linguistic terms. Usually, the input variables are located in the antecedents of the fuzzy rules, whereas the output variables are located in the consequents. In this thesis, where an FLC has multiple inputs and a single output, fuzzy rules take the following form:

$$\begin{aligned}
 R_1 &: IF \ x \text{ is } A_1, \dots, AND/OR \ y \text{ is } B_1 \ THEN \ z \text{ is } C_1 \\
 R_2 &: IF \ x \text{ is } A_2, \dots, AND/OR \ y \text{ is } B_2 \ THEN \ z \text{ is } C_2 \\
 &\dots \\
 R_n &: IF \ x \text{ is } A_n, \dots, AND/OR \ y \text{ is } B_n \ THEN \ z \text{ is } C_n
 \end{aligned} \tag{4.1}$$

In the above equation, x, \dots, y are input variables, while z is the output variable. A_i, \dots, B_i , and C_i are linguistic terms defined for the linguistic variables x, \dots, y , and z , respectively. Furthermore, the importance of a rule can be determined by a weight in the range $[0, 1]$, which is equal to 1 if omitted. In a more general variant of a fuzzy rule, the consequent is represented by a function of the input variables x, \dots, y :

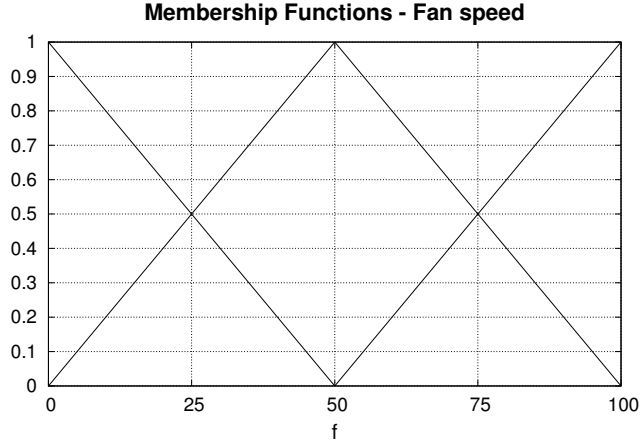


Figure 4.4: Membership functions for the linguistic variable *fan speed*

$$R_i : IF\ x\ is\ A_i, \dots, AND/OR\ y\ is\ B_i\ THEN\ z = f_i(x, \dots, y) \quad (4.2)$$

Note that in the above definitions of fuzzy rules, the different conditions or prepositions in the antecedents can be connected by the conjunctive fuzzy logic operator *AND* or by the disjunctive fuzzy logic operator *OR*. A *fuzzy logic operator* defines an operation between two values. It can be a *T-norm* when it models conjunction or an *S-norm* when it models disjunction. A simple example of a fuzzy rule set might be the following, where the input variable *temperature* and the output variable *fan speed* are taken into account:

$$\begin{aligned} R_1 &: IF\ temperature\ is\ low\ THEN\ fan\ speed\ is\ low \\ R_2 &: IF\ temperature\ is\ medium\ THEN\ fan\ speed\ is\ medium \\ R_3 &: IF\ temperature\ is\ high\ THEN\ fan\ speed\ is\ high \end{aligned} \quad (4.3)$$

Configuration of a Fuzzy Logic Controller

It is important to remark that the current and the following sections will consider Mamdani-type FLCs. However, since in this dissertation TSK-type FLCs are also applied, their differences with respect to Mamdani-type FLCs will be discussed when needed. In order to design an FLC to adapt the parameters of an EA, the following steps are required:

1. *Define the input and output variables.* Input variables should be robust metrics which indicate the behaviour and performance of the EA at each stage of the optimisation process. Several metrics, including diversity measures, have been proposed [155]. Outputs are defined as absolute or relative values for the parameters that are being controlled.
2. *Define the data base.* Firstly, a range of possible values—the *universe of discourse*—has to be assigned to every input and output variable. Then, the set of membership functions for every input and output variable has to be defined over its corresponding range. To do so, the shape of each membership function has to be selected. There are different types of shapes for the membership functions, such as singleton, triangular, or trapezoidal shapes, among others. Lastly, each membership function must be associated with a linguistic term.
3. *Obtain the rule base.* The fuzzy rules describing the human knowledge have to be obtained. They can be created by using the knowledge of an expert in EAs, or by some automatic approach. Automatic approaches can be grouped into *offline learning* methods and *online learning* methods [155].

Additionally, there are some components that must be specified in order to completely define an FLC. They are the following:

- *Fuzzy logic operators.* A T-norm and an S-norm must be selected for the fuzzy logic operators AND and OR, respectively. Examples of T-norms are the *minimum* and the *algebraic product*, whereas the *maximum* and the *algebraic sum* are instances of S-norms.
- *Implication or activation method.* It allows the consequents of the fuzzy rules to be reshaped. It has to be a T-norm.
- *Aggregation or accumulation method.* It allows the fuzzy outputs of the fuzzy rules to be aggregated. It has to be an S-norm.
- *Defuzzification interface.* As in the case of the aforementioned components, there are several defuzzification interfaces that can be selected by the user. One of the most frequently used approaches is the *centroid* method, which computes a crisp value from the centre of mass of a fuzzy set. Other widely applied methods are the *bisector*, the *middle of maximum*, the *largest of maximum*, and the *smallest of maximum*.

Fuzzy Inference Process

Considering the different components that must be specified so that an FLC works, the following set of steps is defined for use during the fuzzy inference process:

1. *Fuzzify input variables.* This step consists of taking the crisp values of the input variables and determining the degree of membership to each linguistic term by means of the membership functions. The input to the fuzzification interface is always a crisp value, while the output is a number representing the degree of membership to the corresponding linguistic term. This fuzzification step must be performed for each existing preposition in the antecedent of every fuzzy rule.
2. *Apply the fuzzy logic operator.* Once the inputs are fuzzified, the degree of membership to which each preposition of the antecedent is satisfied for each rule is known. If more than one preposition exists in the antecedent of a rule, the fuzzy logic operator is then applied to obtain one value that represents the result of the antecedent for that rule. The input to the fuzzy logic operator consists of two or more degrees of membership calculated during the fuzzification of the input variables, whereas the output is a single truth value. This step has to be repeated for each fuzzy rule.
3. *Apply the implication method.* Before applying the implication method, the *activation degree* for each fuzzy rule must be determined. It is important to recall that every rule has an importance weight assigned, which is multiplied by the value of the antecedent obtained after the application of the fuzzy logic operator so as to determine the activation degree. In this dissertation, importance weights are omitted, and consequently the activation degree and the value of the antecedent are equal. Once the activation degree is calculated, the implication method is applied and the consequent of a given fuzzy rule is reshaped using certain functions together with the activation degree. The input to the implication method is the activation degree, and the output is a reshaped fuzzy set. The implication method has to be applied for each fuzzy rule.
4. *Aggregate all outputs.* All the fuzzy rules must be combined so as to make a decision. By using the aggregation method, the fuzzy sets representing the outputs of each rule are combined into a single fuzzy set. The input to the aggregation method is the list of reshaped fuzzy sets given for each rule by the implication method. The output is one aggregated fuzzy set for each output variable.

5. *Defuzzify*. After the aggregation method obtains a single fuzzy set, it is used as the input to the defuzzification interface, whose output is a single number. This step must be performed for each output variable.

The aforementioned inference process is suitable for Mamdani-type FLCs. The inference process used by TSK-type FLCs is very similar. The first two steps—fuzzification of the input variables and application of the fuzzy logic operator—are exactly the same. However, the implication method is a little different, and an aggregation method is not taken into consideration. Moreover, in TSK-type FLCs, the linguistic terms of the output variables are represented by polynomial functions that depend on the input variables, instead of using membership functions as in the case of Mamdani-type FLCs. As a result, in TSK-type FLCs, fuzzy rules take the form shown in Equation 4.2. The order of a TSK-type FLC is given by the maximum degree of the polynomial functions used to represent the linguistic terms of the output variables. Note that zero-order TSK-type FLCs, where the linguistic terms belonging to the output variables are described by zero-order—constant—functions, are assumed in this thesis. Continuing with the example of the output variable *fan speed*, it might be modelled as *low* = 0, *medium* = 50, and *high* = 100, considering a zero-order TSK-type FLC. It can be observed that in this example, the linguistic terms are represented by constant functions.

The defuzzification method is also different because the outputs of the implication method are not reshaped fuzzy sets, but are instead values given by the polynomial functions. The typical defuzzification approaches are the *weighted average* and the *weighted sum*. The weighted average and weighted sum are computed from the values given by the polynomial functions using the activation degrees as weights. Recall that in the case of this research work, the activation degrees are equal to the values of the antecedents since importance weights are omitted. Therefore, the weights used to compute the weighed average and the weighted sum are given by the values of the antecedents.

4.4 Hyper-heuristics as Parameter Control Methods

Hyper-heuristics are methods whose goal—in part—is to automate the design and tuning of heuristic approaches to solve complex search problems [50]. The term hyper-heuristic was introduced by Cowling et al. [80] to describe “*heuristics to choose heuristics*” in the field of optimisation. Hyper-heuristics operate on a search space

of heuristics instead of directly operating on a search space of solutions to the underlying problem being solved. As a result, hyper-heuristics are designed bearing in mind the generality of their application to different problem domains, rather than being developed to deal with a particular application. Hyper-heuristics were founded on two main premises. Firstly, the process of selecting or generating heuristics can be treated as an optimisation problem itself. Secondly, optimisation schemes might be improved by the incorporation of learning mechanisms that allow the search process to be guided. Hyper-heuristics can be classified into two main groups: hyper-heuristics based on heuristic selection, and hyper-heuristics based on heuristic generation [48]. Hence, the following definition of a hyper-heuristic was proposed by Burke et al. [48].

Definition 20 “A **hyper-heuristic** can be viewed as a search method or learning mechanism for selecting or generating heuristics to solve computational search problems”.

Most of the research on hyper-heuristics has been carried out considering single-objective optimisation problems [46]. For instance, in [171] a selection hyper-heuristic was proposed for dealing with single-objective variants of six different problems: boolean satisfiability, one dimensional bin packing, personnel scheduling, permutation flow shop, the VRP, and the TSP. One of the most widely applied type of hyper-heuristics includes approaches inspired by certain meta-heuristics. For instance, a TS-based hyper-heuristic was proposed in [51]. In this hyper-heuristic, every low-level approach represents a definition of a neighbourhood that, when unable to provide any improvement, is inserted into a tabu list. Another example was given in [98], where a hyper-heuristic inspired by a SA algorithm was introduced. Lastly, a GA-based hyper-heuristic was presented in [78]. Every individual in the population is encoded as a sequence of heuristic choices that indicate the low-level approach to be applied at any given moment in the search procedure. The most common approach for designing hyper-heuristics that automatically generate heuristics is GP. GP is an EC technique that evolves a population of individuals that encode programs. If these programs represent heuristics that are then applied to solve a given problem, then GP can be viewed as a hyper-heuristic based on heuristic generation. In a recent research paper [47], a GP-based hyper-heuristic, which generates highly efficient local search methods for dealing with cutting and packing problems, was introduced. Other frequently used hyper-heuristics are designed by means of *choice functions* [34, 79, 180, 182, 198, 339]. In these cases a scoring function assigns to each low-level approach a score that usually represents its historical performance. Therefore, more resources—usually in the form of compu-

tational time—are allocated to the low-level approach, which maximises the scoring function. For example, a hyper-heuristic that incorporates a choice function to carry out its decisions was used to deal with problems in the field of financial forecasting [180]. Another hyper-heuristic based on choice functions, which was applied to predict Deoxyribonucleic Acid (DNA) sequences, was proposed in [34]. In this particular case, different configurations of a TS algorithm are treated as the low-level approaches. In [339] a hyper-heuristic built on choice functions was applied. In that paper the assignment of resources was probabilistic rather than deterministic. Thus, the higher the score assigned to a low-level approach, the higher the probability that said approach will receive a larger number of resources.

The amount of work on hyper-heuristics specifically designed for MOPs is not very extensive [46]. Nevertheless, there are some hyper-heuristics specifically designed to deal with multi-objective approaches [86, 268, 336]. In [336], a GA-based hyper-heuristic was employed to solve a multi-objective variant of the JSP. In this scheme, individuals are represented as sequences of dispatching rules that are called one at a time and used to sequence a number of operations onto machines. The approach simultaneously searches for the best sequence of rules and for the number of operations to be handled by each rule. Another example is given in [86], where a hyper-heuristic based on a choice function chooses from a set of different encodings incorporated into several MOEAs to solve multi-objective versions of two-dimensional strip packing and cutting stock problems. Finally, a multi-objective GP-based hyper-heuristic to automate the design of scheduling policies in JSP environments was presented in [268].

With respect to parallel, cooperative or distributed hyper-heuristics, some schemes have been proposed [32, 273, 282]. Nevertheless, the amount of research in this area is not very expansive. For instance, in [273], a cooperative hyper-heuristic framework was proposed. Different low-level heuristic methods perform a search in the same solution space and cooperate synchronously or asynchronously through the cooperative hyper-heuristic, which is responsible for selecting the best-behaved low-level approaches depending on the solutions provided, as well as for exchanging solutions among said low-level methods.

Several taxonomies have been presented in the literature for grouping the existing hyper-heuristics [21, 22, 49, 64, 314]. The classification proposed by Burke et al. [48] offers a unified view of these taxonomies by taking into consideration two different aspects to classify hyper-heuristics: the search space of heuristics and the feedback source. According to the search space of heuristics, hyper-heuristics can be grouped into:

- *Hyper-heuristics based on heuristic selection.* This type of approach tries to identify and select the most promising method—from among a set of candidate low-level heuristics—to be applied at any point in the search process in order to solve a particular optimisation problem.
- *Hyper-heuristics based on heuristic generation.* This kind of method tries to automatically generate heuristics to solve a specific optimisation problem. Usually, the heuristics are generated by combining different components.

A distinction is also made at this level between constructive and perturbation hyper-heuristics in that hyper-heuristics can be used to select or generate constructive or perturbation low-level heuristics. Constructive heuristics incrementally build solutions from scratch, while perturbation or improvement heuristics start with complete solutions and at each step, try to improve the solutions by modifying them.

With respect to the feedback source, hyper-heuristics can be categorised into three groups:

- *Online learning hyper-heuristics.* They learn to select or to generate heuristics while solving an instance of a problem.
- *Offline learning hyper-heuristics.* They typically learn a mapping between characteristics of a problem—or partially solved problem—during a training phase which can then be applied to unsolved instances.
- *No-learning hyper-heuristics.* They do not use feedback from the search procedure and generally apply a prefixed sequence of heuristics.

In this dissertation, online learning hyper-heuristics based on heuristic selection are applied. Hence, from now on, the term hyper-heuristic will refer to methods that iteratively choose from among a set of candidate low-level heuristics or meta-heuristics to solve an optimisation problem [50]. These methods learn to carry out their choices while solving the optimisation problem at hand. Hyper-heuristics operate at a higher level of abstraction than traditional heuristics because they have no knowledge of the problem domain. The underlying principle in using hyper-heuristics is that different heuristics or meta-heuristics have different strengths and weaknesses, and it makes sense to combine them intelligently. The motivation behind the approach is that, ideally, once a hyper-heuristic is designed, several optimisation problems and/or instances of a problem might be addressed by only replacing the set of low-level heuristics or meta-heuristics. As a result, the aim of using a hyper-heuristic is to raise the level of generality at which the majority of current heuristic approaches operate [50]. Finally, it is worth mentioning that by properly combining the advan-

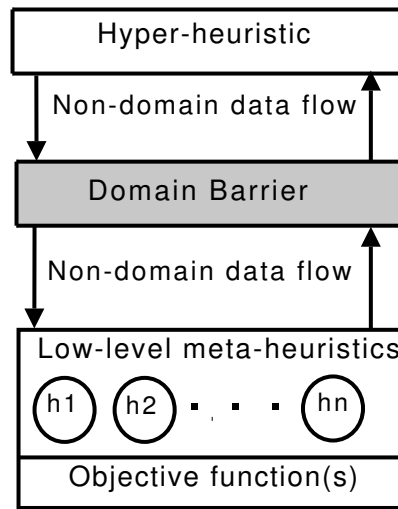


Figure 4.5: General framework of a hyper-heuristic

tages provided by the different low-level approaches, a hyper-heuristic might obtain better results than those given by any of the low-level methods executed separately.

Figure 4.5 shows the general framework of a hyper-heuristic [50]. It shows a problem domain barrier between the low-level meta-heuristics and the hyper-heuristic. The data flow received by the hyper-heuristic might include data on the quality of the solutions or on the resources employed to achieve those solutions, for instance. The hyper-heuristic uses this information to carry out its decisions. The data flow coming from the hyper-heuristic might contain data on the meta-heuristic that must be applied, or even on its parameterisation, for example.

Hyper-heuristics are highly correlated to the problem of parameter control [309]. For example, the low-level approaches might represent different configurations of the same meta-heuristic. The hyper-heuristic then would select the configuration with the most appropriate set of parameters at each point in the search. In fact, hyper-heuristics can be further classified as adaptive parameter control techniques if they receive some kind of feedback from the optimisation process in order to intelligently select the most suitable low-level method. Hyper-heuristics are independent of the methods adapted, and therefore they can be designed to control a wide range of approaches. In those cases where the best configuration, in terms of performance, of the same meta-heuristic varies depending on the current stage of the optimisation process, the hyper-heuristics could be used to select the most suitable configuration for each stage. Thus, it seems reasonable to expect the results obtained by the hyper-heuristic to be better than those obtained by any of the candidate low-level

configurations executed independently. Furthermore, the use of a hyper-heuristic would permit low-level configurations to have variations in both their numeric and symbolic parameters, thus providing a straightforward mechanism for symbolic parameter control. However, the main drawback of the hyper-heuristic approach is the need to specify the set of candidate low-level configurations. Moreover, since the size of the set of candidate low-level approaches is generally fixed and finite, in the case of controlling numeric parameters, the number of possible values that can be assigned to the numeric parameters is therefore also finite. Despite this, hyper-heuristics have successfully been applied as adaptive parameter control methods in several problem domains. For instance, in [193], a hyper-heuristic was used to dynamically select which EA to apply—between a GA and a DE approach—and which operator to employ—from a total amount of five operators for the GA and four operators for the DE—so as to generate improved solutions. The method was tested on a large set of complex benchmark functions. Another research paper proposed a hyper-heuristic whose novelty lay in the fact that the parameters of the low-level approaches, along with the automatic control of which low-level approach was executed at any point in the search process, were dynamically changed during the optimisation procedure [285]. In order to validate this proposal, several instances of the p-median problem were considered.

We should note that in this research work, although the parameters of different MOEAs are controlled, the hyper-heuristics applied are suitable for single-objective schemes. Diversity-based objectives and multi-objectivisation by aggregation are used herein as techniques for dealing with single-objective optimisation problems. As a result, the optimisation problems consist of two objective functions: the original objective function belonging to the single-objective problem and an auxiliary objective function. However, the hyper-heuristics used throughout this dissertation only take into account the original objective function when making their decisions. In Section 6.2, their functioning will be explained.

Part II

Algorithmic Proposals

Advances in Diversity-based Multi-Objective Evolutionary Algorithms

This chapter is focused on describing the novel diversity-based approaches that are proposed in this dissertation. First, a set of completely new diversity-based objectives is introduced in Section 5.1. These diversity-based objectives are integrated with some of the multi-objective optimisation schemes detailed in Chapter 2 to provide a set of novel diversity-based MOEAs for single-objective optimisation. The main benefit of these new diversity-based objectives with respect to the existing proposals in the literature is the inclusion of parameters that improve the performance of the whole diversity-based MOEA. The main disadvantage is that these parameters must be fixed by the user, and their values depend on the problem and/or instance at hand. This issue, however, can be mitigated by using the parameter control schemes presented in Chapter 6. Second, and considering some of the notions exposed in the first section of this chapter, a novel diversity-based survivor selection scheme is presented in Section 5.2. Although diversity-based MOEAs are proposed as an efficient method for preserving diversity in a population of individuals, some combinations of certain MOEAs with certain diversity-based objectives might involve the appearance of premature convergence issues. The novel survivor selection mechanism proposed herein aims to avoid this drawback.

5.1 Diversity-based Objectives with Parameters

In Section 3.1.2, different diversity-based objectives that make use of the Euclidean distance in the genotypic space as a measure of diversity were introduced. These diversity-based objectives are ADI, DBI, and DCN. These approaches have been in-

egrated with various meta-heuristics and applied to different optimisation problems to yield high-quality results in almost every case. As stated earlier, these diversity-based objectives have to be maximised. Consequently, diversity is promoted in the population because individuals are far from each other in the decision space. Nevertheless, a greater distance does not involve a higher quality of individuals. As a result, if the quality of the individuals is very low, the convergence speed of the whole optimisation scheme might be delayed by the use of these diversity-based objectives, even if the diversity of the population is maintained.

In this dissertation, extensions to the ADI, DBI, and DCN approaches are proposed in an effort to deal with the aforementioned drawback, while simultaneously bearing in mind the aim of promoting good diversity among individuals. The novelty of these extensions lies in the fact that they attempt to limit survival of individuals of very low quality through the use of a *threshold ratio* $th \in [0, 1]$, which has to be specified by the user. These new diversity-based objective functions are called *Average Distance to all Individuals with Threshold (ADI-THR)*, *Distance to Best Individual with Threshold (DBI-THR)*, and *Distance to Closest Neighbour with Threshold (DCN-THR)*. From now on, and without loss of generality, the diversity-based objective DCN-THR will be considered to explain the operation of the new proposals.

Firstly, it is important to recall that in this research work two different objective functions are concurrently optimised through the application of a diversity-based MOEA: the original objective function belonging to the single-objective problem at hand and the diversity-based objective function. Taking this into account, the DCN-THR scheme defines a threshold v , which is used to penalise individuals that have low quality with respect to the original objective function by assigning a diversity-based objective value of 0 to such individuals. The threshold v is calculated as follows. If *bestObjectiveValue* is the *original* objective value of the fittest individual in the population, and *shift* is a value that ensures that $bestObjectiveValue - shift \geq 0$ during the whole optimisation procedure, then the threshold value v is defined as:

$$v = \frac{(bestObjectiveValue - shift)}{th} + shift \quad (5.1)$$

Note that the above equation is suitable for an optimisation problem where the original objective has to be minimised. For problems where the original objective has to be maximised, the threshold value v is calculated by means of the following equation:

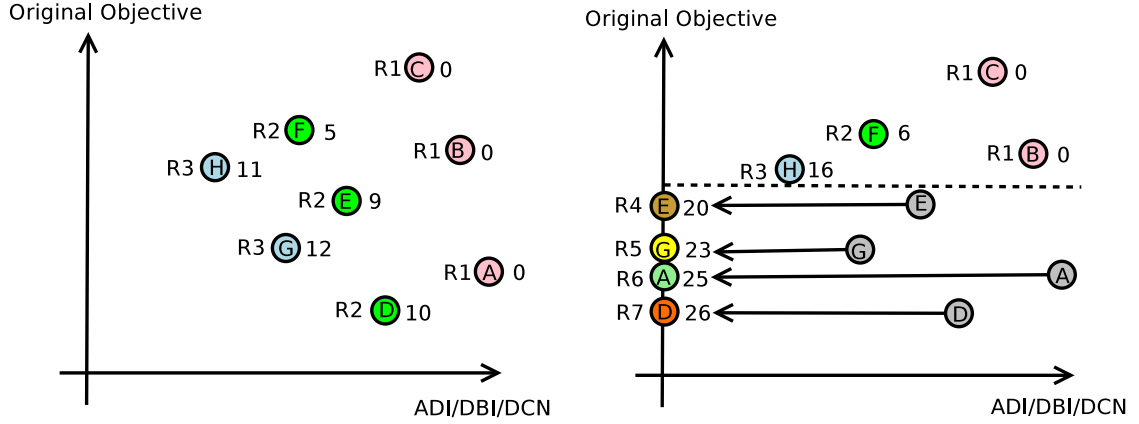


Figure 5.1: Behaviour of the Non-Dominated Sorting Genetic Algorithm II and the Strength Pareto Evolutionary Algorithm 2 combined with diversity-based objectives without threshold (left-hand side) and with threshold (right-hand side)

$$v = (bestObjectiveValue - shift) \cdot th + shift \quad (5.2)$$

After the threshold value is calculated, all individuals whose original objective value is worse than v —higher for a minimisation problem; lower for a maximisation problem—have the value of their diversity-based objective assigned to 0. For the remaining individuals, their diversity-based objective value is calculated as DCN, i.e. the distance to the closest neighbour. Consequently, individuals that are not able to attain the fixed threshold are penalised. In the special case where $th = 0$, individuals are never penalised. Therefore, DCN-THR with $th = 0$ behaves like the DCN approach.

In this thesis, diversity-based objectives are combined with the NSGA-II and the SPEA2, which were described in Section 2.4.1. The left-hand side of Figure 5.1 shows the behaviour of the ADI, DBI, and DCN schemes when they are integrated with both MOEAs. The maximisation of the original and the diversity-based objective functions is assumed. To the left of every candidate solution is a label showing the rank assigned by the NSGA-II. Hence, the label R_i means that the corresponding candidate solution has a rank number i . Moreover, the corresponding raw fitness assigned by the SPEA2 is also shown to the right of every candidate solution. The right-hand side of Figure 5.1 shows the effect of incorporating the threshold to the ADI, DBI, and DCN diversity-based objectives. The dashed line represents the threshold value v . Note that every candidate solution that does not fulfil the

minimum quality level established by the threshold value is shifted in the objective space. Specifically, a value equal to 0 is assigned to the diversity-based objective. The effect of the shift is that the corresponding candidate solution will usually belong to a worse rank in the case of the NSGA-II. In the case of the SPEA2, a higher raw fitness will be assigned to the corresponding candidate solution, thus usually decreasing its survival probability.

Through the appropriate use of the threshold ratio th , poor quality individuals can be discarded while and at the same time the goal of ensuring good diversity in the population can be satisfied. The main disadvantage of these novel approaches, however, is that the most adequate values for the threshold ratio th might differ depending on the problem and/or instance being solved. Additionally, as is the case with other parameters belonging to EAs, the most suitable values for th might depend on the current stage of the optimisation process. As a result, a fixed value of th might not be suitable, meaning that this parameter might have to be adjusted dynamically to yield high-quality results. To carry out this task, different parameter control schemes are applied in this dissertation so as to control the threshold ratio th . Particularly, this parameter is adapted by the application of FLCs, hyper-heuristics, and self-adaptation. Moreover, a considerable number of configurations of the diversity-based MOEAs with fixed values for the parameter th are also considered in an effort to carry out a broad comparison between parameter tuning and parameter control.

5.2 Diversity-based Survivor Selection Scheme

Applying the diversity-based objectives proposed in the previous section might yield high-quality solutions. Different analyses should still be carried out, however, in order to reveal whether premature convergence problems persist when employing them.

For instance, consider the single-objective function with one decision variable shown in Figure 5.2. Each black dot represents an individual belonging to the current generation of a certain diversity-based MOEA. Every individual could be a member of the parent population or a member of the offspring population. This simple function, which must be minimised, presents two local maxima that separate the fitness landscape into three different regions. Furthermore, note that every region is covered by two individuals, meaning that proper diversity is maintained by the diversity-based optimisation scheme, at least for now. Assuming that the parent

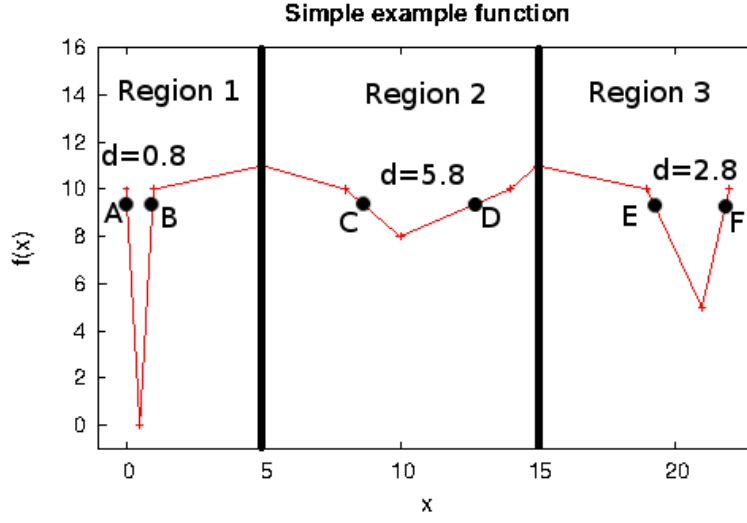


Figure 5.2: Simple function where diversity preservation problems arise

population size N is fixed to three individuals, the survivor selection mechanism is then responsible for selecting three individuals, among the current parents and offspring, which will form part of the parent population for the next generation. Consequently, an individual from every region should be selected so as to preserve diversity in the population. Finally, with the aim of keeping the example as simple as possible, note that the original objective— $f(x)$ —of every individual in Figure 5.2 has the same value. However, similar situations would occur even if the original objective values of the individuals in question were different. Hence, in this particular example, the diversity-based objective determines which individuals are selected through the survivor selection scheme.

In the case of applying the NSGA-II in combination with one of the diversity-based objectives, DCN or DCN-THR, the distance to the closest neighbour, which must be maximised, has to be computed. Bearing this in mind, the crowded comparison operator of the NSGA-II assigns the individuals C and D to the first front—Table 5.1—since the diversity-based objective value of both individuals takes the value $d = 5.8$, which is in fact the maximum distance to the closest neighbour in this example. Afterwards, in the same way, the individuals E and F—those with the second largest distance—are assigned to the second front, and finally, the individuals A and B are assigned to the last front. Therefore, the three individuals selected by the crowded comparison operator are C, D, and E or F. We can thus see that proper diversity is not maintained in some cases, since no individual belonging to the first

Table 5.1: Assignment of the non-domination rank by a diversity-based optimiser using the Non-Dominated Sorting Genetic Algorithm II

Individual	Closest Neighbour	Distance	Non-domination Rank
A	B	0.8	3
B	A	0.8	3
C	D	5.8	1
D	C	5.8	1
E	F	2.8	2
F	E	2.8	2

region survives until the next generation, a fact that might lead to the appearance of premature convergence problems.

The main reason for the improper behaviour of the NSGA-II with the DCN and DCN-THR approaches is that these diversity-based objectives are built upon a direct measure of the diversity introduced by every individual in the current generation. As a result, individuals should not be selected without considering the previously selected survivors.

In order to overcome this drawback and improve diversity, a novel survivor selection scheme called *Distance to Closest Neighbour based on Reference Set (DCN-REF)* is proposed in this dissertation. Its operation is shown in Algorithm 6. First, the fittest individual in the population, i.e. the one with the best original objective value among parents and offspring, is selected to survive, thus ensuring elitism in the approach—lines 1–4. In case of a tie, the fittest individual is randomly selected. Afterwards, while the population of the next generation—*NewPop*—is not filled with N individuals—line 5—the following steps are repeated. First, the diversity-based objective DCN or DCN-THR is recalculated—line 6. This recalculation procedure considers the currently selected individuals as the reference set, i.e. for each non-selected individual in *CurrentMembers*, the distance to its closest neighbour in *NewPop* is calculated. It is important to remark that in order to update the diversity-based objective of the individuals belonging to *CurrentMembers*, only the distances between these individuals and the last individual introduced in *NewPop* have to be computed. Then, the non-dominated individuals from *CurrentMembers* are calculated—line 7. Finally, one non-dominated individual is randomly chosen to survive for the next generation—lines 8–10. Therefore, the stochastic behaviour of the new survivor selection scheme might contribute even more to avoid the problem of premature convergence.

In order to show the proper behaviour of the novel survivor selection scheme DCN-

Algorithm 6 Pseudocode of the novel diversity-based survivor selection scheme

```

1: Initialise CurrentMembers with every individual belonging to the current parent and
   offspring populations
2: Best = Individual with the best original objective value from CurrentMembers. In
   case of a tie, it is randomly selected
3: NewPop = {Best}
4: CurrentMembers = CurrentMembers − {Best}
5: while ( $|NewPop| < N$ ) do
6:   Calculate the DCN or DCN-THR diversity-based objective for every individual in
     CurrentMembers considering NewPop as the reference set
7:   ND = Non-dominated individuals from CurrentMembers
8:   Selected = Randomly select an individual from ND
9:   NewPop = NewPop ∪ {Selected}
10:  CurrentMembers = CurrentMembers − {Selected}
11: end while

```

REF, its application to the example of the single-objective function—Figure 5.2—is described in the following lines. Since every individual takes the same value for the original objective function, the first individual is randomly selected. Assuming that the first individual selected is E, then since A is the farthest individual from E, this individual is selected in second place. Finally, the individual C is chosen because it is far enough from both A and E. Hence, an individual from each region is selected. In fact, independently of the individual selected in the first place, the proposed approach always chooses an individual from each region.

Innovations in Parameter Control Schemes

This chapter is devoted to describing the novel parameter control schemes which are applied throughout this dissertation. In particular, a novel FLC that can be used to dynamically adapt different discrete and continuous numeric parameters belonging to different EAs is described in Section 6.1. Afterwards, the hyper-heuristic that is implemented in this thesis as an approach to parameter control is detailed in Section 6.2. So as to enable the use of this hyper-heuristic in parallel environments, its combination together with a parallel island-based model is depicted in Section 6.3. Finally, a novel hybrid control scheme that combines the use of the proposed FLC and hyper-heuristic is exposed in Section 6.4. Its main aim is to combine the benefits that both approaches provide, thus introducing a control mechanism for simultaneously adapting symbolic and numeric parameters.

6.1 Fuzzy Logic Controllers with Multiple Rule Bases

As was previously stated in Section 4.3, the feature common to most of the research described in the literature is that FLCs are applied to adapt the parameters of the crossover and mutation operators, the population size or combinations of all three. Additionally, they are designed as tailored methods for a specific EA and/or parameters, and they only make use of a unique rule base. The main novelties of the FLC proposed in this dissertation with respect to the schemes that can be found in the literature are therefore the following:

- The approach is general in that it can be used to adapt any numeric parameters of any EAs.
- The proposed system contains multiple rule bases. A rule base is enabled at a certain moment depending on historical information extracted from the optimisation process. This historical data is used to guide the adjustment of the parameter being considered.
- It is the first time that an FLC is used to control the parameters of a diversity-based MOEA, including the parameters of the novel diversity-based objectives described in Section 5.1.

The FLC is based on the general architecture depicted in Figure 4.3. In what follows, the parameter that is going to be controlled is denoted by p . The pseudocode of the proposal is shown in Algorithm 7.

First, the initialisation and learning stages—lines 1–4—are carried out. During the initialisation stage, different sample values are generated for the parameter p and distributed uniformly in its corresponding range. In order to generate them, a value Δ is considered as the difference between two consecutive samples. Although Δ might be considered as a parameter of the FLC, in this dissertation it is assigned a constant value regardless of the problem instance being solved. Then, in the learning stage, the optimisation scheme—in the particular case of this thesis, the diversity-based MOEA being controlled—is executed for *numGen* generations for each of the generated samples in order to gather sufficient information. Once these two stages are complete, the FLC infers the change to be applied over the parameter p —lines 6–13—taking into account the values of the input variables and the selected rule base. Then, the optimisation scheme is executed for *numGen* generations—line 14—with the new value of p . This process is repeated until the stopping criterion of the controlled optimisation scheme is reached.

After step 13 of Algorithm 7, a continuous value for the parameter p is obtained, meaning that in order to deal with discrete numeric parameters, this value must be defined. Equation 6.1 shows the function used to transform a continuous value into a discrete one. The random value r is sampled from a continuous uniform distribution defined in the range $[0, 1]$. Therefore, if the continuous value of p is, for instance, equal to 12.3, there is a 70% probability that the discrete value will be 12 and a 30% probability that it will be 13.

$$p = \begin{cases} \lceil p \rceil & \text{if } r \leq p - \lfloor p \rfloor \\ \lfloor p \rfloor & \text{if } r > p - \lfloor p \rfloor \end{cases} \quad (6.1)$$

Algorithm 7 Pseudocode of the novel fuzzy logic controller

- 1: **Initialisation:** Generate sample values for the parameter p distributed uniformly in its corresponding range considering a certain value Δ as the difference between two consecutive samples
 - 2: **for** (each generated sample value of the parameter p) **do**
 - 3: **Learning:** Execute *numGen* generations of the optimisation scheme with this value for the parameter p in order to gather knowledge
 - 4: **end for**
 - 5: **while** (the stopping criterion of the optimisation scheme is not satisfied) **do**
 - 6: **Transformation of the parameter p .** If the range of parameter p is different from the range $[0, 1]$, the current value of this parameter is scaled to the range $[0, 1]$ and called p'
 - 7: **Calculation of input variables.** Set the values for the input variables IMP, VAR, P-IN, BEST-P-IN
 - 8: **Selection of the rule base.** Select the most suitable rule base considering the last k decisions carried out by the FLC and the scoring function shown in Equation 6.3
 - 9: **Fuzzification.** Transform the crisp values of the input variables to fuzzy sets using the fuzzification interface
 - 10: **Mamdani's Fuzzy inference.** Apply the fuzzy logic operator AND (min), the implication method (min), and the aggregation method (max) using the selected rule base to obtain the fuzzy set of the output variable P-OUT
 - 11: **Defuzzification:** Transform the fuzzy set of the output variable P-OUT to a crisp value Δ_p using the defuzzification interface (centroid method)
 - 12: **Parameter update:** $p' = p' + \Delta_p$. The value of p' is enclosed in the range $[0, 1]$
 - 13: **Transformation of the parameter p' .** If the range of the parameter p is different from the range $[0, 1]$, the current value of p' is scaled to the range of p
 - 14: **Execution:** Execute *numGen* generations of the optimisation scheme with the new value of p
 - 15: **end while**
-

For the fuzzy inference process—lines 9–11—note that Mamdani's fuzzy inference method is used. In addition, the fuzzy logic operator AND¹ uses the minimum T-norm, the implication method uses the minimum T-norm, the aggregation method applies the maximum S-norm, and the centroid algorithm is applied as the defuzzification method. All of these components were selected because they are usually implemented together with Mamdani-type FLCs. Nevertheless, it is worth pointing out that a zero-order TSK-type FLC is also considered herein. Its main difference with respect to the Mamdani-type FLC lies in the fact that it applies the weighted average as the defuzzification method. Furthermore, it does not require the use of

¹Only the fuzzy logic operator AND is used in the antecedents of the fuzzy rules.

an aggregation method. The remaining components of the fuzzy inference process are the same as those selected for the Mamdani-type FLC.

The input variables of the FLCs—line 7—are the following:

- *IMP*. Calculated as the improvement in the original objective value of the best individual achieved by the optimisation scheme—line 14 of Algorithm 7—over the last *numGen* generations. This input variable is normalised to delimit it to the range $[0, 1]$.
- *VAR*. A measure of the diversity of the population in the decision space. The higher its value, the more diverse the population. The calculation of this input variable with no normalisation is shown in Equation 6.2. The values of the decision variable i of individuals j and k are given by $x_j[i]$ and $x_k[i]$. The total number of decision variables is represented by D and N is the population size. The value of VAR^* is normalised to enclose the variable VAR in the range $[0, 1]$.

$$VAR^* = \sum_{i=0}^{D-1} \left[\sum_{j=0}^{N-1} \left[x_j[i] - \frac{1}{N} \cdot \left(\sum_{k=0}^{N-1} x_k[i] \right) \right]^2 \right] \quad (6.2)$$

- *P-IN*. Defined as the current value of parameter p within the range $[0, 1]$.
- *BEST-P-IN*. Defined as that value of parameter p that has yielded the maximum improvement in the original objective value considering the last k values of the parameter p inferred by the FLC. Its value is also in the range $[0, 1]$.

Two different versions of the Mamdani-type and the TSK-type FLCs are defined. The first one makes use of the input variables *IMP*, *VAR*, and *P-IN*. This variant is called *FUZZY-A* for the Mamdani-type FLC and *FUZZY-A-TSK* in the case of the TSK-type FLC. The second version utilises the input variables *IMP*, *P-IN*, and *BEST-P-IN*. It is called *FUZZY-B* for the Mamdani-type FLC and *FUZZY-B-TSK* for the TSK-type FLC. For all the above variants of FLCs, only one output variable is defined, referred to as *P-OUT*, which represents the increment or decrement to be applied to the parameter p in order to change its value. The membership functions for both the input and output variables are shown in Figure 6.1. Due to the computational simplicity and efficiency advantages they offer, triangular-shaped membership functions were selected for the input and output variables. The linguistic terms represented by the membership functions—from left to right in Figure 6.1—are as follows:

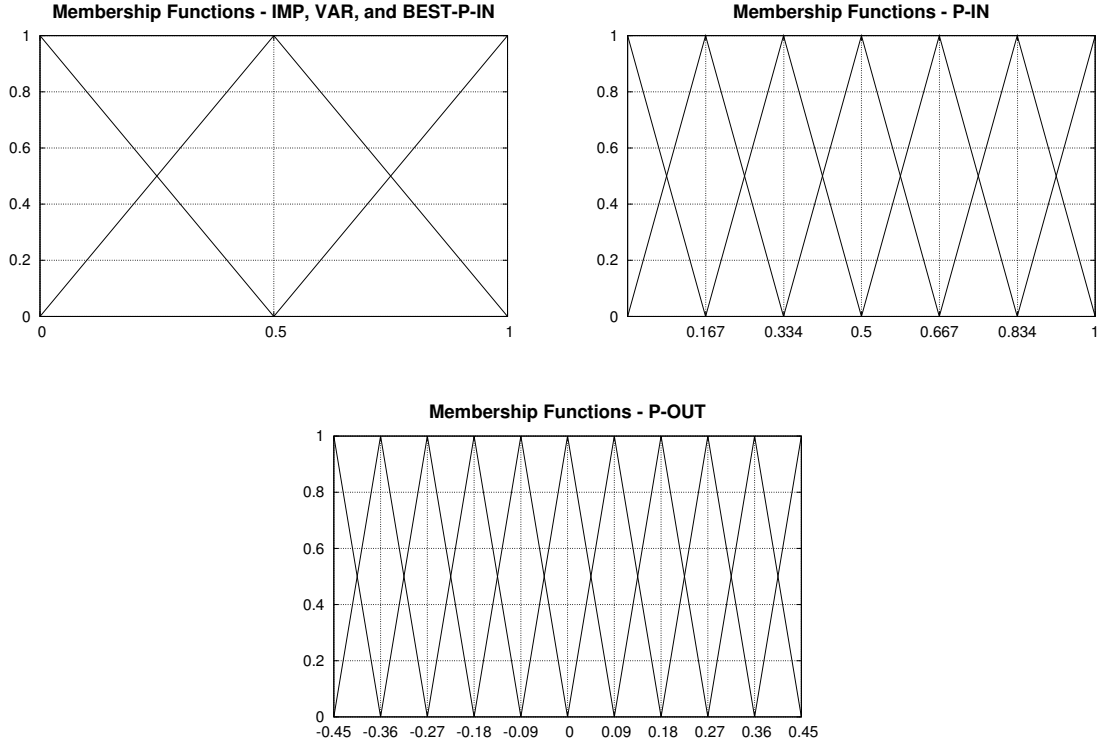


Figure 6.1: Membership functions for the input and output variables

- IMP, VAR, and BEST-P-IN: *low* (L), *medium* (M) and *high* (H).
- P-IN: *low* (L), *low-medium-b* (LMB), *low-medium-a* (LMA), *medium* (M), *medium-high-a* (MHA), *medium-high-b* (MHB), and *high* (H).
- P-OUT: *neg-giant* (NG), *neg-huge* (NU), *neg-high* (NH), *neg-medium* (NM), *neg-low* (NL), *zero* (Z), *pos-low* (PL), *pos-medium* (PM), *pos-high* (PH), *pos-huge* (PU), and *pos-giant* (PG).

In the case of the TSK-type FLCs, the linguistic terms of the output variable P-OUT are described by zero-order—constant—functions, instead of being represented by the aforementioned membership functions. These functions are *neg-giant* = -0.45, *neg-huge* = -0.36, *neg-high* = -0.27, *neg-medium* = -0.18, *neg-low* = -0.09, *zero* = 0, *pos-low* = 0.09, *pos-medium* = 0.18, *pos-high* = 0.27, *pos-huge* = 0.36, and *pos-giant* = 0.45.

For each FLC different rule bases are defined. The reason for using different rule bases is that different fuzzy rules will be applicable depending on the behaviour exhibited during the previous execution. For instance, if the best results were historically obtained by low values of the parameter p , the fuzzy rules should promote the use of such low values. Every rule base is composed of different fuzzy rules. The left-hand side of Table 6.1 shows one of the rule bases defined for the FUZZY-A and FUZZY-A-TSK approaches, while the right-hand side shows another one for the FUZZY-B and FUZZY-B-TSK schemes. Only the fuzzy logic operator AND is used in the antecedents of these fuzzy rules. Generally, every fuzzy rule considers three input and one output variables. In those cases where a ‘-’ is shown, the corresponding fuzzy rule has no dependency on the corresponding variable. The remaining rule bases are not shown due to space constraints but are similar to those shown here ².

In order to select the most suitable set of rules, in this dissertation a novel scoring function is proposed. It relies on a weighted average that considers historical data for both the improvement in the original objective value and the degrees of membership of parameter p to each term defined for the input variable P-IN.

The value of k is defined as the amount of historical knowledge considered by the FLC, i.e. information on the last k decisions made by the FLC is taken into account. In contrast, d is the total number of decisions that the FLC has carried out, and $numTerms$ is the number of linguistic terms defined for the input variable P-IN. The score assigned to each linguistic term $i \in [0, numTerms - 1]$ is given by Equation 6.3. The improvement achieved during execution $d - j$ of the optimisation scheme—line 14 of Algorithm 7—is given by $\gamma[d - j]$. In addition, the degree of membership of parameter p to the linguistic term i during execution $d - j$ is represented by $\delta[i][d - j]$. Thus, the linguistic term i will be assigned a higher score if the values of parameter p have larger degrees of membership to said linguistic term, and if, at the same time, the values of parameter p are able to achieve higher improvements in the original objective value. Finally, we note that the scoring function assigns more importance to the last decisions inferred by the FLC. Thus, for each linguistic term the equation represents a weighted average of its improvement, where greater importance is given to the last executions in which values of the controlled parameter have a high degree of membership to the corresponding linguistic term.

²The complete specifications for all of the rule bases are available in Appendix B.

6.1. Fuzzy Logic Controllers with Multiple Rule Bases

Table 6.1: Rule bases designed for FUZZY-A and FUZZY-A-TSK (left-hand side); and for FUZZY-B and FUZZY-B-TSK (right-hand side)

Rules					Rules				
		Inputs					Inputs		
ID	P-IN	IMP	VAR	P-OUT	ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PG	1	L	L	L	NL
2	L	M	-	PL	2	L	L	M	PL
3	L	H	-	Z	3	L	L	H	PL
4	LMB	L	-	PG	4	L	M	-	Z
5	LMB	M	-	PL	5	L	H	-	Z
6	LMB	H	-	Z	6	LMB	L	-	NM
7	LMA	L	-	PG	7	LMB	M	-	NL
8	LMA	M	-	PL	8	LMB	H	-	Z
9	LMA	H	-	Z	9	LMA	L	-	NH
10	M	L	-	PU	10	LMA	M	-	NL
11	M	M	-	PL	11	LMA	H	-	Z
12	M	H	-	Z	12	M	L	-	NU
13	MHA	L	-	PH	13	M	M	-	NL
14	MHA	M	-	PL	14	M	H	-	Z
15	MHA	H	-	Z	15	MHA	L	-	NG
16	MHB	L	-	PM	16	MHA	M	-	NL
17	MHB	M	-	PL	17	MHA	H	-	Z
18	MHB	H	-	Z	18	MHB	L	-	NG
19	H	L	L	PL	19	MHB	M	-	NL
20	H	L	M	PL	20	MHB	H	-	Z
21	H	L	H	NL	21	H	L	-	NG
22	H	M	-	Z	22	H	M	-	NL
23	H	H	-	Z	23	H	H	-	Z

$$score[i] = \frac{\sum_{j=1}^{min(k,d)} \gamma[d-j] \cdot \delta[i][d-j] \cdot (min(k,d) - j + 1)}{\sum_{j=1}^{min(k,d)} \delta[i][d-j] \cdot (min(k,d) - j + 1)} \quad (6.3)$$

Note that if *numTerms* linguistic terms are defined for the variable P-IN, *numTerms* rule bases have to be implemented such that the FLC works with the proposed scoring function. Figure 6.1 shows that seven linguistic terms are defined for the input variable P-IN, thus seven different rule bases are implemented. Different numbers

of fuzzy rule bases were tested and it was found that the higher the number of rule bases, the smoother the variations in the parameter p inferred by the FLC, and thus the steadier the FLC. However, when considering more than seven fuzzy rule bases, the performance started to degrade somewhat, as was also the case with a lower number of fuzzy rule bases. Thus, seven rule bases were selected as this yielded the best performance for the FLC. This fact also justifies the use of seven linguistic terms for the input variable P-IN, instead of using three linguistic terms as in the case of the remaining input variables. For the remaining input variables, three linguistic terms are used so as to maintain the rule bases as simple as possible.

Once the scores are calculated, the linguistic term with the maximum score is selected. This means that those values of parameter p with a large enough degree of membership to this linguistic term should provide better performance than the other values. Therefore, if the linguistic term i is selected as the most appropriate one, rule base i is enabled. This selected rule base is responsible for adapting the value of parameter p so that it approaches the values represented by term i . For instance, assume that the current value of parameter p is 0.01 and the most suitable rule base—considering the scoring function—is the one represented by the linguistic term *high* of the input variable P-IN. This means that historically high values of parameter p have yielded good improvements in the original objective value. Thus, the rule base to be applied in this case is precisely the one shown in the left-hand side of Table 6.1, considering, for instance, the FUZZY-A approach. If a fuzzy set for the variable IMP, which has a large degree of membership to the term *low*, since P-IN—with value 0.01—is represented by a fuzzy set with a large degree of membership to the term *low*, then the output fuzzy set—the one corresponding to the output variable P-OUT—will have a large degree of membership to the linguistic term *pos-giant*. Consequently, the value of parameter p will be considerably increased so that it will tend towards higher values.

Finally, it is worth mentioning that although in this dissertation the parameters of diversity-based MOEAs are adapted, only the improvement in the original objective is considered by the variable IMP and by Equation 6.3 to measure the performance of the optimisation scheme at all times, thus discarding the information given by the diversity-based objective. For this reason, indicators that measure the performance of multi-objective optimisers are not required. Nevertheless, if the improvement were measured by some multi-objective performance metric rather than by using the original objective, the FLCs proposed herein might be applied to control the parameters of “*pure*” multi-objective approaches.

6.2 Hyper-heuristics

An extension of the hyper-heuristic first described in [339], which is based on the usage of choice functions and a probabilistic selection scheme, is implemented in this thesis as an approach to parameter control. Recall that in choice functions, a scoring method assigns a score to each low-level method that typically represents its historical performance. As a result, more resources are granted to the low-level approach that maximises this scoring function.

Traditionally, choice functions have been applied to select from among a set of candidate neighbourhood definitions. Moreover, they usually make use of some kind of memory mechanism. For instance, considering a tabu memory mechanism, the low-level methods that have not been able to improve upon a certain solution will not be applied anymore until the whole optimisation scheme escapes from the solution. However, it was discovered that for many problems, the low-level approaches generally improve on the initial solutions, independently of their performance. Hence, it makes no sense to use a tabu memory mechanism that avoids the application of methods with a poor past performance. This is why the hyper-heuristic applied herein as a parameter control technique is based on the idea of choice functions, but it attempts to mitigate the above drawback, which stems from using certain memory mechanisms. Finally, it is worth pointing out that instead of selecting the most suitable neighbourhood definition from among a set of candidates, this hyper-heuristic is used to adapt the parameters belonging to EAs.

The hyper-heuristic implemented in this dissertation, which is called HH-PROB, is based on using a scoring strategy and a probabilistic selection strategy for choosing the most appropriate low-level configuration for the algorithm to be executed. Once a low-level configuration is selected, only that configuration is executed until a local stopping criterion is achieved. When this happens, another low-level configuration is selected and executed, and the final population of the last low-level configuration used becomes the initial population of the new low-level configuration. This process continues until a global stopping criterion is satisfied. At the beginning of an execution, every low-level configuration is run one time. The order in which the different low-level configurations are executed during this initialisation stage is randomly determined. Once the initial stage is finished, the low-level configuration that must be executed is selected as follows.

First, the scoring strategy assigns a score to each low-level configuration. This score estimates the improvement that each low-level configuration can achieve starting from the current solution set. Thus, larger values are assigned to more promising

schemes in light of their historical behaviour. In order to calculate this estimate, the previous improvements achieved by each low-level configuration are used. Since in this thesis diversity-based MOEAs are controlled by this hyper-heuristic, the score of a certain low-level configuration is computed through the improvements achieved in the original objective, thus discarding the information given by the diversity-based objective. As a result, the improvement γ is defined as the difference, in terms of the original objective value, between the best resulting individual and the best initial individual. Considering a configuration $conf$ that has been executed j times, the score $s(conf)$ is calculated as the weighted average of the last k improvements, as Equation 6.4 shows. Note that if the improvement γ was given by a multi-objective performance metric, instead of taking into account the original objective value, this hyper-heuristic might be used to control the parameters of “pure” multi-objective schemes. In fact, this idea is addressed in [296].

$$s(conf) = \frac{\sum_{i=1}^{\min(k,j)} (\min(k,j) + 1 - i) \cdot \gamma[conf][j - i]}{\sum_{i=1}^{\min(k,j)} i} \quad (6.4)$$

In Equation 6.4, $\gamma[conf][j - i]$ represents the improvement achieved by configuration $conf$ in execution number $j - i$. The adaptation level of the hyper-heuristic, i.e. the total amount of historical knowledge that it considers in order to perform its decisions, can be varied depending on the value of k . Finally, note that the weighted average assigns a greater importance to the most recent executions.

The score $s(conf)$ is used to calculate the probability of selecting a particular low-level configuration. However, the stochastic behaviour of the low-level EAs involved may lead to variations in the results they yield. Therefore, the probability calculation also enables a fraction of selections based on a random scheme and is implemented as follows. Specifically, the hyper-heuristic can be tuned by means of the parameter β , which represents the minimum selection probability that should be assigned to a low-level configuration. If n_h is the number of low-level configurations involved, a random selection based on a uniform distribution is performed in $\beta \cdot n_h$ percent of the cases. Therefore, the probability of selecting each configuration $conf$ is defined as shown in Equation 6.5.

$$prob(conf) = \beta + (1 - \beta \cdot n_h) \cdot \left[\frac{s(conf)}{\sum_{i=1}^{n_h} s(i)} \right] \quad (6.5)$$

Another variant of the above hyper-heuristic is also implemented in this dissertation. As in the case of the previous version, the scoring function is given by Equation 6.4. However, instead of resorting to a probabilistic selection strategy, an elitist selection strategy is used. In this case, the low-level configuration with the maximum score $s(conf)$ is always selected, in addition to the random selections performed following a uniform distribution in $\beta \cdot n_h$ percent of the cases. This variant is named HH-ELI.

6.3 Dynamic-mapped Island-based Model

The hyper-heuristic presented in the previous section is combined together with a parallel island-based model so as to enable its usage in parallel environments. This hybrid approach is known as *Dynamic-mapped Island-based Model (DYN)*. The architecture of the dynamic-mapped model is similar to the island-based model detailed in Section 2.6.1. The population is decomposed into a number of independent and separate sub-populations, each belonging to a *worker island*, in which the corresponding sub-population is evolved in isolation by applying a certain low-level configuration of an EA. Furthermore, as in the island-based model, a tuneable migration stage allows for the occasional exchange of individuals among neighbouring islands. In this dissertation, different migration stages are tested with the DYN model.

In the standard island-based model there exists a static mapping between the islands and configurations, i.e. each island executes the same configuration over the course of the complete run. In a homogeneous island-based model, there is only one configuration that is executed by every worker island. In a heterogeneous island-based model, the configurations executed by worker islands are different. However, in the model in question, a dynamic mapping among the islands and configurations is applied. Thus, the configurations executed in each island over the course of the run can vary. This dynamic mapping is performed using the hyper-heuristic exposed in the last section. So as to manage the dynamic mapping, i.e. to apply the hyper-heuristic, a new special island, called the *master island*, is introduced into the

scheme. In order to implement it, two kinds of stopping criteria are defined. First, a local stopping criterion is fixed for the execution of the configurations on the worker islands. When a local stopping criterion is reached, the island's execution is stopped. The local results are then sent to the master island. At this point, the master island applies the hyper-heuristic in order to decide which low-level configuration is going to be applied in the idle island. This configuration is applied by taking as the initial population the final population obtained by the previous configuration. The above process is repeated until a global stopping criterion is satisfied. When this global stopping criterion is reached, every worker island sends its local solution to the master and the run ends.

An additional modification is introduced into the DYN model with the aim of adapting the hyper-heuristic to parallel environments. In island-based models, the migration stage might completely change the quality of a given sub-population. Since the hyper-heuristic evaluates the performance of the configurations by considering the members of their sub-populations, it must take into account the effect caused by the immigrants. Particularly, the improvements obtained during the migration stage are calculated and subtracted from the overall improvement that has been obtained. In this way, the migration stage is kept, but its effect on the decisions performed by the hyper-heuristic is minimised. Finally, it is important to remark that the same local stopping criterion is fixed for every worker island. This facilitates the operation of the hyper-heuristic because it does not need to consider the time invested by each low-level configuration to carry out its decisions. In addition, the hyper-heuristic completely ignores the features of the underlying architecture. When considering a homogeneous architecture, this fact does not present any problems. For heterogeneous architectures, however, the resources might not be properly assigned.

6.4 Hybrid Control Scheme based on Fuzzy Logic Controllers and Hyper-heuristics

In this section, a novel hybrid control approach based on FLCs and hyper-heuristics is proposed. It combines the FLC presented in Section 6.1 with the hyper-heuristic detailed in Section 6.2. Recall that the main advantage of this hyper-heuristic is that it is able to control symbolic and numeric parameters. However, since the size of the set of low-level candidate configurations is generally fixed and finite, in the case of controlling numeric parameters, the number of possible values that can be

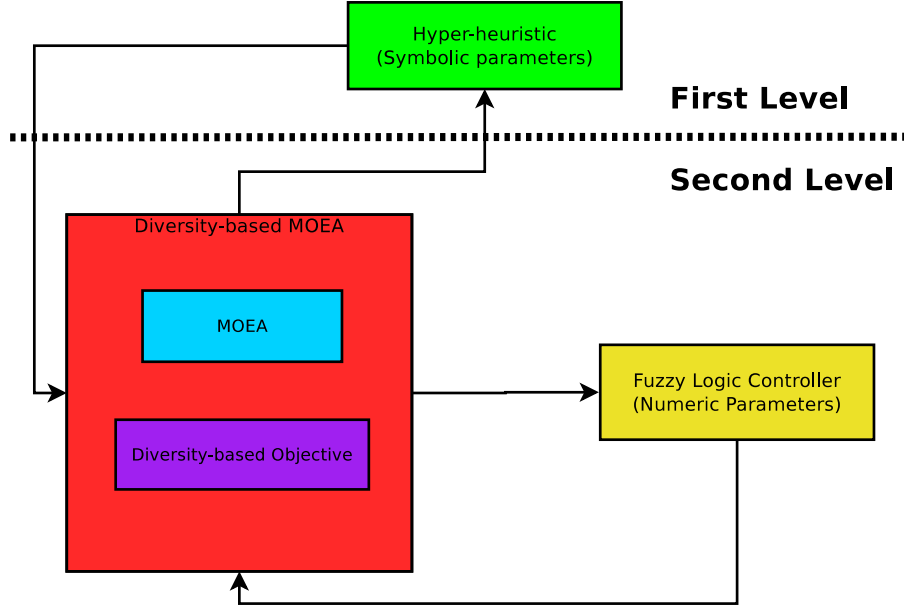


Figure 6.2: Multi-level architecture of the novel hybrid parameter control scheme

assigned to these numeric parameters is therefore also finite. In contrast, the main benefit of using the proposed FLC as a control scheme is that the possible values that can be assigned to a certain parameter are not selected from a finite set. Its main drawback lies in the fact that it cannot be directly applied to control symbolic parameters. In order to avoid this drawback, and to profit from the strong points of the two aforementioned approaches, a completely new hybrid control scheme is proposed in this dissertation that provides a parameter control mechanism that is able to simultaneously adapt symbolic and numeric parameters.

Figure 6.2 shows the multi-level architecture of this hybrid control scheme when it is applied to a diversity-based MOEA. In the first level, the hyper-heuristic described in Section 6.2 is used to control the symbolic parameters. To do so, it selects the most promising low-level configuration from among a set of candidates, depending on their past performance. A low-level configuration in this case refers to an instance of a diversity-based MOEA—other EAs might be controlled by means of this scheme—with a particular setting for the symbolic parameters that are controlled, such as the variation operators. All of the algorithm’s other parameters remain constant, except for the numeric parameters, which are adapted by the second-level FLC. Once a low-level configuration is selected, only that configuration is executed until the local stopping criterion established by the hyper-heuristic is satisfied. When

this happens, another low-level configuration, which could be the same as the last execution, is selected and executed. This process is repeated until a global stopping criterion is reached.

In the second level, the FLC introduced in Section 6.1 is used to adapt numeric parameters. The FLC carries out its decisions by considering historical information on the values of the parameters inferred in past executions. It is important to recall that, in this level, the low-level configuration is executed until the local stopping criterion established by the hyper-heuristic is achieved. However, the FLC also infers changes over the parameters periodically, so another local stopping criterion is established by the FLC. In order to clarify this last fact, consider for instance a global stopping criterion equal to $2.5 \cdot 10^6$ function evaluations, and $2.5 \cdot 10^4$ function evaluations for the local stopping criterion established by the hyper-heuristic. This means that the hyper-heuristic is able to carry out $1 \cdot 10^2$ decisions during the whole optimisation process, changing the values for the symbolic parameters, and that every selected low-level configuration is executed for $2.5 \cdot 10^4$ function evaluations. If the local stopping criterion established by the FLC is equal to $5 \cdot 10^2$ function evaluations, then the FLC infers 50 changes over the numeric parameters during every execution of a low-level configuration.

Part III

Validation of the Algorithmic Proposals: Benchmark Problems and Complex Applications

Benchmark Problems

The goal of this chapter is to validate the different algorithmic proposals introduced throughout this dissertation by conducting a comprehensive experimental evaluation involving a set of single-objective benchmark problems. The main benefit of using benchmark functions is that the main features and properties of these problems, such as their global optimum, are usually known. Hence, the performance of a certain optimisation scheme can be directly measured. Additionally, the time used to evaluate an individual that represents a solution to a benchmark problem is significantly shorter than the time needed to evaluate an individual that represents a solution belonging to a real-world problem. As a result, by considering benchmark problems, a vast number of computational tests can be performed. One of the main drawbacks, however, is that the number of problems available might not be large enough, which could hamper the extraction of general conclusions.

There have been several attempts to define test suites or toolkits for building test suites. For instance, the *DeJong test suite* [88], which consists of five benchmark functions, was very popular for several years. However, some more novel test functions that incorporate additional features have since been defined. These include the *OneMax problem*, the *Sphere Model*, the *Schwefel function*, and the *Generalised Rastrigin function* [104]. Another well-known compendium of benchmark functions is that provided in the *Black-Box Optimization Benchmarking (BBOB)* test suite ¹. It was proposed and has been used in recent years in a competition session organised at the GECCO conference. In the particular case of this dissertation, a set of 19 benchmark functions that was recently proposed in a special issue of the *Soft Computing* journal [220] is considered. One of the main advantages of this test suite

¹<http://coco.gforge.inria.fr/doku.php?id=bbob-2013>

Table 7.1: Definition of the F1–F11 benchmark functions

Function	Name	Range	Optimum
F1	Shif. Sphere	$[-100, 100]^D$	-450
F2	Shif. Schwefel 2.21	$[-100, 100]^D$	-450
F3	Shif. Rosenbrock	$[-100, 100]^D$	390
F4	Shif. Rastrigin	$[-5, 5]^D$	-330
F5	Shif. Griewank	$[-600, 600]^D$	-180
F6	Shif. Ackley	$[-32, 32]^D$	-140
F7	Shif. Schwefel 2.22	$[-10, 10]^D$	0
F8	Shif. Schwefel 1.2	$[-65.536, 65.536]^D$	0
F9	Shif. Extended f_{10}	$[-100, 100]^D$	0
F10	Shif. Bohachevsky	$[-15, 15]^D$	0
F11	Shif. Schaffer	$[-100, 100]^D$	0

is that all its functions are scalable, i.e. the user can define the number of decision variables for each. These problems are called the *F1–F19* benchmark functions.

The rest of this chapter is organised as follows. In Section 7.1 the mathematical formulation of the F1–F19 benchmark problems is described. Then, the different optimisation schemes defined for dealing with this set of problems are presented in Section 7.2. Afterwards, the control approaches that are proposed to adapt the parameters of said optimisation schemes are introduced in Section 7.3. Finally, Section 7.4 details the experimental evaluation conducted using the proposed optimisation schemes and parameter control approaches on the F1–F19 functions.

7.1 Formal Definition

The F1–F19 benchmark functions are scalable continuous single-objective optimisation problems that combine different properties involving *modality*—a function is *multi-modal* if it has multiple local optima, and is *uni-modal* if it has a single global optimum—*separability*—a function is separable if it can be expressed as a product of sub-functions, each depending on a single decision variable—and the ease of dimension-by-dimension optimisation—whether or not the function can be optimised by independently adjusting each decision variable.

A summary of the main features of the F1–F11 functions is shown in Table 7.1, while its properties are shown in Table 7.2. In this thesis, the value for the parameter D —the number of decision variables in the problems—is generally fixed to 500.

Table 7.2: Properties of the F1–F11 benchmark functions

Function	(U)nimodal/ (M)ultimodal	Shifted	Separable	Easily optimised dimension by dimension
F1	U	Y	Y	Y
F2	U	Y	N	N
F3	M	Y	N	Y
F4	M	Y	Y	Y
F5	M	Y	N	N
F6	M	Y	Y	Y
F7	U	Y	Y	Y
F8	U	Y	N	N
F9	U	Y	N	Y
F10	U	Y	N	N
F11	U	Y	N	Y

Table 7.3: Definition of the F12–F19 hybrid composition benchmark functions

Function	F_{ns}	F'	m_{ns}	Range	Optimum
F12	NS-F9	F1	0.25	$[-100, 100]^D$	0
F13	NS-F9	F3	0.25	$[-100, 100]^D$	0
F14	NS-F9	F4	0.25	$[-5, 5]^D$	0
F15	NS-F10	NS-F7	0.25	$[-10, 10]^D$	0
F16	NS-F9	F1	0.5	$[-100, 100]^D$	0
F17	NS-F9	F3	0.75	$[-100, 100]^D$	0
F18	NS-F9	F4	0.75	$[-5, 5]^D$	0
F19	NS-F10	NS-F7	0.75	$[-10, 10]^D$	0

Table 7.3 shows the definition of the F12–F19 benchmark problems. These are built by combining a non-separable function F_{ns} with another function F' . It is important to remark that F' can also be a non-separable function. The following functions were considered when building the F12–F19 problems:

- **Non-Separable Functions:**
 - F3: Shifted Rosenbrock
 - NS-F9: Non-shifted Extended f_{10}
 - NS-F10: Non-shifted Bohachevsky
- **Other Component Functions:**

- F1: Shifted Sphere
- F4: Shifted Rastrigin
- NS-F7: Non-shifted Schwefel 2.22

The following steps are required to obtain a hybrid composition function $F_{ns} \oplus F'$: divide the solution into two parts, evaluate each part with a different function, and finally, combine the two results. A parameter m_{ns} is used to specify the rate at which variables are evaluated by F_{ns} . If a higher value of m_{ns} is used, the hybrid function becomes more difficult to optimise dimension by dimension.

Finally, we should note that some of the experiments were performed using only the F1–F11 functions. For other cases, however, all 19 benchmark problems were taken into account.

7.2 Optimisation Schemes

This section is devoted to describing the optimisation schemes that are applied herein to deal with the F1–F19 benchmark functions. Specifically, the optimisation methods considered are different variants of a single-objective GA, as well as separate versions of a diversity-based MOEA based on the well-known NSGA-II.

7.2.1 Single-objective Genetic Algorithms

In this chapter, three different variants of the single-objective GA proposed in Section 2.3.1 are considered for solving the benchmark functions. Each of them is based on one of the three survivor selection strategies exposed in Section 2.2.4, i.e. the SS-S, the GEN-S, and the RW-S survivor selection mechanisms. For these three variants of the GA, individuals were encoded through a vector of D real values, with D being the number of decision variables in the benchmark problems in question. The mutation operator used was the UM and the crossover operator was the SBX, the operations of which were described in Sections 2.2.3 and 2.2.2, respectively.

7.2.2 Diversity-based Multi-objective Evolutionary Algorithms

The novel diversity-based MOEA, which is used here to tackle the benchmark problems, is based on the NSGA-II—Section 2.4.1. Since a diversity-based MOEA is applied, an additional diversity-based objective function must be considered together with the objective function of every benchmark in question. Different genotypic diversity-based objectives were taken into account. In particular, the diversity-based objectives tested were ADI, DBI, and DCN—Section 3.1. Moreover, the diversity-based objective with parameters DCN-THR—Section 5.1—was also applied. Finally, it is also important to remark that some experiments were carried out with the novel DCN-REF diversity-based survivor selection scheme proposed in Section 5.2.

Finally, so as to completely define this diversity-based MOEA, individuals were represented by means of a vector of D real numbers, where D is the number of decision variables of the benchmark problems at hand. The mutation and crossover operators used were also the UM and the SBX, respectively.

7.3 Parameter Control Schemes

This section focuses on introducing the different parameter control approaches that are used herein to adapt some of the parameters contained in the optimisation schemes detailed in preceding sections. First, FLCs and hyper-heuristics are used as external parameter control approaches to adapt some of the parameters belonging to the diversity-based MOEA described in Section 7.2.2. Particularly, the threshold ratio th of the DCN-THR diversity-based objective, which was introduced in Section 5.1, is dynamically adjusted during the course of a run. Furthermore, a novel proposal based on self-adaptation, which was specifically designed to deal with said parameter, is used as the comparison approach. Finally, a novel hybrid control scheme based on FLCs and hyper-heuristics is also applied to simultaneously adjust different symbolic and numeric parameters belonging to the diversity-based MOEA. Said parameters are the crossover and mutation operators, as well as the mutation rate p_m .

7.3.1 Fuzzy Logic Controllers and Hyper-heuristics

The novel FLCs proposed in Section 6.1, as well as the hyper-heuristics described in Section 6.2, are used herein to control the parameter th of the diversity-based objective DCN-THR. The main novelty of the FLCs proposed lies in the incorporation of a set of different rule bases that are enabled depending on historical information extracted from the optimisation process. Recall that two different variants of the Mamdani-type and TSK-type FLCs were defined. Particularly, the FUZZY-A and FUZZY-B Mamdani-type FLCs, as well as the FUZZY-A-TSK and FUZZY-B-TSK TSK-type FLCs, were considered when carrying out the experimental evaluation. However, the differences between the Mamdani-type and the TSK-type FLCs were not statistically significant. That is the why only data involving the FUZZY-A and FUZZY-B Mamdani-type FLCs are shown in Section 7.4. Hence, the conclusions presented in that section for the Mamdani-type FLCs are also valid for the TSK-type FLCs. On the other hand, the HH-PROB and HH-ELI hyper-heuristics approaches are used for comparison purposes. It is important to remark that this is the first time that parameter control techniques based on FLCs and hyper-heuristics have been applied to a diversity-based MOEA in which the parameters of the diversity-based objective are adapted.

One of the main drawbacks of the threshold ratio th is that the most suitable values might depend on the problem being solved or even on the current stage of the optimisation process, meaning that modifying it during the execution could be beneficial. Consequently, the application of parameter control techniques to automatically adapt this parameter ought to significantly improve both the behaviour and the robustness of the diversity-based MOEAs proposed. This idea seems to be very promising and is addressed in detail in this thesis.

7.3.2 Self-adaptation

In order to enable the parameter th of the DCN-THR diversity-based objective to undergo self-adaptation, it is encoded within the chromosome of individuals and is thus subjected to mutation and crossover operators that change its value during the optimisation process. This relies on the premise that better values of the parameter th will produce better individuals, which in turn will have more opportunities to survive, and consequently propagate better parameter values. This is “*the idea of the evolution of evolution*” [106]. In the case of self-adaptive approaches to parameter control, the selection and variation operators of the EA are responsible for changes in the parameter values, i.e. the updating mechanism that adapts the parameters

$x[0]$	$x[1]$	$x[2]$	\dots	$x[D-2]$	$x[D-1]$	th
--------	--------	--------	---------	----------	----------	------

Figure 7.1: Chromosome representing an individual for the self-adaptive schemes

is implicit. This differentiates the method from the previously described FLCs and hyper-heuristics, in which the adaptation mechanism is external to the EA used.

Figure 7.1 shows the chromosome of an individual that considers the self-adaptation of the threshold ratio th . For an individual representing a solution to a problem with D decision variables, the values $x[i], i \in [0, D - 1]$ represent these decision variables.

Three completely novel versions of the self-adaptive approach are proposed herein:

- *SELF-A*. The value of the parameter th belonging to the best individual in the population—the one with the lowest original objective value—is applied to each generation to calculate the diversity-based objective of every individual.
- *SELF-B*. The mean value of the parameter th considering all individuals in the population is used at each generation to calculate the diversity-based objective value of every individual.
- *SELF-C*. The corresponding encoded value of the parameter th is applied to each individual at each generation in order to calculate its own diversity-based objective value.

With respect to the classifications proposed in [9, 315], SELF-A and SELF-B act at the population level, while the SELF-C approach acts at an individual level. Thus, the encoding of parameter th into the chromosome can be interpreted differently, supplying different algorithm variants in which the *scope* of the parameter varies.

7.3.3 Hybrid Control Scheme based on Fuzzy Logic Controllers and Hyper-heuristics

The novel hybrid control scheme proposed in Section 6.4, which is based on FLCs and hyper-heuristics, is applied herein so as to adapt the crossover and mutation operators, as well as the mutation rate p_m of the diversity-based MOEA described in Section 7.2.2. Recall that the hyper-heuristic controls the symbolic parameters, while the FLC adapts the numeric parameters. Hence, the hyper-heuristic is responsible for controlling the crossover and mutation operators, whereas the FLC

is responsible for adapting the mutation rate p_m . Specifically, the approaches considered to constitute the hybrid control scheme were the FUZZY-A FLC and the HH-PROB hyper-heuristic. Lastly, we should note that this is the first time that FLCs and hyper-heuristics are combined into a hybrid parameter control scheme.

7.4 Experimental Evaluation and Discussion

In this section the experiments performed on the F1–F19 functions using the aforementioned optimisation schemes and parameter control approaches are presented.

Experimental Method. The different optimisation schemes, as well as the parameter control approaches, were implemented using METCO. The tests were run on a Debian GNU/Linux computer with four 1.7 GHz AMD ® Opteron™ processors (model number 6164HE) and 64 Gb RAM. The compiler was the GCC 4.7.2, while the FLCs were implemented using the *fuzzylite* 3.1 library [281]. Since every experiment applied stochastic algorithms, every execution was repeated 32 times. As a result, comparisons were performed by applying the statistical analysis detailed in Section 1.2.6.

7.4.1 Performance of the Diversity-based Multi-objective Evolutionary Algorithm

The goal of the first experiment is to measure the performance of the proposed diversity-based MOEA. Particularly, it would be interesting to discover if the use of such a diversity-based MOEA offers any benefits over using a single-objective GA. To do so, a comparison between the diversity-based MOEA described in Section 7.2.2 and the single-objective GA detailed in Section 7.2.1 was carried out. The following configurations for both optimisation schemes were considered:

- 9 configurations of the diversity-based MOEA, which were created by combining three separate diversity-based objective functions and three values for the population size.
- 9 configurations of the single-objective GA, which were defined by the use of three different survivor selection mechanisms and three values for the population size.

All the above configurations for the two optimisation schemes were applied to solve the F1–F11 benchmark functions with $D = 500$ decision variables. The values for the parameters of the diversity-based MOEA were set as follows:

- Population size N fixed to 5, 10, and 20 individuals.
- Diversity-based objectives ADI, DBI, and DCN.
- The UM operator was applied with $p_m = \frac{1}{D} = 0.002$.
- The SBX operator was applied with $p_c = 1$.
- Stopping criterion fixed to a total number of $5000 \cdot D = 2.5 \cdot 10^6$ function evaluations.

In the case of the single-objective GA, its parameterisation was the following:

- Population size N fixed to 5, 10, and 20 individuals.
- Survivor selection mechanisms SS-S, GEN-S, and RW-S.
- The UM operator was applied with $p_m = \frac{1}{D} = 0.002$.
- The SBX operator was applied with $p_c = 1$.
- Stopping criterion fixed to a total number of $5000 \cdot D = 2.5 \cdot 10^6$ function evaluations.

Table 7.4 shows, for each population size and benchmark function considered, the best configurations of the single-objective GA and the diversity-based MOEA. Comparisons were made in terms of the median of the original objective value achieved by every configuration at the end of the executions. Note that the superiority of the GEN-S approach in the case of the single-objective GA, and of the DCN scheme for the case of the diversity-based MOEA, is clear for most of the problems. In a small number of test cases, however, they were not able to provide the best results.

Table 7.5 shows the median of the error attained by the best configurations of the single-objective GA and the diversity-based MOEA for each population size and benchmark function considered. The error was defined as the difference between the original objective value and the global optimum of the benchmark function at hand—Table 7.1. It was calculated assuming an accuracy equal to $1 \cdot 10^{-6}$. In addition, a statistical comparison between the best configurations of the single-objective GA and the diversity-based MOEA was carried out for each population size and test case. For cases where differences were statistically significant, the data for the best

Table 7.4: Best configurations for the single-objective and the diversity-based approaches

	F1	F2	F3	F4	F5	F6
Single_5	GEN-S	RW-S	GEN-S	GEN-S	GEN-S	GEN-S
Single_10	GEN-S	RW-S	GEN-S	GEN-S	GEN-S	GEN-S
Single_20	GEN-S	RW-S	GEN-S	GEN-S	SS-S	GEN-S
Multi_5	DCN	ADI	DCN	DCN	DCN	DCN
Multi_10	DCN	DCN	DCN	DCN	DCN	DCN
Multi_20	DCN	DCN	DCN	DCN	DCN	DCN

	F7	F8	F9	F10	F11
Single_5	GEN-S	GEN-S	GEN-S	GEN-S	GEN-S
Single_10	GEN-S	GEN-S	GEN-S	GEN-S	GEN-S
Single_20	GEN-S	GEN-S	GEN-S	GEN-S	GEN-S
Multi_5	DCN	DCN	DCN	DCN	DCN
Multi_10	DCN	DCN	DCN	DCN	DCN
Multi_20	DCN	DCN	DCN	DCN	DCN

of the two approaches are shown in bold ². Note that, in general, the optimisation schemes configured with a lower population size obtained a lower median for the error. Furthermore, it can be observed that the diversity-based MOEA provided statistically better results than the single-objective GA in a significant amount of test cases. In other cases, however, the best configuration of the single-objective GA statistically outperformed the best configuration of the diversity-based MOEA. From a total number of 33 statistical tests, in 19 of them the best configuration of the diversity-based MOEA was statistically better than the best configuration of the single-objective GA. In contrast, the best configuration of the single-objective GA was statistically superior in 6 statistical tests. Finally, 8 statistical tests did not show statistically significant differences between both strategies.

It is important to know which kinds of problems are better suited to be solved by the diversity-based MOEA. We should note that it provided more benefits when it was applied to a uni-modal problem. For instance, for a population size equal to 5 individuals, the best configuration of the diversity-based MOEA was statistically better than the best configuration of the single-objective GA for all the uni-modal benchmark problems, i.e. F1, F2, and F7–F11. However, when multi-modal prob-

²Throughout this dissertation scheme A is said to be statistically better than scheme B if the differences between them are statistically significant, and if the mean and median obtained by A are better—one of the metrics might be equal—than the mean and median obtained by B.

Table 7.5: Median of the error achieved by the best configurations of both optimisation schemes

	F1	F2	F3	F4	F5	F6
Single_5	$3.00 \cdot 10^{-3}$	$1.67 \cdot 10^1$	$1.32 \cdot 10^3$	$3.00 \cdot 10^{-3}$	$8.00 \cdot 10^{-3}$	$3.00 \cdot 10^{-3}$
Multi_5	$< 1 \cdot 10^{-6}$	$1.14 \cdot 10^1$	$1.26 \cdot 10^3$	$3.00 \cdot 10^{-3}$	$< 1 \cdot 10^{-6}$	$1.00 \cdot 10^{-3}$
Single_10	$2.00 \cdot 10^{-3}$	$1.41 \cdot 10^1$	$1.47 \cdot 10^3$	$3.95 \cdot 10^{-2}$	$< 1 \cdot 10^{-6}$	$3.00 \cdot 10^{-3}$
Multi_10	$5.00 \cdot 10^{-3}$	$1.81 \cdot 10^1$	$1.48 \cdot 10^3$	$1.00 \cdot 10^0$	$1.00 \cdot 10^{-3}$	$4.00 \cdot 10^{-3}$
Single_20	$6.50 \cdot 10^{-2}$	$1.54 \cdot 10^1$	$1.78 \cdot 10^3$	$9.28 \cdot 10^{-1}$	$8.00 \cdot 10^{-3}$	$1.70 \cdot 10^{-2}$
Multi_20	$3.70 \cdot 10^{-2}$	$2.23 \cdot 10^1$	$1.87 \cdot 10^3$	$8.95 \cdot 10^{-2}$	$4.50 \cdot 10^{-3}$	$1.30 \cdot 10^{-2}$

	F7	F8	F9	F10	F11
Single_5	$4.02 \cdot 10^{-2}$	$1.93 \cdot 10^{11}$	$7.53 \cdot 10^1$	$1.84 \cdot 10^{-3}$	$7.40 \cdot 10^1$
Multi_5	$1.37 \cdot 10^{-2}$	$1.41 \cdot 10^{11}$	$4.49 \cdot 10^1$	$3.46 \cdot 10^{-4}$	$4.44 \cdot 10^1$
Single_10	$2.57 \cdot 10^{-2}$	$1.83 \cdot 10^{11}$	$8.42 \cdot 10^1$	$1.32 \cdot 10^{-3}$	$8.24 \cdot 10^1$
Multi_10	$2.99 \cdot 10^{-2}$	$1.58 \cdot 10^{11}$	$6.30 \cdot 10^1$	$4.60 \cdot 10^{-3}$	$6.19 \cdot 10^1$
Single_20	$1.77 \cdot 10^{-1}$	$2.04 \cdot 10^{11}$	$2.08 \cdot 10^2$	$6.24 \cdot 10^{-2}$	$2.08 \cdot 10^2$
Multi_20	$3.83 \cdot 10^{-2}$	$1.73 \cdot 10^{11}$	$7.78 \cdot 10^1$	$2.76 \cdot 10^{-2}$	$7.63 \cdot 10^1$

lems were considered, the best configuration of the diversity-based MOEA was able to statistically outperform the best configuration of the single-objective GA only in the case of the F6 benchmark problem. For the remaining multi-modal benchmark functions, i.e. F3–F5, the best configurations of both optimisation schemes did not present statistically significant differences.

The above analyses demonstrate the validity of the diversity-based MOEA in terms of the quality attained. However, it is important to quantify the improvement that can be achieved by using the diversity-based MOEA in terms of the number of evaluations invested. To do this, the RLDs described in Section 1.2.6 were applied. The quality level was fixed such that 100% of the executions carried out by the best configurations of both optimisation schemes were able to attain it. The number of evaluations required to achieve a 50% success rate were calculated for the best-behaved configurations of the single-objective GA and the diversity-based MOEA, regardless of the population size. Table 7.6 shows the percentage of saved evaluations by the best configuration of the diversity-based MOEA with respect to the best configuration of the single-objective GA. A negative value means that the single-objective GA was able to converge on the defined quality level faster than the diversity-based MOEA. Once more, we should note the superiority of the diversity-based MOEA. Its best-behaved configuration provided benefits in 9 test cases from a total number of 11. Furthermore, for most of the benchmark functions, the percentage of saved evaluations was significant. For instance, the diversity-based MOEA was able to save 38.7% of the function evaluations in comparison to the single-objective GA for the particular case of the F8 benchmark problem.

Table 7.6: Percentage of evaluations saved by the best configuration of the diversity-based multi-objective evolutionary algorithm

Function	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
%	26.9	-7.7	25.0	-21.2	29.4	28.0	25.9	38.7	29.4	35.3	29.4

Table 7.7: Statistical comparison between different values of the threshold ratio

Function	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
<i>th</i>	0	0.8	0.2, 0.4	0.4, 0.6	0	0, 0.2	0.2	0, 0.2	0.8	0	0.8

7.4.2 On the Application of Diversity-based Objectives with Parameters

This section focuses on analysing the performance of the diversity-based objectives with parameters. Since in the previous experiment the diversity-based MOEA configured with the DCN diversity-based objective provided the best results for most of the problems, the current experiment takes into consideration the combination of the diversity-based MOEA with the DCN-THR diversity-based objective. The main aim of this experiment is therefore to determine the relationship that exists between the values of the threshold ratio th and the quality of the solutions produced. To do so, five configurations of the diversity-based MOEA were applied by considering five different values for the parameter th of the DCN-THR diversity-based objective. These five configurations were applied to the F1–F11 benchmark problems with $D = 500$ decision variables. The parameter values of these configurations were set as follows:

- Since a lower median of the error was achieved by the configurations with lower population sizes in the previous experiment, the population size N was set to 5 individuals.
- Threshold ratio th equal to 0, 0.2, 0.4, 0.6, and 0.8 for the DCN-THR diversity-based objective.
- The UM operator was applied with $p_m = \frac{1}{D} = 0.002$.
- The SBX operator was applied with $p_c = 1$.
- Stopping criterion fixed to a total number of $2 \cdot 10^7$ function evaluations.

Table 7.7 shows, for each benchmark problem, the threshold ratio of the configuration that attained the lowest median of the original objective value in $1.25 \cdot 10^6$ evaluations. It also shows the threshold ratios of those configurations which had no

Table 7.8: Number of evaluations required by different threshold ratios to attain a given quality level

	$th = 0$	$th = 0.2$	$th = 0.4$	$th = 0.6$	$th = 0.8$
F1	$1.25 \cdot 10^6$	$2.20 \cdot 10^6$	$3.05 \cdot 10^6$	$4.70 \cdot 10^6$	$6.05 \cdot 10^6$
F2	$1.35 \cdot 10^6$	$1.35 \cdot 10^6$	$1.40 \cdot 10^6$	$1.40 \cdot 10^6$	$1.25 \cdot 10^6$
F3	$1.75 \cdot 10^6$	$1.25 \cdot 10^6$	$1.35 \cdot 10^6$	$1.70 \cdot 10^6$	$1.80 \cdot 10^6$
F4	$2.15 \cdot 10^6$	$1.45 \cdot 10^6$	$1.25 \cdot 10^6$	$1.25 \cdot 10^6$	$1.50 \cdot 10^6$
F5	$1.25 \cdot 10^6$	$4.00 \cdot 10^6$	$5.15 \cdot 10^6$	$6.90 \cdot 10^6$	$1.74 \cdot 10^7$
F6	$1.25 \cdot 10^6$	$1.20 \cdot 10^6$	$1.95 \cdot 10^6$	$2.50 \cdot 10^6$	$3.75 \cdot 10^6$
F7	$2.30 \cdot 10^6$	$1.25 \cdot 10^6$	$2.00 \cdot 10^6$	$3.85 \cdot 10^6$	$8.60 \cdot 10^6$
F8	$1.25 \cdot 10^6$	$1.30 \cdot 10^6$	$1.45 \cdot 10^6$	$1.40 \cdot 10^6$	$1.45 \cdot 10^6$
F9	$1.90 \cdot 10^6$	$1.85 \cdot 10^6$	$1.75 \cdot 10^6$	$1.65 \cdot 10^6$	$1.25 \cdot 10^6$
F10	$1.25 \cdot 10^6$	$1.80 \cdot 10^6$	$2.40 \cdot 10^6$	$3.25 \cdot 10^6$	$5.00 \cdot 10^6$
F11	$1.85 \cdot 10^6$	$1.85 \cdot 10^6$	$1.85 \cdot 10^6$	$1.60 \cdot 10^6$	$1.25 \cdot 10^6$

statistically significant differences compared to the best-behaved one. We should note that the most suitable values of the parameter th depends on the benchmark function at hand. Hence, in order to decide on the proper value to use, *a priori* information of the optimisation problem being solved is required.

In order to measure the impact on performance, RLDs were calculated. In this case, the quality level was fixed as the median of the original objective value achieved by the best configuration of the diversity-based MOEA in $1.25 \cdot 10^6$ function evaluations. Table 7.8 shows, for each threshold ratio, the number of evaluations needed to attain a 50% success rate taking into account the above quality level. For each benchmark function, the lowest number of evaluations required by the corresponding threshold value is shown in bold. Considering the number of evaluations invested by sub-optimal configurations of the diversity-based MOEA, the importance of correctly setting the parameter th is clear. For instance, for the F5 benchmark problem, the number of evaluations required by the best-behaved configuration represents about 7.18% of the evaluations invested by the worst one.

Another open research issue is whether the proper value of th depends on the stage of the optimisation process or not. With the aim of answering this question the following experiment was performed. First, four different optimisation stages were defined for each benchmark function. To define these stages the median of the original objective value achieved by the best configuration of the diversity-based MOEA for the benchmark problem at hand was calculated using $2.5 \cdot 10^5$, $5 \cdot 10^5$, $7.5 \cdot 10^5$, and $1 \cdot 10^6$ evaluations. Then, for each of these four median values, 32 individuals with a similar original objective value to the median value in question were gener-

Table 7.9: Statistical comparison among different threshold values for each optimisation stage – F4 benchmark function

	Stage 0					Stage 1				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\downarrow
0.2	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\downarrow
0.4	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.6	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.8	\uparrow	\uparrow	\uparrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
	Stage 2					Stage 3				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
0.2	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.4	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.6	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.8	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 7.10: Statistical comparison among different threshold values for each optimisation stage – F5 benchmark function

	Stage 0					Stage 1				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
0.2	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow
0.4	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.6	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.8	\leftrightarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
	Stage 2					Stage 3				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
0.2	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.4	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.6	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.8	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

ated. In order to produce these 32 individuals, the best-behaved configuration for the benchmark at hand was executed 32 times. For each of these 32 executions, the first individual able to attain an original objective value lower than the median value calculated for the optimisation stage in question was selected. After doing this, for each stage and for each benchmark problem, the diversity-based MOEA

Table 7.11: Statistical comparison among different threshold values for each optimisation stage – F11 benchmark function

	Stage 0					Stage 1				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.2	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.4	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.6	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
0.8	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
	Stage 2					Stage 3				
	0	0.2	0.4	0.6	0.8	0	0.2	0.4	0.6	0.8
0	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow
0.2	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow
0.4	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow
0.6	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow
0.8	\leftrightarrow	\uparrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\leftrightarrow

was executed for different values of the parameter th . Considering a certain test case, for each optimisation stage, every execution of the diversity-based MOEA included in its initial population one of the corresponding 32 individuals generated by the above procedure. The remaining individuals in the population were randomly generated. In this way, the performance of the diversity-based MOEA configured with different threshold values was tested when it started from different quality levels. The stopping criterion was fixed to $5 \cdot 10^5$ function evaluations. Lastly, the separate configurations of the diversity-based MOEA were compared in terms of the original objective value attained at the end of the executions.

Tables 7.9, 7.10, and 7.11 show the comparison of the threshold values used for the F4, F5, and F11 benchmark functions, respectively. For each optimisation stage, they show whether the scheme located in a given row is statistically better (\uparrow), not different (\leftrightarrow), or worse (\downarrow) than the corresponding scheme situated in a certain column. For each benchmark problem, the statistical tests demonstrate that differences among configurations depend on the stage of the optimisation process. For instance, for stage number 0 in problem F4, the configuration that used $th = 0.8$ is better than the configuration where $th = 0$. In the optimisation stage number 3, however, the configuration with $th = 0$ performed better than the configuration with $th = 0.8$. In the case of problem F5, $th = 0$ seems to be the most appropriate value for the whole run because there was no better value in any of the analysed stages. For these same reasons, the value $th = 0.8$ seems to be the most appropriate

Table 7.12: Number of evaluations required by the hyper-heuristic to attain a given quality level

	F1	F2	F3	F4	F5	F6
HH-PROB	$1.45 \cdot 10^6$	$1.35 \cdot 10^6$	$1.30 \cdot 10^6$	$1.15 \cdot 10^6$	$1.40 \cdot 10^6$	$1.25 \cdot 10^6$
	F7	F8	F9	F10	F11	
HH-PROB	$1.50 \cdot 10^6$	$1.40 \cdot 10^6$	$1.55 \cdot 10^6$	$1.20 \cdot 10^6$	$1.55 \cdot 10^6$	

for problem F11.

7.4.3 Improving the Robustness of the Diversity-based Multi-objective Evolutionary Algorithm

In the preceding section, it was shown that the appropriate value for the parameter th of the DCN-THR diversity-based objective not only depends on the optimisation problem considered, but also on the current stage of the optimisation process. As a result, this experiment is devoted to analysing whether the application of a hyper-heuristic that controls the threshold ratio th belonging to the DCN-THR diversity-based objective is able to improve both the behaviour and the robustness of the diversity-based MOEA. To do so, the HH-PROB hyper-heuristic was applied to the F1–F11 benchmark functions with $D = 500$ decision variables. It was configured with an adaptation level $k = \infty$, and the value of β was set such that 10% of its decisions used a uniform distribution, i.e. $\beta \cdot n_h = 0.1$. The low-level approaches consisted of $n_h = 5$ configurations of the diversity-based MOEA defined in the previous section. Recall that the values 0, 0.2, 0.4, 0.6, and 0.8 were considered for the threshold ratio th . Hence, the low-level configurations only differ in the value that this parameter takes. Finally, the local stopping criterion of the HH-PROB hyper-heuristic was fixed to $1 \cdot 10^4$ function evaluations, whereas the global stopping criterion was set to $2.5 \cdot 10^6$ evaluations.

Table 7.12 shows, for each benchmark function, the number of evaluations required by the hyper-heuristic to attain a 50% success rate. The same quality level as that applied to obtain the results shown in Table 7.8 was used, i.e. for each test case, it was fixed as the median of the original objective value achieved by the best-behaved low-level configuration of the diversity-based MOEA after $1.25 \cdot 10^6$ evaluations. Note that the advantages of controlling the parameter th by means of the HH-PROB scheme are clear. In fact, it was the only approach able to reach the specified quality level in under $1.55 \cdot 10^6$ evaluations for all the benchmark

Table 7.13: Resources assigned by the hyper-heuristic for benchmark problems with no variability in the threshold ratio

	$th = 0$	$th = 0.2$	$th = 0.4$	$th = 0.6$	$th = 0.8$
F5	56.98%	15.77%	10.82%	9.79%	6.64%
F11	15.96%	16.05%	13.82%	20.89%	33.28%

problems considered. In addition, for two benchmark functions—F4 and F10—it was the approach that required the fewest number of evaluations to achieve the specified quality level. This means that the hyper-heuristic was able to allocate more resources to the most suitable low-level configuration for each optimisation stage. Furthermore, for the remaining benchmark functions, the HH-PROB hyper-heuristic was able to reach the specified quality level in a number of evaluations similar to that required by the best low-level configuration. Lastly, considering that the hyper-heuristic provides its solutions in a single run, the benefits are even more noticeable, since it provides similar or even better results than those obtained by several low-level configurations executed independently.

Usually, the hyper-heuristic requires a certain amount of time to determine which configuration is the fittest approach for each stage of the optimisation process. Moreover, it ensures that every low-level configuration considered will be executed with a probability higher than or equal to $\beta \cdot n_h$. Thus, by the end of the executions, every low-level configuration will have been executed several times. Consequently, in cases where the same threshold value is better than the remaining ones in every stage, the hyper-heuristic is not able to converge to high-quality solutions as fast as the best-behaved low-level configuration. It would be interesting to know, however, the amount of resources allocated to the best low-level configuration for these types of problems. Table 7.13 shows the percentage of resources allocated to each low-level configuration for the F5 and F11 benchmark problems. These problems were selected because for both cases there is a value for the parameter th that behaves properly in all the optimisation stages studied. As was stated in the preceding section, the best-behaved low-level configuration used $th = 0$ for the F5 benchmark. More than 50% of the resources were allocated to this configuration. For the F11 test case, the best-behaved low-level approach used $th = 0.8$. In this case, the hyper-heuristic allocated the largest amount of resources to this configuration. However, in no case was the difference among the resources allocated to each configuration as significant as in the case of F5. The reason is that the statistical differences among the low-level approaches were not so clear, as shown in Table 7.11. Therefore, the hyper-heuristic allocated more resources to other configurations.

Table 7.14: Statistical comparison between the hyper-heuristic and different low-level configurations

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
↑	4	1	2	5	4	3	4	0	4	5	4
↔	0	3	3	0	0	2	0	3	0	0	0
↓	1	1	0	0	1	0	1	2	1	0	1

Finally, in order to better analyse the benefits of the HH-PROB hyper-heuristic, the original objective value was statistically compared to that obtained at the end of the executions. Table 7.14 compares the HH-PROB scheme against its five corresponding low-level configurations. For each benchmark function, it shows the number of cases where the hyper-heuristic was statistically better (↑), not different (↔), or worse (↓) than the low-level configurations. We should note that in four problems—F3, F4, F6, and F10—none of the low-level configurations was able to statistically outperform the hyper-heuristic. In fact, for the F4 and F10 benchmark functions, the hyper-heuristic was statistically better than every low-level approach. In six problems—F1, F2, F5, F7, F9, and F11—only one low-level configuration was able to statistically outperform the HH-PROB approach. Lastly, two low-level configurations were statistically better than the hyper-heuristic in the case problem F8. However, as was previously stated, the HH-PROB hyper-heuristic was able to provide solutions of a quality similar to that produced by the best low-level configurations for these benchmark problems.

7.4.4 Analysis of the Parameter Control Schemes Over a Short Evaluation Timeframe

In the previous section, a hyper-heuristic was applied as a control approach to adapt the parameter th of the DCN-THR diversity-based objective. As was stated, the use of this hyper-heuristic was able to significantly improve the performance of the proposed diversity-based MOEA. In the current experiment, the different parameter control schemes described in Sections 7.3.1 and 7.3.2 are applied to the parameter th of the DCN-THR diversity-based objective in order to solve each of the F1–F19 benchmark functions with $D = 500$ decision variables. The main aim of this study, therefore, is to carry out a broad comparison among different kinds of parameter control techniques using a relatively short period of $5 \cdot 10^5$ function evaluations.

The experiments conducted used a common parameterisation for the diversity-based MOEA and the different parameter control schemes. Table 7.15 shows the pa-

Table 7.15: Parameterisation of the diversity-based multi-objective evolutionary algorithm

Parameter	Value	Parameter	Value
Stopping criterion	$2.5 \cdot 10^6$ evals.	Mutation rate (p_m)	0.002
Population size (N)	5 individuals	Crossover rate (p_c)	1

Table 7.16: Parameterisation of the HH-ELI and HH-PROB hyper-heuristics

Parameter	Value	Parameter	Value
Local stopping criterion	$2.5 \cdot 10^3$ evals.	Minimum selection rate (β)	0.1
Low-level configs. (n_h)	6	Historical knowledge (k)	5

Table 7.17: Parameterisation of the FUZZY-A and FUZZY-B fuzzy logic controllers

Parameter	Value	Parameter	Value
Number of generations ($numGen$)	$5 \cdot 10^2$	Difference among samples (Δ)	0.1
Linguistic terms ($numTerms$)	7	Historical knowledge (k)	5

parameterisation of the diversity-based MOEA. The parameterisations of the different parameter control approaches are shown in Tables 7.16 and 7.17 for the hyper-heuristics and the FLCs, respectively. The parameter values for the control methods—minimum selection rate, historical knowledge, number of low-level configurations, number of linguistic terms, etc.—were the same regardless of the benchmark problem considered. This means that the control approaches proposed herein are robust, since promising results can be obtained for a wide range of test cases without changing these parameter values. Thus, the parameters of the control methods do not impose additional configuration burdens on the diversity-based MOEA.

The HH-ELI and HH-PROB hyper-heuristics were applied using $n_h = 6$ low-level configurations. Generally, having a high number of low-level configurations leads to lower quality solutions because the hyper-heuristic is not able to make the right decisions when a large set of candidate approaches is defined. For this reason, six low-level configurations were selected instead of setting a larger value for the parameter n_h . The only difference among the low-level configurations was the value assigned to the parameter th . The values were distributed uniformly in the range $[0, 1]$. Hence, the low-level configurations were defined with values for the parameter th equal to 0, 0.2, 0.4, 0.6, 0.8, and 1. The remaining parameters of the low-level configurations took the same values as those shown in Table 7.15. Describing the parameterisation of the self-adaptive parameter control approaches is not necessary.

Table 7.18: Mean original objective value for the best parameter control schemes after $5 \cdot 10^5$ evaluations

Problem	SELF	HH-ELI	HH-PROB	FUZZY-A	FUZZY-B
F1	-4.478003e+02	-4.478371e+02	-4.474119e+02	-4.483882e+02*	-4.481555e+02
F2	-3.976507e+02	-4.002689e+02*	-3.995730e+02	-3.977567e+02	-3.984597e+02
F3	7.860950e+03	6.307528e+03*	6.892168e+03	7.114925e+03	7.206458e+03
F4	-2.788402e+02	-3.106858e+02*	-3.054929e+02	-3.050468e+02	-3.054261e+02
F5	-1.798369e+02	-1.798284e+02	-1.798023e+02	-1.798794e+02	-1.798873e+02*
F6	-1.399212e+02	-1.399205e+02	-1.399115e+02	-1.399267e+02	-1.399412e+02*
F7	3.740799e-01	2.928847e-01	3.744123e-01	2.395465e-01*	2.492768e-01
F8	3.570116e+05	3.664986e+05	3.720928e+05	3.563382e+05*	3.647259e+05
F9	2.968789e+02	2.957150e+02	3.014180e+02	2.928711e+02	2.866811e+02*
F10	1.812262e+00*	2.249266e+00	2.478268e+00	2.140365e+00	1.893920e+00
F11	2.949134e+02*	3.035599e+02	3.073306e+02	2.963466e+02	2.999489e+02
F12	4.009808e+01	3.308254e+01*	3.481530e+01	3.874683e+01	3.827866e+01
F13	1.017289e+04	3.860397e+03*	4.409750e+03	7.421083e+03	8.632196e+03
F14	4.569614e+01	3.072660e+01*	3.281441e+01	3.343408e+01	3.537184e+01
F15	4.122504e+00	2.468268e+00*	2.646807e+00	3.470253e+00	3.276949e+00
F16	1.114501e+02	9.613186e+01*	1.033108e+02	1.109923e+02	1.092793e+02
F17	2.689799e+03	1.287036e+03*	1.564690e+03	2.383842e+03	2.068250e+03
F18	5.346952e+01	5.112424e+01	5.419024e+01	4.860644e+01*	5.111202e+01
F19	2.062142e+00	1.619345e+00*	1.982881e+00	2.149803e+00	2.119459e+00

Since no additional parameters are defined for them, their parameterisation was the same as that applied with the diversity-based MOEA—Table 7.15. Finally, note that in order to compare the different control approaches, all the methods were run using a stopping criterion equal to $2.5 \cdot 10^6$ function evaluations, as shown in Table 7.15.

Table 7.18 shows, for each benchmark function, the mean of the original objective value achieved by the different parameter control approaches after $5 \cdot 10^5$ evaluations. In the case of the different versions of the self-adaptive approach, only the data for the scheme that attained the lowest mean of the original objective value after $5 \cdot 10^5$ evaluations are shown. Those parameter control approaches whose data are shown in bold with an asterisk obtained the lowest mean of the original objective value. It is important to remark that said approaches exhibited statistically significant differences compared to the control methods whose data are not shown in bold. If the data belonging to several parameter control approaches are shown in bold without an asterisk, then those schemes did not exhibit statistically significant differences relative to those that achieved the lowest mean of the original objective value. The following observations apply:

- One or both of the FUZZY-A and FUZZY-B FLCs were statistically better than the remaining parameter control approaches in **four** problems: F1, F5, F6, and F7.

- One or both of the HH-ELI and HH-PROB hyper-heuristics were statistically better than the remaining parameter control schemes in **eight** problems: F2, F3, F4, F12, F13, F15, F16, and F17.
- In **five** problems—F8, F9, F10, F14, and F18—there were no statistically significant differences between both types of external controllers, i.e. between hyper-heuristics and FLCs.
- The self-adaptive approaches did not provide any statistically significant advantage on their own in any of the benchmarks.

Hence, following a short evaluation period, the hyper-heuristics, and in particular the HH-ELI approach, appear better able to adapt the parameter th than the parameter control approaches based on FLCs and self-adaptation. This could be due to the fact that the hyper-heuristic-based methods only have to select the most suitable value for the parameter th from amongst a finite set of candidate values that coarsely cover the parameter space. In contrast, the FLCs are able to select from an infinite range of possible values. Thus, given only a *short* learning period, the hyper-heuristics outperform the other methods as they are able to exploit existing values that may lie close to the optimal ones, rather than having to explore the space for suitable values. As a result, the methods based on hyper-heuristics are able to provide better solutions than the FUZZY-A and/or FUZZY-B schemes in the short term for many problems. With respect to the self-adaptive approaches, although they did not outperform either of the other control approaches, for some problems they were able to achieve the same quality level as the other parameter control methods. In the case of the F19 problem, they outperformed the FLCs together with the hyper-heuristics, whereas in the case of the F11 benchmark function, they outperformed the hyper-heuristics together with the FLCs.

7.4.5 Analysis of the Parameter Control Schemes Over a Long Evaluation Timeframe

In this section, the parameter control schemes are analysed over a longer evaluation period of $2.5 \cdot 10^6$ evaluations, i.e. at the end of the executions. Table 7.19 shows the same information as Table 7.18, but for $2.5 \cdot 10^6$ function evaluations. The following observations apply:

- One or both of the FUZZY-A and FUZZY-B FLCs were statistically better than the remaining parameter control approaches in **four** problems: F3, F7, F12, and F14.

Table 7.19: Mean original objective value for the best parameter control schemes after $2.5 \cdot 10^6$ evaluations

Problem	SELF	HH-ELI	HH-PROB	FUZZY-A	FUZZY-B
F1	-4.499892e+02	-4.499992e+02*	-4.499988e+02	-4.499988e+02	-4.499992e+02*
F2	-4.341193e+02	-4.350975e+02*	-4.349620e+02	-4.340944e+02	-4.341680e+02
F3	1.427243e+03	1.807100e+03	1.718612e+03	1.282274e+03*	1.547327e+03
F4	-3.299951e+02	-3.300000e+02*	-3.299999e+02	-3.299998e+02	-3.299998e+02
F5	-1.799766e+02	-1.799713e+02	-1.799836e+02*	-1.799717e+02	-1.799792e+02
F6	-1.399978e+02	-1.399990e+02	-1.399988e+02	-1.399991e+02*	-1.399991e+02*
F7	7.065683e-03	4.597348e-03	5.994893e-03	2.057316e-03*	2.533643e-03
F8	1.668117e+05	1.676752e+05	1.690159e+05	1.624589e+05*	1.706433e+05
F9	3.248838e+01	1.681051e+01	2.372986e+01	1.617945e+01	1.613916e+01*
F10	3.394049e-03	2.852807e-04*	3.736034e-04	5.373773e-04	3.173994e-04
F11	3.371631e+01	1.598078e+01*	2.490122e+01	1.728919e+01	1.728853e+01
F12	9.409282e-01	9.932408e-01	1.060437e+00	5.706366e-01*	5.746609e-01
F13	1.948127e+03	1.782808e+03	1.290099e+03*	1.473589e+03	1.581966e+03
F14	3.839717e-01	1.958826e-01	3.125988e-01	1.418254e-01	1.371941e-01*
F15	1.514281e-02	6.026768e-03*	7.459278e-03	6.769493e-03	6.590881e-03
F16	3.635383e+00	2.254722e+00	3.054237e+00	2.237479e+00	2.185866e+00*
F17	8.514682e+02	5.468047e+02	5.769930e+02	7.511799e+02	5.339560e+02*
F18	3.500672e+00	1.813913e+00*	2.260901e+00	2.039246e+00	1.859091e+00
F19	5.165002e-03	9.889879e-04	1.524792e-03	1.077074e-03	9.886900e-04*

- One or both of the HH-ELI and HH-PROB hyper-heuristics were statistically better than the remaining parameter control schemes in **two** problems: F2 and F4.
- In **thirteen** problems, there were no statistically significant differences between both types of external controllers: F1, F5, F6, F8, F9, F10, F11, F13, F15, F16, F17, F18, and F19.
- The self-adaptive approaches did not provide any statistically significant advantage on their own in any of the benchmarks.

It is clear that given a long enough evaluation time, the parameter control methods become less distinguishable. In 13 test cases in the long evaluation period, no statistically significant differences in the external methods were observed, compared to 5 cases in the short evaluation period. It is also clear that given a longer runtime, the FLCs were able to provide the best results in a higher number of problems than the hyper-heuristics. Regarding the self-adaptive approaches, as in the short term, again they did not provide any benefit for any benchmark in the long term, and in fact were outperformed by the FLCs and hyper-heuristics for all problems, except for the F5 and F8 benchmark functions.

7.4.6 Comparison between Short and Long Evaluation Periods

It is illuminating to compare the change in the performance of the control methods for each of the problems under the two different evaluation scenarios, short term and long term. The *winning* method is defined herein as the parameter control method that statistically outperforms each of the other control methods. Hence, the winner is either self-adaptation, hyper-heuristic, FLC or none if no single method differs statistically from all of the others. If we then examine the change in the winning method as we switch from short-term to long-term evaluation, the following observations can be made:

- For **eight** problems—F4, F7, F8, F9, F10, F11, F18, and F19—changing the length of the evaluation period had no impact on the winning control method.
- For **seven** problems—F1, F5, F6, F13, F15, F16, and F17—while there was a clear winner in the short-term evaluation experiment, there was no significant difference between methods when evaluated for a longer time period.
- For **two** problems—F3 and F12—the best method switched from the hyper-heuristic in the short-term evaluation to the FLCs in the long-term evaluation experiment.
- For **one** problem—F14—while there was no significant difference in method in the short term, the FLCs was the best method in the long term.
- For **one** problem—F2—while there was no significant difference in method in the short term, the hyper-heuristic was the best method in the long term.

These results suggests that the coarse-grained approach of the hyper-heuristic, which defines a fixed set of possible values for the parameter th spread uniformly across the full range of possible values, provides sufficient variety in this parameter to yield high quality results for most of the problems. The results also suggest that the FLCs select similar values to those already defined by the hyper-heuristics and that the problems are relatively robust to the exact value of th over some interval. This is evidenced by the fact that in 7 problems, the performance of the hyper-heuristic and FLCs converged given a long enough evaluation time. In addition, for 13 test cases, hyper-heuristics and FLCs did not present statistically significant differences at the end of the executions.

For 3 problems—F3, F12, and F14—in the long term the FLC emerged as the winner while in the short term, either the hyper-heuristic was the clear winner or

no method dominated. This suggests that these problems are particularly sensitive to the parameter th , and that the FLC is able to find a value not present in any of the hyper-heuristic configurations that gives better results.

In contrast, for only one problem, while no method arose as a winner when only evaluated for a short time period, the hyper-heuristic method dominated in the long evaluation period. This can perhaps be explained by the fact that finding good solutions for this problem is more challenging, though given a long enough evaluation period, the hyper-heuristic is able to eventually dominate. It is possible that in this case, running the experiment for an even longer period would enable the FLCs to catch up.

To sum up, the most appropriate parameter control approach depends on the optimisation problem being solved. However, for a significant number of cases, FLCs or hyper-heuristics can be applied to obtain promising results, whereas the self-adaptive approaches are not able to provide any advantage over the other control methods when the parameter th is adapted. The results appear to verify the statement made by Eiben [106]: *“self-adaptive methods are efficient methods when applicable ... but are outperformed by clever adaptive methods”*.

7.4.7 Comparing Parameter Control and Parameter Tuning

The parameter control methods applied in preceding sections adapt the value of the parameter th during the course of an evolutionary run; thus, a single run of the optimisation algorithm may utilise many different values over the entire execution. In this section, we assess the benefit of adapting the value of the parameter over the course of the run as opposed to simply choosing a single value that remains fixed throughout. The latter approach requires a suitable value for th to be defined; 21 different configurations of the diversity-based MOEA with the same parameterisation as that used in previous experiments—Table 7.15—were defined, in which each configuration differs only in the value of th . The 21 values of th tested were distributed uniformly in the range $[0, 1]$.

Figure 7.2 shows the mean of the original objective value achieved in $2.5 \cdot 10^6$ evaluations by the parameter control approaches and by the diversity-based MOEA executed with a range of fixed values of the parameter th —FIXED—for several of the benchmark functions. The conclusions drawn from the plots shown are generalisable to the remaining test cases.

Observing the plots, at least one of the parameter control methods was able to

7.4. Experimental Evaluation and Discussion

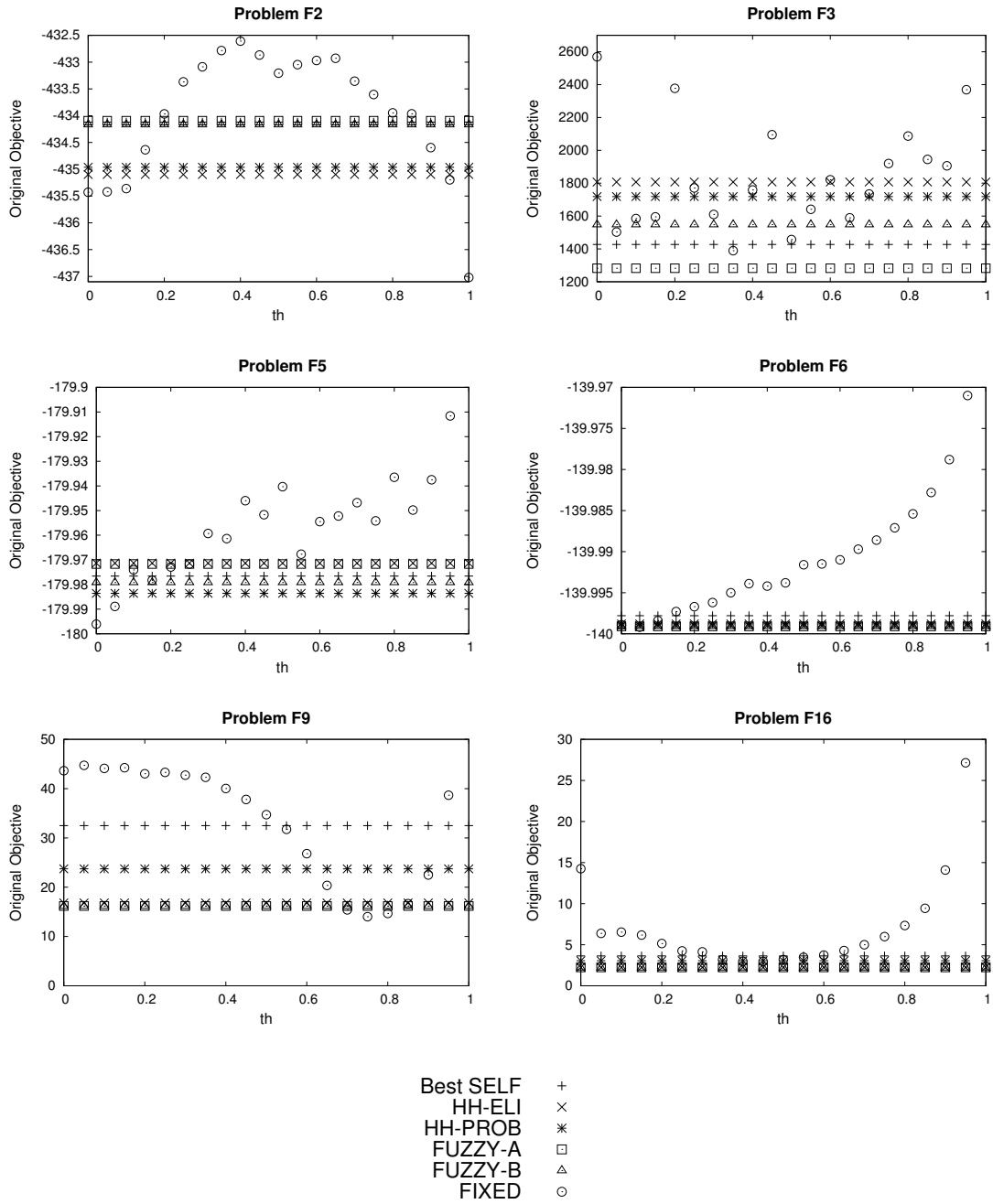


Figure 7.2: Mean of the original objective value achieved by the parameter control methods and by fixed values of the threshold ratio

obtain either similar or better results than any configuration of the FIXED scheme in 12 of the 19 problems, namely in problems F3, F6, and F16 shown in Figure 7.2, and also in benchmarks F1, F4, F7, F10, F12, F13, F14, F15, and F19. Hence, it appears that the majority of the benchmarks benefitted from an approach in which th could be varied during the course of the run and an optimal fixed value of th could not be found.

In contrast, in 7 of the 19 test cases—F2, F5, F8, F9, F11, F17, and F18—some configurations of the FIXED approach yielded better results than the parameter control approaches, suggesting that there exists some fixed value for the parameter th that is adequate during the whole optimisation process. An alternative explanation however might lie in the fact that adapting th may improve the algorithm but that the changes in the values of the parameter th take place so fast that parameter control approaches are not able to detect these changes at the rate required. In this case, fixing the parameter to a suitable value produces more robust behaviour in the diversity-based MOEA. Despite this fact, the results obtained by the control techniques were close to those provided by the best configurations of the FIXED approach for this set of 7 problems.

It is crucial to note that finding a suitable fixed value for th required 21 separate runs of the optimisation algorithm. These results are in comparison to a *single* run of the parameter control methods. Consequently, in addition to the fact that the parameter control methods obtain high quality solutions in the majority of problems, the savings in computational resources and time required to produce a good solution are significant across all benchmark functions.

In order to quantify these savings, an additional analysis was conducted involving the RLDs. In this case, the quality level was set as the highest median of the original objective value achieved by the schemes in question at the end of the executions, i.e. after $2.5 \cdot 10^6$ evaluations.

We further calculate the percentage of evaluations p saved by a certain scheme, in comparison to the approach that required the largest number of evaluations. This savings was calculated using Equation 7.1, where the number of evaluations performed by the scheme considered is denoted by $numEvals$, and $maxEvals$ is the largest number of evaluations performed by any approach, for a particular benchmark function.

$$p = \left(1 - \frac{numEvals}{maxEvals}\right) \cdot 100 \quad (7.1)$$

Table 7.20: Maximum number of evaluations required to attain the specified quality level and percentage of evaluations saved by each approach

Problem	SELF	HH-ELI	HH-PROB	FUZZY-A	FUZZY-B	FIXED
F1	2.45e+06	36.73%	30.61%	38.78%	40.82%	42.86%
F2	2.50e+06	6.00%	6.00%	2.50e+06	2.50e+06	20.00%
F3	16.00%	4.00%	2.50e+06	34.00%	22.00%	22.00%
F4	12.50%	39.58%	31.25%	33.33%	39.58%	2.40e+06
F5	6.67%	42.22%	2.25e+06	46.67%	53.33%	53.33%
F6	2.40e+06	27.08%	18.75%	37.50%	31.25%	50.00%
F7	2.50e+06	16.00%	8.00%	42.00%	36.00%	44.00%
F8	2.00%	2.00%	2.00%	10.00%	2.50e+06	12.00%
F9	2.50e+06	24.00%	12.00%	24.00%	24.00%	28.00%
F10	2.50e+06	32.00%	30.00%	36.00%	42.00%	28.00%
F11	2.50e+06	28.00%	12.00%	26.00%	22.00%	28.00%
F12	6.00%	6.00%	2.50e+06	32.00%	28.00%	34.00%
F13	2.50e+06	72.00%	64.00%	54.00%	52.00%	80.00%
F14	2.50e+06	26.00%	20.00%	28.00%	32.00%	14.00%
F15	2.50e+06	18.00%	12.00%	16.00%	14.00%	20.00%
F16	2.50e+06	16.00%	8.00%	16.00%	18.00%	10.00%
F17	2.50e+06	72.00%	68.00%	46.00%	48.00%	82.00%
F18	2.50e+06	24.00%	12.00%	16.00%	24.00%	28.00%
F19	2.50e+06	28.00%	20.00%	28.00%	30.00%	26.00%
Mean	2.27%	27.35%	18.66%	29.70%	29.31%	32.75%

The results are shown in Table 7.20. For each benchmark problem, the table gives the actual number of evaluations taken by the method that required the highest number to achieve a 50% success rate. In every other column, the percentage of evaluations saved by the corresponding method is shown. In order to calculate the values, in the case of the self-adaptive control scheme, the variant that obtained the lowest mean of the original objective at the end of the executions was used. The single value of the threshold ratio th that was found to be optimal in the experiments performed with the FIXED approach was also taken into account. The data in bold highlight the approaches that were able to save the highest number of evaluations for each problem. Finally, the last row shows the mean percentage of saved evaluations for all the test cases.

Note that for the majority of the benchmarks, the self-adaptive approach required the highest number of evaluations to achieve a 50% success rate. The FIXED approach provided the most savings in evaluations in 13 benchmark functions. However, it is important to note that in order to find the appropriate value of th to

use in this experiment, 21 separate configurations had to be executed, a significant hidden cost that is not apparent in this table. Quite the opposite, parameter control methods based on FLCs and hyper-heuristics obtained the largest number of saved evaluations in 8 test cases and required only a single run of the algorithm. In 6 of these 8 test cases, the FUZZY-B FLC provided the largest savings.

We should note that the size of the savings in the case of the control approaches was also significant for the 13 test cases where the FIXED scheme provided the greatest savings. For instance, for the F1 benchmark problem the FIXED approach resulted in 42.86% fewer evaluations, while the FUZZY-B scheme provided a 40.82% of savings. This fact is also evidenced by the mean percentages of saved evaluations. It can be observed that the FLCs obtained mean percentages quite close to the mean percentage provided by the FIXED scheme. This demonstrates the advantages of using the proposed control approaches over searching for a fixed value for the parameter th , i.e. the advantages of parameter control against parameter tuning.

7.4.8 Analysis of the Hybrid Control Scheme based on Fuzzy Logic Controllers and Hyper-heuristics

In preceding experiments, different parameter control approaches based on FLCs, hyper-heuristics and other methods were compared. The main advantage of hyper-heuristics is that they are able to control symbolic and numeric parameters. The size of the set of low-level configurations is generally fixed and finite, however, meaning that in the case of controlling numeric parameters, the number of possible values that can be assigned to them is therefore also finite. In contrast, the main benefit of using the FLCs is that the possible values that can be assigned to a certain parameter are not selected from a finite set. Its main drawback lies in the fact that it cannot be directly applied to control symbolic parameters. In order to avoid their drawbacks, and to profit from the strong points of the two approaches, the hybrid control scheme described in Section 7.3.3 is applied herein to simultaneously adapt symbolic and numeric parameters. The main aim of this experiment is twofold. First, to study the performance of the proposed hybrid control scheme. Second, to analyse whether parameter control offers any benefits with regard to tuning the crossover and mutation operators, and the mutation rate p_m belonging to the diversity-based MOEA depicted in Section 7.2.2. We will proceed by considering the F1–F19 benchmark functions with $D = 500$ decision variables.

Table 7.21 shows the parameterisation of the diversity-based MOEA. First, 72 different configurations of the diversity-based MOEA with fixed values for the parameters

Table 7.21: Parameterisation of the diversity-based multi-objective evolutionary algorithm

Parameter	Value	Parameter	Value
Stopping criterion	$2.5 \cdot 10^6$ evals.	p_c	1
Population size (N)	5 individuals	p_m – UM-ONE, PM-ONE	0.2, 0.36, 0.52, 0.68, 0.84, 1
Crossover operators	SBX, AX, PBX- α	p_m – UM-ALL, PM-ALL	0.0002, 0.00056, 0.00092, 0.00128, 0.00164, 0.002
Mutation operators	UM-ONE, PM-ONE UM-ALL, PM-ALL	Diversity-based objective	DCN-THR

Table 7.22: Parameterisation of the HH-PROB hyper-heuristic

Parameter	Value	Parameter	Value
Local stopping criterion	$2.5 \cdot 10^4$ evals.	Minimum selection rate (β)	0.1
Number of low-level configs. (n_h)	12 configs.	Historical knowledge (k)	5

Table 7.23: Parameterisation of the FUZZY-A fuzzy logic controller

Parameter	Value	Parameter	Value
Local stopping criterion ($numEvals$)	$5 \cdot 10^2$ evals.	Difference among samples (Δ)	0.1
Number of linguistic terms ($numTerms$)	7	Historical knowledge (k)	5
Range of p_m – UM-ONE, PM-ONE	[0.2, 1]	Range of p_m – UM-ALL, PM-ALL	[0.0002, 0.002]

were executed for each benchmark function. The configurations were obtained by combining the values shown in Table 7.21 for the crossover and mutation operators, and for the mutation rate p_m . Two different variants of the UM and PM operators were used. The variants UM-ONE and PM-ONE mutate a unique parent gene with probability p_m , whereas the variants UM-ALL and PM-ALL mutate every parent gene with probability p_m . The values assigned to p_m differed depending on the mutation operator applied. In addition, the hybrid control scheme was executed for each benchmark problem to adapt the values of the crossover and mutation operators, and the mutation rate p_m of the diversity-based MOEA. The remaining parameters were kept as shown in Table 7.21. The hyper-heuristic and the FLC parameterisations are shown in Tables 7.22 and 7.23, respectively. Note that the hyper-heuristic of the hybrid scheme had to select among $n_h = 12$ low-level configurations. This is because twelve possible combinations resulted from using the three crossover operators and the four variants of the mutation operators shown in Table 7.21. Therefore, the differences among the low-level configurations lied in the particular values given to the crossover and mutation operators. Lastly, depending on the mutation operator defined for each low-level configuration, the range of possible values that the FLC was able to infer for p_m was different.

Table 7.24 shows, for each benchmark function, the parameter values of the best configuration—the one that achieved the lowest median of the error with respect

Table 7.24: Values for the parameters of the best fixed configuration

Problem	Crossover	Mutation	p_m
F1	SBX	UM-ALL	0.0002
F2	PBX- α	UM-ALL	0.00092
F3	PBX- α	UM-ALL	0.00056
F4	SBX	UM-ONE	0.68
F5	SBX	UM-ONE	0.2
F6	PBX- α	PM-ALL	0.00056
F7	PBX- α	PM-ALL	0.0002
F8	AX	PM-ALL	0.00128
F9	PBX- α	UM-ALL	0.0002
F10	SBX	UM-ALL	0.0002
F11	PBX- α	UM-ALL	0.0002
F12	PBX- α	UM-ALL	0.0002
F13	SBX	UM-ALL	0.0002
F14	PBX- α	UM-ALL	0.0002
F15	PBX- α	PM-ALL	0.0002
F16	PBX- α	UM-ALL	0.0002
F17	SBX	UM-ALL	0.0002
F18	PBX- α	UM-ALL	0.0002
F19	SBX	UM-ALL	0.0002

to the original objective value—of the diversity-based MOEA executed with fixed parameters. Recall that 72 different configurations of the diversity-based MOEA with fixed parameters were executed for each function. With regard to parameter tuning, the PBX- α crossover operator seems to be the most suitable, while the most appropriate values for the mutation operator and its rate p_m seem to be the UM-ALL operator and 0.0002, respectively, for a wide range of benchmark functions. However, the most appropriate values for the crossover and mutation operators, and for the mutation rate p_m , change depending on the benchmark function in question. As a result, it would be interesting to check whether the novel hybrid control approach is able to provide similar results without the need to execute a large amount of configurations of the diversity-based MOEA in order to look for its best parameter values. For benchmarks where different parameter values are the most appropriate depending on the stage of the search process, the hybrid control scheme might provide even better results than those given by the best fixed configuration of the diversity-based MOEA, as was stated in previous sections.

Table 7.25 shows, for each problem, the median of the error achieved by the hybrid parameter control scheme and by the best fixed configuration of the diversity-based

Table 7.25: Median of the error attained by the hybrid control scheme and by the best fixed configuration

Problem	Hybrid Scheme	Best Fixed Conf.	↑	↓	↔
F1	1.3281037e-06	3.4106051e-13	66	5	1
F2	7.6975000e+00	1.0592500e+01	72	0	0
F3*	9.4457000e+02	2.9075500e+02	40	18	13
F4	3.0795973e-05	7.8476094e-04	72	0	0
F5*	2.6413124e-07	8.3753093e-10	68	0	2
F6	8.3838312e-05	7.5147903e-05	71	0	1
F7	1.6071950e-04	2.1048950e-04	72	0	0
F8	4.7819750e+04	3.2891500e+04	55	15	2
F9	3.5459550e+00	5.5819750e+00	72	0	0
F10	1.5165250e-06	5.1276500e-13	65	5	2
F11	3.6804850e+00	5.6666000e+00	72	0	0
F12	1.5824200e-01	1.1245450e-01	71	1	0
F13*	7.8921500e+02	2.9386450e+02	62	2	4
F14	5.1547950e-02	7.4521000e-02	72	0	0
F15	1.3515900e-04	9.6688000e-05	71	1	0
F16	1.0623750e+00	1.7717600e+00	72	0	0
F17	9.0679350e+01	7.1111250e+01	67	0	5
F18	4.2261050e-01	6.4425800e-01	72	0	0
F19	3.0685100e-05	3.0367050e-08	67	5	0

MOEA—the one shown in Table 7.24. For benchmarks where differences were statistically significant, data belonging to the approach that obtained the lowest median and mean of the error are shown in bold. In contrast, if differences between the hybrid control scheme and the best fixed configuration were not statistically significant, the data for both approaches are not shown in bold. This was the case for **three** benchmark problems: F5, F6, and F17. Furthermore, for each problem, the hybrid control scheme was statistically compared to the 72 fixed configurations of the diversity-based MOEA. Hence, Table 7.25 also shows the number of fixed configurations which were statistically outperformed (↑) by the hybrid scheme, the number of fixed configurations that statistically outperformed the hybrid scheme (↓), and the number of fixed configurations which did not present statistically significant differences with the hybrid method (↔). Finally, for test cases where an asterisk is shown, some fixed configurations of the diversity-based MOEA presented statistically significant differences with the hybrid scheme. However, one of the two approaches obtained the lowest mean of the error, while the other obtained the lowest median of the error, so the winning approach could not be determined.

It is worth mentioning that for **eight** benchmark functions—F2, F4, F7, F9, F11, F14, F16, and F18—the hybrid control scheme was able to statistically outperform every fixed configuration of the diversity-based MOEA. This can be explained by the fact that depending on the stage of the optimisation process, the most appropriate values for the controlled parameters are different, and the hybrid control scheme is able to detect these changes. Thus, the hybrid scheme provides better results since it is able to adapt the parameters of the diversity-based MOEA while it is being executed, whereas the fixed configurations are executed with the same parameter values during the whole run without considering any adaptation.

In contrast, note that for **eight** test cases—F1, F3, F8, F10, F12, F13, F15, and F19—some fixed configurations of the diversity-based MOEA obtained statistically significant better results than the hybrid parameter control scheme, indicating that, for these benchmarks, there exist some fixed values for the parameters that are suitable for the entirety of the optimisation process. For these cases, fixing the parameters to suitable values produces more robust behaviour in the diversity-based MOEA. Despite this fact, the results obtained by the hybrid control scheme were also competitive for this set of eight problems.

Lastly, it is worth pointing out that the benefits of the hybrid control approach are even greater if we consider the fact that 72 different configurations of the diversity-based MOEA were executed in order to look for the best set of values for its parameters, while only a single run of the hybrid control scheme was required. Therefore, besides the fact that the hybrid scheme obtained high quality results for most of the problems, the savings in computational resources and time required to produce good solutions were significant across all benchmarks when using it.

7.4.9 Analysis of the Diversity-based Survivor Selection Operator

This experiment, unlike the preceding one, does not deal with parameter control approaches. Instead, its main aim is to study the behaviour of the novel DCN-REF diversity-based survivor selection operator, which was proposed in Section 5.2. The F1–F11 benchmark functions with $D = 50$ decision variables were used to carry out two types of analyses. The first one focused on studying the average behaviour of the new proposed scheme, whereas the aim of the second one was to analyse the properties of said approach with respect to premature convergence. To perform the latter analysis, the behaviour of the worst executions obtained by the proposal were studied.

In the first analysis, two different variants of the diversity-based MOEA described in Section 7.2.2 were executed. Both variants apply the DCN-THR diversity-based objective. The main difference between them, however, lies in the fact that the first one does not make use of the DCN-REF survivor selection operator, while the second one does. In order to differentiate them, the first variant will be called *NO-REF*, while the second one will be referred to as *REF*. Different configurations of both variants were considered by defining different values for the parameter th of the DCN-THR diversity-based objective. In addition, for each benchmark function, every configuration was executed 100 times. The values for the remaining parameters of both schemes were fixed as follows:

- Population size N fixed to 100 individuals.
- Values for the threshold ratio th of the DCN-THR diversity-based objective: 0, 0.2, 0.4, 0.6, and 0.8.
- The PM operator was applied with probability $p_m = \frac{1}{D} = 0.02$.
- The SBX operator was applied with probability $p_c = 1$.
- The stopping criterion was fixed to $1 \cdot 10^5$ function evaluations.

Figure 7.3 shows, for each benchmark problem, the median of the original objective value attained by the approaches at the end of the executions. In cases where the differences between the results obtained with the NO-REF and the REF schemes were statistically significant, a vertical line joining their medians is shown. In most of the test cases—F2 and F3 are the exceptions—the REF model provided a higher median of the original objective value than the NO-REF scheme. Additionally, differences were statistically significant for most of the problems considered. The reason is that maintaining a proper diversity might reduce the convergence speed of the average case. In fact, the aim of maintaining a proper diversity is not to improve the convergence speed of the average case, but to prevent the occurrence of highly sub-optimal results in the worst executions.

In order to better understand the reasons why the novel proposal exhibits a sub-optimal behaviour, the probability of survival of the fittest individuals was experimentally calculated when using the REF and the NO-REF approaches using a threshold ratio $th = 0$. Table 7.26 shows the mean probability of survival of the 10 and 20 best individuals for both schemes. In every case, the probability of selecting the individuals with the best original objective values was higher for the NO-REF than for the REF approach. This means that the former focuses on the fittest individuals, while the latter performs a more diverse selection. As a result, for cases

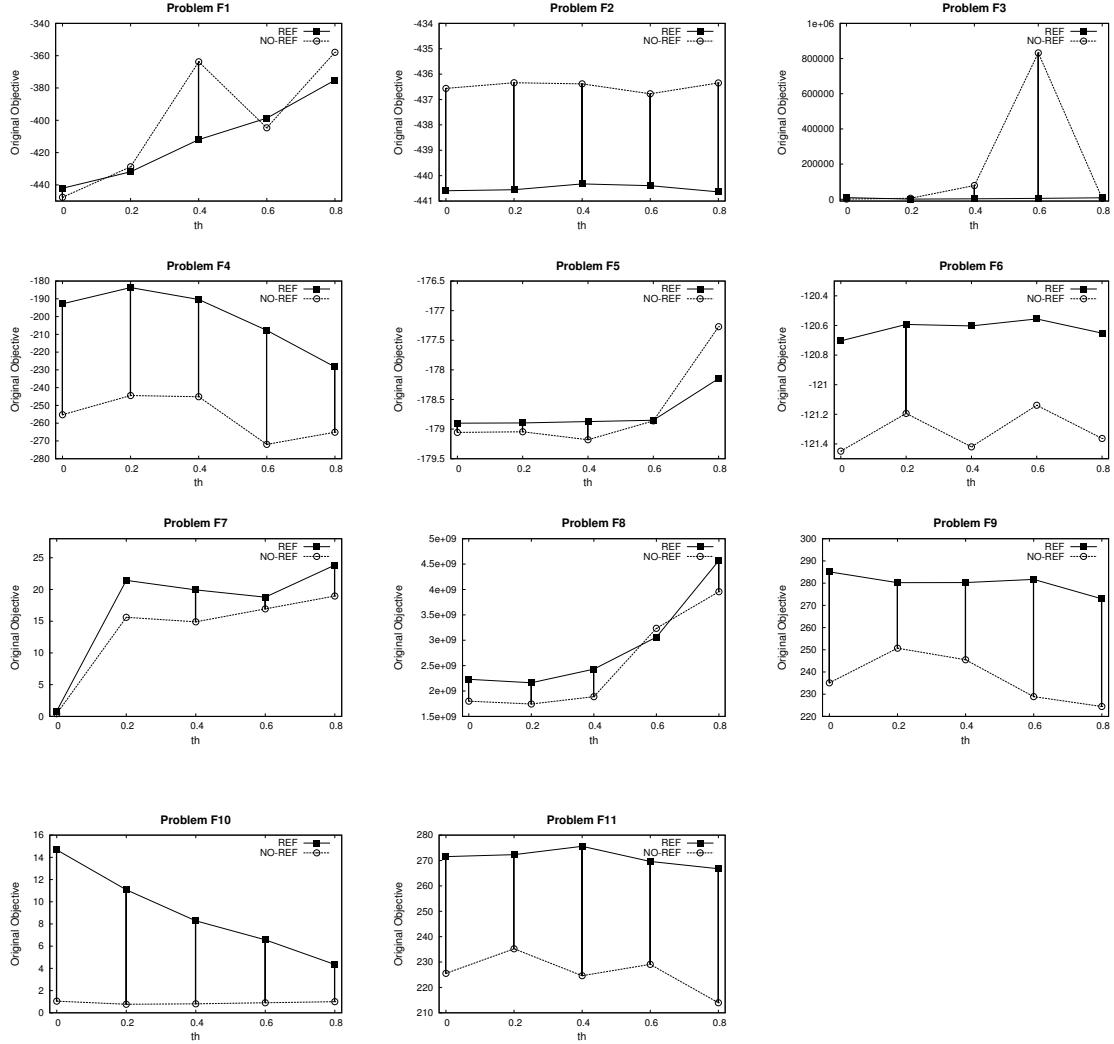


Figure 7.3: Median of the original objective value achieved for different threshold ratios

where premature convergence does not occur—the majority of the executions—the NO-REF scheme converges faster. We should note that other values for the threshold ratio th were considered in order to calculate the probabilities. For these cases, the NO-REF scheme also yielded higher probabilities than the REF model.

The above analysis demonstrated the superiority of the NO-REF scheme for the average case. In order to quantify its improvement, RLDs were calculated. Table 7.27 shows the percentage of evaluations that were saved by the NO-REF scheme in com-

Table 7.26: Probability of selecting the best individuals

	Prob. of 10 best ind.		Prob. of 20 best ind.	
	NO-REF	REF	NO-REF	REF
F1	90.01%	79.68%	85.02%	72.17%
F2	94.44%	87.73%	89.89%	83.37%
F3	91.05%	86.57%	85.10%	79.37%
F4	92.11%	91.04%	87.02%	86.35%
F5	89.97%	79.86%	84.03%	72.42%
F6	99.50%	97.12%	94.52%	92.93%
F7	90.84%	84.32%	84.80%	78.04%
F8	92.63%	90.24%	84.60%	85.62%
F9	93.16%	90.64%	90.46%	87.37%
F10	91.09%	82.52%	85.11%	74.03%
F11	92.75%	90.56%	88.44%	87.79%

Table 7.27: Evaluations saved by not using the diversity-based survivor selection scheme

	$th = 0$	$th = 0.2$	$th = 0.4$	$th = 0.6$	$th = 0.8$
F1	20%	-5%	-40%	5%	-17.5%
F2	-17.5%	-17%	-17.5%	-15%	-17.5%
F3	30%	-22.5%	-27.5%	-27.5%	5%
F4	32.5%	35%	32.5%	37.5%	27.5%
F5	25%	25%	30%	2.5%	-25%
F6	35%	40%	45%	40%	35%
F7	17.5%	25%	28.20%	7.89%	40%
F8	10%	10%	12.5%	-2.5%	12.5%
F9	35%	23.07%	27.5%	37.5%	30%
F10	45%	42.5%	37.5%	32.5%	22.5%
F11	30%	32.5%	37.5%	30%	32.5%

parison to the REF model using a 50% success rate. For each benchmark function and for each threshold value, the quality level was set as the highest median of the original objective value attained at the end of the executions by either of the two schemes. The negative values indicate that the REF scheme saved a larger number of evaluations. Finally, for each problem, the data belonging to the threshold ratio that achieved the lowest median of the original objective value are shown in bold. Note that, on average, the number of additional evaluations required by the REF approach was noticeable.

In the second analysis, the worst-case behaviour of the schemes considered above

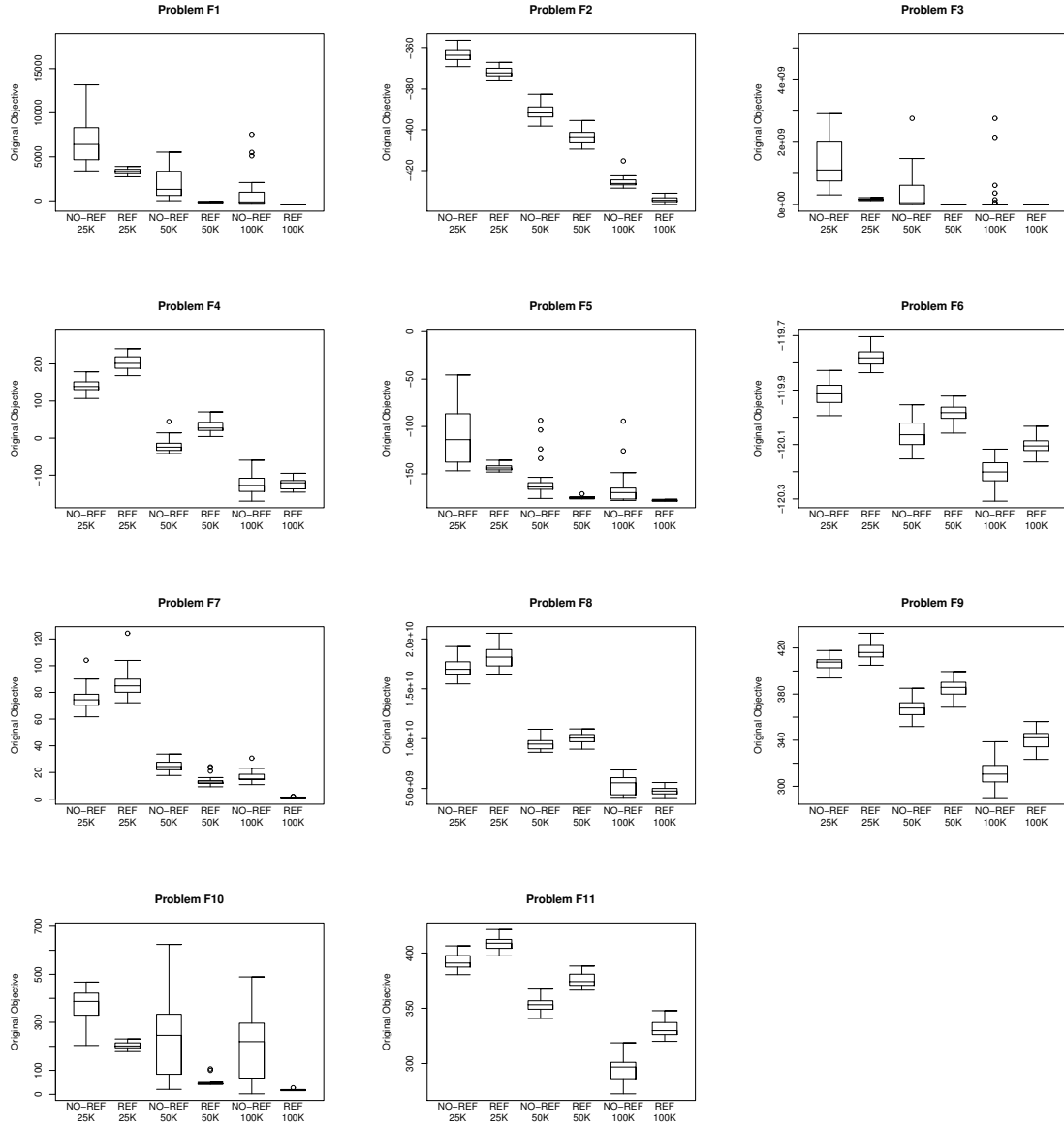


Figure 7.4: Box plots for the worst-behaved executions considering different stopping criteria

was studied. For each benchmark function, the threshold value th that yielded the best results in the preceding analysis was used. The remaining parameter values were the same as those previously used, although the two approaches were executed

Table 7.28: Statistical comparison considering different stopping criteria

	$2.5 \cdot 10^4$	$5 \cdot 10^4$	$1 \cdot 10^5$
F1	↑	↑	↑
F2	↑	↑	↑
F3	↑	↑	↑
F4	↓	↓	↔
F5	↑	↑	↑
F6	↓	↓	↓
F7	↓	↑	↑
F8	↓	↓	↑
F9	↓	↓	↓
F10	↑	↑	↑
F11	↓	↓	↓

3,000 times. The executions were arranged into groups of 100 executions, and the worst of each group—the one with the worst individual in terms of the original objective value achieved at the end of the executions—was stored. This yielded a set of 30 results for each approach and benchmark. The data represent the worst results obtained considering a probability equal to 1%.

Figure 7.4 shows the box plots of the original objective values obtained by the NO-REF and the REF approaches at the end of the worst executions. Three different stopping criteria were set: $2.5 \cdot 10^4$, $5 \cdot 10^4$, and $1 \cdot 10^5$ evaluations. Note that the advantages of using the REF model are clear for most of the problems. Table 7.28 shows the statistical comparison between the two schemes. The symbol ↑ is used to denote that differences between the models were statistically significant and that the REF approach obtained a lower median and mean of the original objective value than the NO-REF scheme. In cases where the opposite is true, the symbol ↓ is used. Lastly, for cases in which differences between both models were not statistically significant, the symbol ↔ is used. The REF scheme was statistically better than the NO-REF model in 7 test cases when the stopping criterion was set to $1 \cdot 10^5$ function evaluations.

A more in-depth analysis was performed for those problems where the REF approach did not provide any benefits in the latter study: F4, F6, F9, and F11. For these benchmark problems, convergence was not reached after carrying out $1 \cdot 10^5$ evaluations. The NO-REF and the REF schemes were executed considering a longer stopping criterion of $2.5 \cdot 10^6$ evaluations. For the F4 benchmark, the REF model statistically outperformed the NO-REF model. For the remaining functions, however, the results obtained by both models were similar. In fact, the statistical analyses

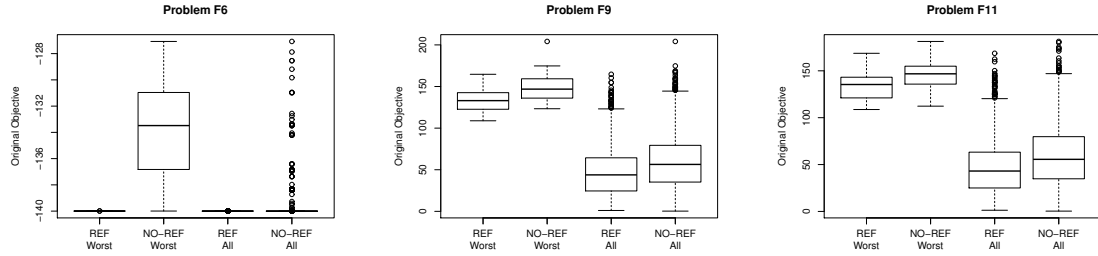


Figure 7.5: Box plots considering a population size equal to 500 individuals

showed that any differences between them were not significant.

Finally, with the aim of avoiding sub-optimal results in the functions F6, F9, and F11, both schemes were executed using a population size of 500 individuals. Figure 7.5 shows the box plots obtained with both approaches after $2.5 \cdot 10^6$ evaluations. The data tagged with the label *Worst* were produced by grouping the executions into sets of 100 items, and selecting the worst ones. Data tagged with the label *All* consider the complete set of 3,000 executions. Note that the REF model is clearly superior in every case. The sub-optimal results obtained with a population of 100 individuals was avoided by increasing it to 500 individuals. In fact, the statistical comparison demonstrated the superiority of the REF approach. Furthermore, the benefits of the latter apply not only to the worst-behaved executions, since the advantages are also clear when considering the complete set of executions.

Antenna Positioning Problem

This chapter aims to describe the solution to an optimisation problem involving the engineering of wireless telecommunication networks. This problem is called the *Antenna Positioning Problem (APP)*, although in the literature it is also known as the *Radio Network Design (RND)* or the *Base Station Transmitters Location (BSTL)* problem. The goal of the APP is to locate a set of Base Stations (BSs) or antennas at potential sites, so as to fulfil certain goals while satisfying specific constraints. Several objectives can be considered when designing a network. The most frequently used are minimising the number of antennas, maximising the amount of traffic carried by the network, maximising the quality of service, and/or maximising the coverage area. The APP has been analysed by many researchers and been shown to be an *NP*-hard optimisation problem [136]. The APP plays a major role in various engineering, industrial, and scientific applications because its outcome usually affects cost, profit, and other high-impact business performance metrics. As a result, the performance of the optimisation schemes considered has a direct effect on financial planning for industry.

The rest of this chapter is organised as follows. In Section 8.1 the mathematical formulation of the APP used in this thesis is described. Then, the different optimisation schemes defined for dealing with this problem are detailed in Section 8.2. Finally, Section 8.3 shows the experimental evaluation of the proposed optimisation schemes for several instances of the APP. The results obtained from the experiments are also discussed in this section.

8.1 Formal Definition

The APP is defined as the problem of identifying the infrastructure required to establish a wireless network. Several mathematical formulations for the APP have been proposed [5, 327]. In some cases, a network simulator that incorporates a wave propagation model is used to estimate the quality of a certain solution. Examples of these models are the *free space*, the *Okumura-Hata* and the *Walfisch-Ikegami* models [299]. In other cases, the canonical formulation of the APP is applied [348]. The main advantage of considering this canonical variant lies in the fact that the formulation is independent of the technology considered, and consequently new instances of the problem can be easily analysed. This canonical mathematical formulation of the APP is the one addressed throughout this thesis. It comprises two different objective functions:

- Maximising the coverage—*Coverage*—of a given geographical area.
- Minimising the amount of *BSs*—*Transmitters*—deployed.

Hence, the APP is defined as a MOP. A BS is a radio signal transmitting device that is able to irradiate some type of wave model. The area covered by a BS is called a cell. Furthermore, in this definition of the APP, BSs can only be situated at certain potential locations. In [5, 348], a single-objective version of the APP was proposed as an attempt to simplify the problem. This single-objective variant is considered herein. Its original objective function, which must be maximised, is defined as follows:

$$f(solution) = \frac{Coverage^\alpha}{Transmitters} \quad (8.1)$$

The practitioner must select a value for the parameter α shown in Equation 8.1. This parameter is tuned considering the importance given to the coverage with respect to the number of BSs deployed. In this dissertation, $\alpha = 2$ is used, since this value has been widely applied in previous research [5, 348].

The geographical area G where a network is deployed is divided into a finite number of points or locations. Tam_x and Tam_y are the number of vertical and horizontal sub-divisions, respectively. They are selected by experts in the communications field depending on various features of the region and of the transmitters. $U = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ is the set of locations where a BS can be deployed. $U[i]$ refers to the location i . The coordinates x and y belonging to the location i are labelled $U[i]_x$ and $U[i]_y$, respectively. When a BS is located at position i , its

corresponding cell is covered. This cell is termed $C[i]$. In the canonical mathematical formulation, an isotropic radiating model is assumed for each cell. The set $P = \{(\Delta x_1, \Delta y_1), (\Delta x_2, \Delta y_2), \dots, (\Delta x_m, \Delta y_m)\}$ determines the locations covered by a BS. Thus, if the BS i is deployed, the locations covered are given by the set $C[i] = \{(U[i]_x + \Delta x_1, U[i]_y + \Delta y_1), (U[i]_x + \Delta x_2, U[i]_y + \Delta y_2), \dots, (U[i]_x + \Delta x_m, U[i]_y + \Delta y_m)\}$. If $B = [b_0, b_1, \dots, b_n]$ is the binary vector describing the BSs deployed, the following equations hold for the APP:

$$Transmitters = \sum_{i=0}^n b_i \quad Coverage = \sum_{i=0}^{tam_x} \sum_{j=0}^{tam_y} covered(i, j) \quad (8.2)$$

where:

$$covered(x, y) = \begin{cases} 1 & \text{If } \exists i \mid \{(b_i = 1) \wedge ((x, y) \in C[i])\} \\ 0 & \text{Otherwise} \end{cases} \quad (8.3)$$

8.2 Optimisation Schemes

In this section we define the different optimisation schemes that are applied herein for dealing with the APP. Specifically, the optimisation methods considered are a single-objective ILS algorithm, several diversity-based MOEAs, as well as different parallel homogeneous island-based models.

8.2.1 Single-objective Iterated Local Search

The main idea behind an *Iterated Local Search (ILS)* [218] is that a sequence of solutions iteratively built by a certain heuristic, usually in the form of a local search procedure, might provide better solutions than those obtained by different independent executions of this heuristic approach. This idea was initially proposed in [29] but it has been rediscovered by many authors, resulting in many different names like *Iterated Descent* [28], *Large-step Markov Chain* [235], and *Chained Local Optimisation* [234]. The two main features of an ILS are the following:

- There exists a single chain that must be followed.
- The intensification is performed in a reduced space defined by the output of a black-box heuristic, which is usually based on a local search procedure.

Algorithm 8 Generic pseudocode for an iterated local search

```
1:  $s_0 = \text{generateInitialSolution}();$ 
2:  $s = \text{localSearch}(s_0);$ 
3:  $\text{updateBestSolution}(s)$ 
4: while (not stopping criterion) do
5:    $s' = \text{perturbation}(s, \text{history})$ 
6:    $s'' = \text{localSearch}(s')$ 
7:    $s = \text{acceptanceCriterion}(s, s'', \text{history})$ 
8:    $\text{updateBestSolution}(s'')$ 
9: end while
```

ILS is a trajectory-based meta-heuristic that iteratively applies a local search procedure to the current solution. Additionally, it usually outperforms “*simpler*” local search methods since it is able to escape from local optima and continue searching for better potential solutions.

Algorithm 8 shows the generic pseudocode for an ILS. At the beginning of the algorithm—line 1—an initial solution s_0 is generated. Afterwards, a new solution s is produced—line 2—by applying the local search procedure to s_0 , and the best solution is updated if required—line 3. Then, until a given stopping criterion is satisfied—line 4—a set of steps is repeated. First, a diversification step is applied by perturbing s to obtain s' —line 5. Second, an intensification step is performed—line 6—over s' by applying the local search procedure so as to obtain a new solution s'' . Then, if s'' satisfies a given acceptance criterion, it replaces s and the next iteration is carried out starting from this new solution—line 7. The best solution found is updated—line 8—at every iteration if needed. Lastly, when the stopping criterion is satisfied, the best solution found is returned.

In order to obtain a particular configuration of an ILS, some components have to be specified. For the particular case of the APP, they are the following:

- *Generation of the initial solution.* In this thesis, the following steps are carried out to produce the initial solution. First, the grid representing the terrain is divided into a set of sub-grids or windows. All windows have size $W \times W$, where W is randomly selected from the range $[(R - 9) \cdot 2, R \cdot 2]$, where R is the coverage radius of the BSs. Then, the centre of every sub-grid is calculated and every coordinate is shifted considering random values in the range $[-5, 5]$. The nearest BS to every computed position is inserted into the current solution.
- *Local search procedure.* In this dissertation, a hill climbing method is used as the local search procedure. The neighbourhood of a solution is defined as

follows:

1. For every available location that has not been considered in the solution, a neighbour that includes a BS in this location is generated.
2. For every available location where a BS has been placed, a neighbour that does not consider this BS in the solution is generated.
3. For every available location where a BS has been placed, a neighbour that replaces this BS by the nearest one is generated.

During the local search, the complete neighbourhood is generated. The best neighbour is selected from among the different neighbours generated. The local search finishes when it reaches a local optimum or when it iterates a maximum number of ms times.

- *Perturbation strategy.* In this thesis, a random perturbation mechanism is applied. It selects, on the one hand, a set of deployed BSs to be removed from the current solution, and on the other hand, a set of locations where to include an extra BS. The number of BSs that are removed and inserted is randomly determined by a Gaussian distribution of mean $\mu * strength$ and standard deviation σ . The BSs that are removed and inserted are randomly selected from among the available locations using a uniform distribution. Once these modifications have been introduced into the solution, a final step is performed. For every location, the fitness of the solution including the BS—if it is not considered—or excluding the BS—if it is—is computed and the best choice is selected as the final solution. The ILS controls the value of the parameter *strength*. Initially, it is set to 1. If the best solution is not improved during the last k iterations, *strength* is increased. Finally, if the search remains trapped in a local optimum after *strength* is increased t times, the algorithm is restarted from a new generated initial solution. It is important to note that the perturbation scheme might make use of historical data on the execution, as Algorithm 8 shows.
- *Acceptance criterion.* This component is responsible for deciding which solution between s and s'' is going to be used in the next iteration of the ILS. In this dissertation, a predefined rule that involves selecting the fittest of both solutions is always applied. This decision, however, might be also based on historical information concerning the optimisation process.

In order to completely define the specific ILS applied to the APP, we should note that individuals were encoded through binary strings with n items, where n is the

number of candidate BSs. Every gene determines whether or not the corresponding BS is deployed in the network.

8.2.2 Diversity-based Multi-Objective Evolutionary Algorithms

The novel diversity-based MOEA, which is used here to tackle the APP, is based on the NSGA-II—Section 2.4.1. Since a diversity-based MOEA is applied, an additional diversity-based objective function must be considered together with the original objective function of the APP defined in Equation 8.1. Different encoding-independent and genotypic diversity-based objectives were taken into account. In particular, the diversity-based objectives tested were *time stamp*, *random*, *inversion*, ADI, DBI, and DCN—Section 3.1. Moreover, the diversity-based objective with parameters DBI-THR—Section 5.1—was also applied.

So as to completely define this diversity-based MOEA, it is worth pointing out that individuals were represented by means of binary strings with n items, where n is the number of candidate BSs. Every gene determines whether the corresponding BS is deployed or not. Additionally, the mutation operator used was the bit flip mutation—Section 2.2.3. Finally, the crossover operator employed was the *Geographic Crossover (GC)* [327], which exchanges the BSs that are located within a given radius r from a randomly selected BS.

8.2.3 Parallel Homogeneous Island-based Models

The parallelisation of the diversity-based MOEA described in the previous section is also considered herein through the use of a homogeneous island-based model—Section 2.6.1. Recall that in a homogeneous island-based model, every island executes the same algorithm with the same parameterisation. Therefore, in this case, the same parameterisation of the diversity-based MOEA introduced in the preceding section is executed by every island.

Different migration stages were tested with this homogeneous island-based model. These migration stages were obtained by combining two different migration schemes with two different replacement schemes, thus yielding four different migration stages. Particularly, the migration schemes were ELI-M and RND-M, whereas the replacement schemes were ELI-R and RND-R—Section 2.6.1. So as to differentiate the migration stages, the nomenclature *migration-replacement* is employed. Hence, for

instance, the migration stage *ELI-M-RND-R* applies the migration scheme ELI-M and the replacement scheme RND-R. Lastly, the migration topology ALL—Section 2.6.1—was used to define the four migration stages. With this topology, when a generation of the algorithm located on a certain island finishes, a set number of individuals—migration rate—is sent from the island to the remaining ones as determined by a given migration probability.

8.3 Experimental Evaluation and Discussion

In this section we present the different experiments conducted on the aforementioned optimisation schemes using different instances of the APP.

Experimental Method. The different optimisation schemes were implemented using METCO. Tests were run on a Debian GNU/Linux computer with four AMD ® Opteron TM processors (model number 6164HE) at 1.7 GHz and 64 Gb RAM. The compiler was the GCC 4.4.5. Communications among different islands of the homogeneous island-based model were implemented asynchronously using the MPICH library. Since every experiment used stochastic algorithms, every execution was repeated 32 times. As a result, the comparisons were performed by applying the statistical analysis detailed in Section 1.2.6.

APP Instances. Studies were conducted using two different instances. The first was a real world-sized problem instance defined by the geographical layout of the city of Malaga in Spain, which covers an urban area of 27.2 Km^2 . The terrain was modelled using a 450×300 grid, where each point represents a surface area of approximately $15 \times 15 \text{ m}^2$. The dataset contains $n = 1000$ candidate sites for the BSs. The second instance, which was artificially generated, represents a hypothetical city. In this case, the terrain was modelled using a 287×287 grid. The dataset contains $n = 349$ candidate sites for the BSs.

8.3.1 On the Comparison of Diversity-based Objectives

In this first experiment the different diversity-based objective functions are compared by using the diversity-based MOEA presented in Section 8.2.2 in combination with

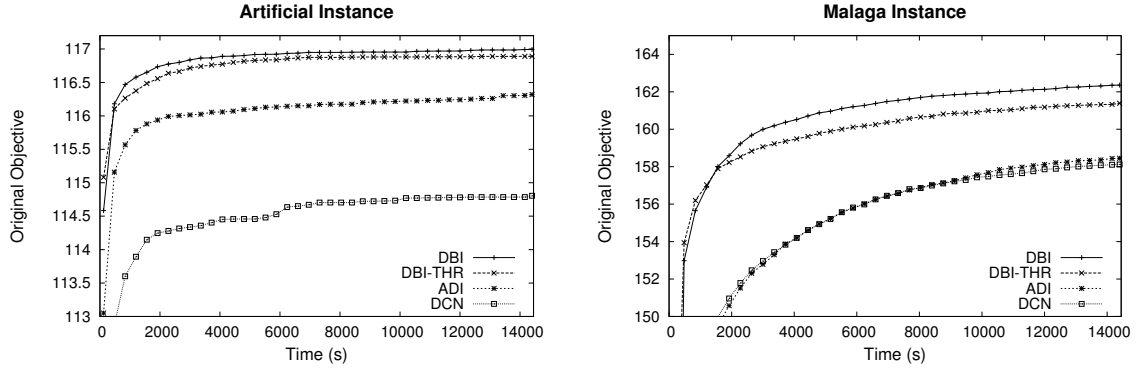


Figure 8.1: Evolution of the mean original objective value for different diversity-based objective functions

every diversity-based objective considered. The parameterisation of the diversity-based MOEA was the following:

- Population size $N = 100$ (Malaga instance) and $N = 50$ (artificial instance).
- Mutation rate $p_m = \frac{1}{n}$, with n being the number of candidate BSs.
- Crossover rate $p_c = 1$.
- Radius of the GC operator $r = 30$.
- Diversity-based objectives *time stamp*, *random*, *inversion*, ADI, DBI, DCN, and DBI-THR.
- Threshold value $th = 0.7$ for the diversity-based objective DBI-THR.
- Stopping criterion set to 4 hours.

This analysis was carried out in terms of the performance achieved in the original objective. Hence, the evolution of the mean original objective value achieved by the diversity-based MOEA for every diversity-based objective function considered is shown in Figure 8.1. Only the results for the best-behaved diversity-based objective functions are shown. Firstly, note that the information on diversity-based objectives built upon an encoding-independent measure is not shown. The reason is that the *time stamp*, *random*, and *inversion* approaches yielded poor results with respect to the schemes that make use of measures in the genotypic space. Considering the genotypic diversity-based objectives, the worst results were provided by the ADI and DCN schemes for both instances. In contrast, the best results were obtained by the DBI method, with the differences being even more noticeable in the case of the

Table 8.1: Statistical comparison among different diversity-based objectives

	Artificial Instance				Malaga Instance			
	DBI	DBI-THR	ADI	DCN	DBI	DBI-THR	ADI	DCN
DBI	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
DBI-THR	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\downarrow	\leftrightarrow	\uparrow	\uparrow
ADI	\downarrow	\downarrow	\leftrightarrow	\uparrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow
DCN	\downarrow	\downarrow	\downarrow	\leftrightarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow

Malaga instance. The DBI-THR diversity-based objective also achieved high-quality solutions. The expected behaviour was for the DBI-THR scheme to outperform the DBI approach. This was not so, however, and the reason why might be the improper selection of the parameter th . This parameter was selected by performing a preliminary study in which different configurations of the diversity-based MOEA and the DBI-THR approach were executed with fixed values for this parameter. The best results were achieved by the configuration with the value $th = 0.7$. The most appropriate value for the parameter th can change, however, depending on the stage of the optimisation procedure, so keeping a fixed value during the whole execution could be counterproductive. Since in this first experiment the main goal was to compare different diversity-based objectives and not to look for the most suitable value for the parameter th of the DBI-THR scheme, no special consideration was given to this issue. Consequently, more attention should be paid to the proper selection of this parameter for the particular case of the APP. Despite this, other analyses carried out in this dissertation in the field of different optimisation problems revealed that applying certain parameter control methods to this parameter provides high-quality results that are even better than those given by configurations of the corresponding optimisation schemes executed with fixed values for th .

In order to perform a statistical comparison, Table 8.1 shows the statistical differences among the best-behaved diversity-based objective functions. Particularly, the table shows whether the scheme located in a given row is statistically better (\uparrow), not different (\leftrightarrow), or worse (\downarrow) than the corresponding scheme situated in a certain column. This table confirms that the conclusions extracted from the data shown in Figure 8.1 are valid from a statistical point of view. In the case of the artificial instance, the DBI and DBI-THR approaches were statistically better than the ADI and DCN schemes, and the ADI scheme outperformed the DCN approach. However, the DBI approach did not exhibit any statistically significant differences from the DBI-THR approach. Considering the Malaga instance, the DBI diversity-based objective was statistically better than every other scheme, while the DBI-THR method outperformed the ADI and DCN approaches, which did not present any significant

differences between them. In summary, the statistical analyses conclude that the DBI diversity-based objective was able to provide the best results for both instances, as was stated earlier.

At this point, it is also important to remark that the diversity-based objective functions proposed as the best ones in [45, 330] do not match the best-behaved diversity-based objective functions studied in this experiment. It is also true that other optimisation problems were tackled in those research papers. Consequently, the proper diversity-based objective depends on the problem and/or even on the instance in question. Finally, it is worth noting that for both instances considered, the original objective value was still increasing after 4 hours of execution, demonstrating the ability of the proposed diversity-based MOEA to avoid the problem of premature convergence.

8.3.2 Diversity-based Multi-Objective Evolutionary Algorithms vs. Single-objective Iterated Local Search

The results obtained by the diversity-based MOEA in the previous experiment were better than those provided by the single-objective optimisers introduced in [242]. These single-objective optimisation schemes do not make use of problem-dependent information. However, the solutions given by the diversity-based MOEA did not achieve the same quality as that attained by the problem-dependent approaches proposed in [242]. The aim of this second experiment is therefore twofold. Firstly, to study whether the diversity-based MOEA is able to yield, in the long term, the same high-quality solutions obtained by the best-behaved problem-dependent approach proposed in [242]. Secondly, to quantify the run-time behaviour of the diversity-based MOEA with respect to the problem-dependent approach.

In order to carry out this experiment, the diversity-based MOEA combined with the DBI diversity-based objective was compared against the single-objective algorithm that returned the best results in [242], which is the ILS described in Section 8.2.1. In the case of the diversity-based MOEA, the parameterisation was the one used in the previous experiment. The parameterisation of the ILS was the one applied in [242]. Particularly, the values $R = 30$, $\mu = 3$, $\sigma = 1$, $ms = 100$, $k = 250$, and $t = 2$ were used. For both approaches, a stopping criterion equal to 24 hours of execution was fixed.

The run-time behaviour of both approaches was analysed through the application of RLDs, which were described in Section 1.2.6. Two different quality levels were

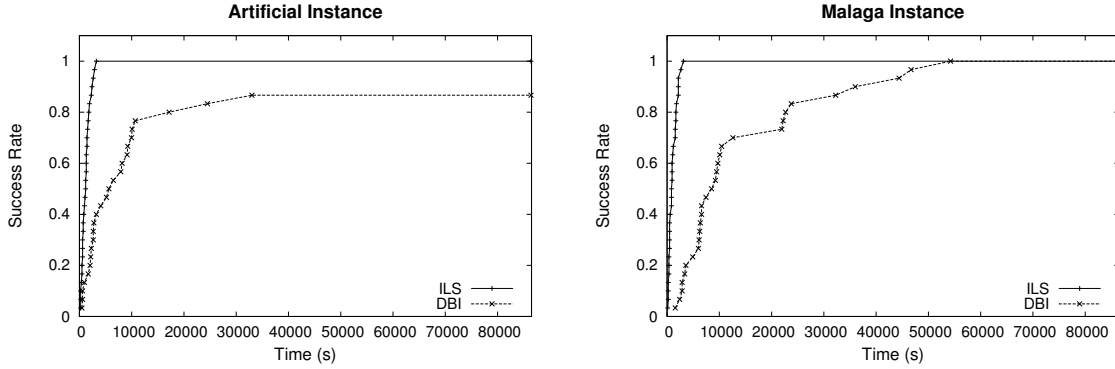


Figure 8.2: Run-length distributions for quality level L1

considered so as to compute the RLDs. These quality levels were defined as the original objective value belonging to the worst individual in the population given by the ILS considering 1 and 2 hours of execution. They are referred to as L1 and L2, respectively. Figure 8.2 shows the RLDs for the diversity-based MOEA and the ILS for quality level L1. The same information is shown in Figure 8.3, but for quality level L2. Note that, since ILS is an approach that considers a large amount of problem-dependent information, all executions—a 100% success rate—were able to attain both fixed quality levels very fast for both instances. In contrast, the diversity-based MOEA showed a slower convergence, but a considerable amount of its executions were able to yield the same high-quality results. This fact was even more noticeable for the Malaga instance, where all executions of the diversity-based MOEA achieved both fixed quality levels.

So as to quantify the slower convergence of the diversity-based MOEA, Table 8.2 shows, for both instances, the amount of additional resources—time—that this approach invested in comparison to the ILS to attain quality levels L1 and L2, considering different success rates. For instance, for a 50% success rate, the diversity-based scheme invested 4.70 times more resources than the ILS to achieve quality level L1 in the case of the artificial instance. If quality level L2 is considered, 14.75 times more resources were required by the diversity-based MOEA. It can be observed that for this particular instance the higher the success rate the greater the amount of additional time that the diversity-based MOEA required with respect to the ILS to attain the two quality levels. In the case of the Malaga instance, the diversity-based MOEA needed 10.14 times more resources than the ILS to reach quality level L1, while for L2, the factor was equal to 10.12 at a 50% success rate. For this instance, the higher the success rate the lower the amount of additional time that

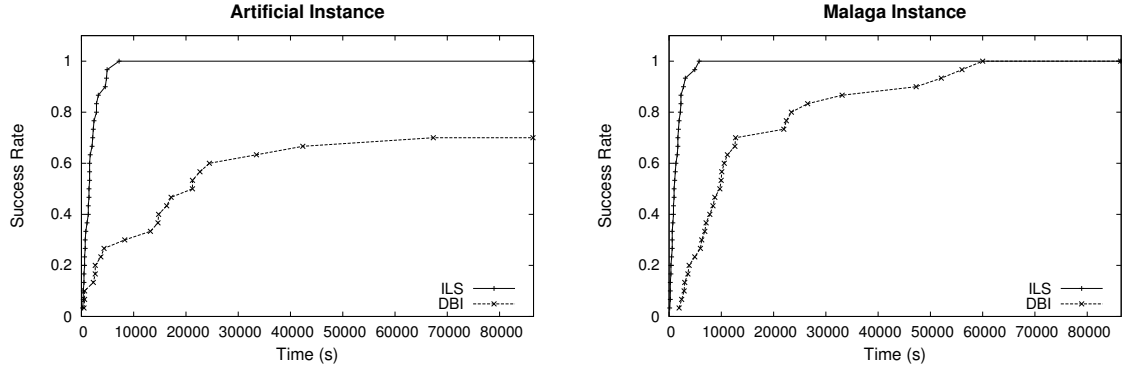


Figure 8.3: Run-length distributions for quality level L2

Table 8.2: Amount of additional resources invested by the diversity-based multi-objective evolutionary algorithm with respect to the iterated local search

Success Rate	50%		60%		70%	
Quality Level	L1	L2	L1	L2	L1	L2
Artificial Instance	4.70	14.75	6.18	15.69	6.91	31.16
Malaga Instance	10.14	10.12	10.12	8.00	8.07	7.57

the diversity-based MOEA needed to reach the two quality levels in comparison to the ILS. Therefore, the diversity-based scheme seemed to converge somewhat faster for the Malaga instance than for the artificial instance.

Finally, we should note that although the diversity-based MOEA was slower to converge than the ILS, in the long term, the former was able to obtain the same high-quality solutions as the latter, thus mitigating the problem of premature convergence. In fact, executing the diversity-based MOEA together with the DBI approach for 24 hours improved on the best-known solutions published in [242]. Furthermore, note that the diversity-based MOEA does not use problem-dependent information, whereas a significant number of components belonging to the ILS applied herein were specifically designed to deal with the APP. Therefore, the contribution of the diversity-based MOEA is even greater if this last fact is considered.

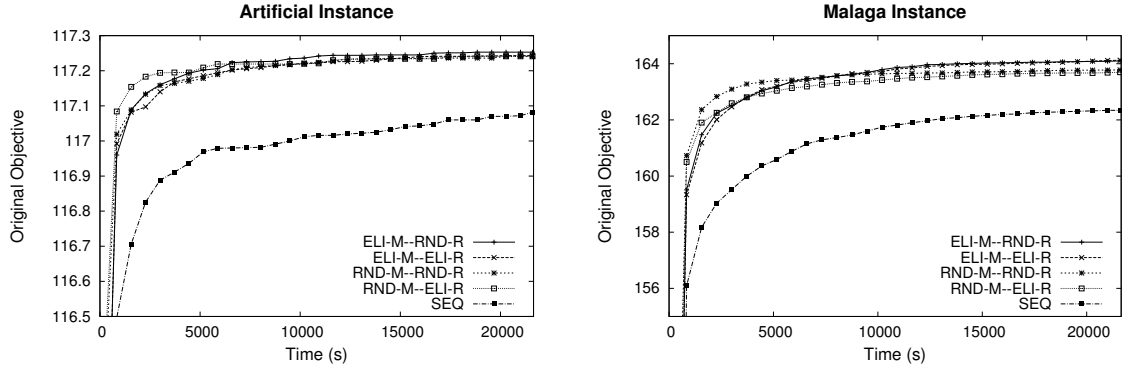


Figure 8.4: Evolution of the mean original objective value for the homogeneous island-based models executed with 4 islands

8.3.3 Analysing the Robustness of the Homogeneous Island-based Model

In the previous section, we showed that the diversity-based MOEA was able to attain the same high-quality solutions as those provided by the best single-objective approach available in the literature. However, due to its slow convergence, the time required by the diversity-based MOEA was considerably longer than the time invested by the single-objective method. In order to reduce this convergence time, this third experiment relies on a homogeneous island-based model to parallelise the diversity-based MOEA. Furthermore, the robustness of the homogeneous island-based model in terms of the applied migration stage is also analysed.

To this end, the homogeneous island-based model described in Section 8.2.3 was executed considering 4 islands and a stopping criterion of 6 hours. Additionally, as was stated earlier, four different migration stages were tested. For all them, the migration rate was fixed to 1 individual, whereas the migration probability was set to 0.01. Since a homogeneous island-based model was considered, every island executed the best-behaved configuration of the diversity-based MOEA applied in previous experiments, i.e. the configuration that makes use of the DBI approach. The values for the remaining parameters of the diversity-based MOEA were the same as those used in the first experiment—Section 8.3.1.

Figure 8.4 shows, for both instances, the evolution of the mean original objective value achieved by the homogeneous island-based model executed with each of the four different migration stages. The sequential version of the diversity-based MOEA combined with the DBI approach, which is referred to as SEQ, is also shown in

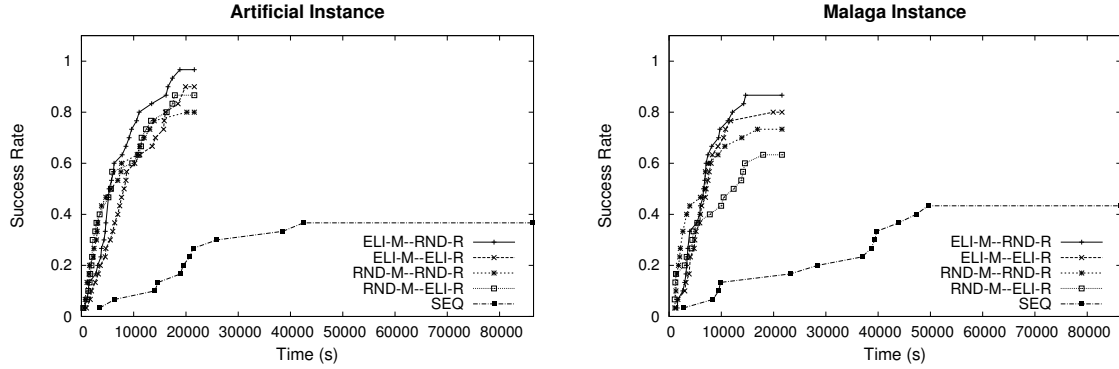


Figure 8.5: Run-length distributions for the homogeneous island-based models executed with 4 islands

this figure. For both instances, note that the parallel models were able to clearly improve the results obtained by the sequential strategy. With regard to the four migration stages, they all yielded similar results. In fact, the statistical analysis revealed that there were no significant differences between them. This thus proves the robustness of the homogeneous island-based model, since high-quality results were obtained regardless of the migration stage. We should mention, however, that for both instances the highest mean original objective value was obtained by the homogeneous island-based model executed with the ELI-M-RND-R migration stage.

Given that the parallel island-based models made use of a larger amount of computational resources than the sequential diversity-based MOEA, the gap in the improvement between the two approaches should be quantified. To do so, RLDs were calculated for the four homogeneous island-based models and for the sequential strategy. In the case of the artificial instance, since every parallel island-based model was able to achieve the best-known original objective value, this value was selected as the quality level. In the case of the Malaga instance, the variance in the results was higher than for the artificial instance. Hence, if the best-known original objective value for this instance had been fixed as the quality level, very low success rates would have been obtained. As a result, the quality level was selected as the original objective value that allowed every homogeneous island-based model to attain a success rate of at least 60%.

Figure 8.5 shows, for both instances, the RLDs for the homogeneous island-based model executed with each of the four migration stages, and for the sequential variant of the diversity-based MOEA. In the case of the sequential model, a maximum execution time of 24 hours was used. For the parallel schemes, the maximum execution

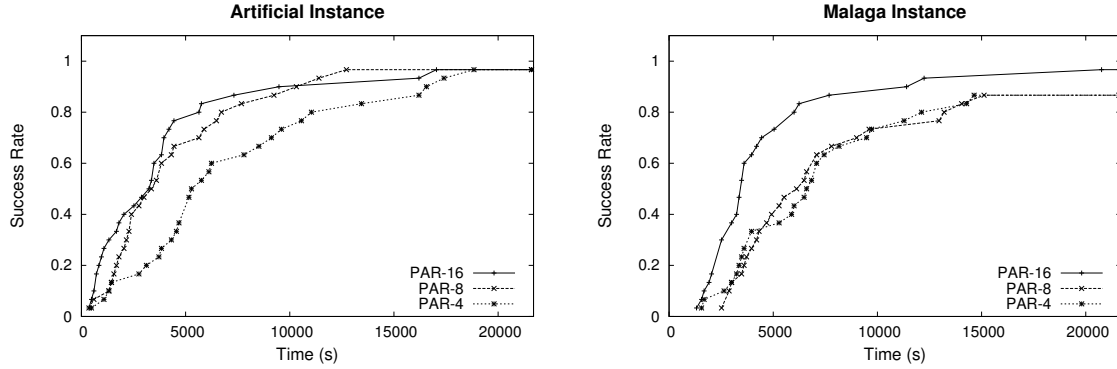


Figure 8.6: Run-length distributions for the homogeneous island-based models executed with 4, 8, and 16 islands

time was set to 6 hours. Observing the RLDs, the superiority of the homogeneous island-based models with respect to the sequential version of the diversity-based MOEA is confirmed. The main reason for this behaviour might be the ability of the island-based model to explore the search space more efficiently, and consequently to deal with stagnation in local optima. With regard to the different migration stages, there were no significant differences in their run-times, though if a high success rate is considered, the best performance was exhibited by the ELI-M-RND-R migration stage.

8.3.4 Analysing the Scalability of the Homogeneous Island-based Model

The fourth experiment aims to analyse the scalability of the homogeneous island-based model. In order to conduct this experiment, the homogeneous island-based model combined with the ELI-M-RND-R approach—from now on referred to as PAR-4—which was the best-behaved migration stage in the previous analysis, was executed with 8 islands—PAR-8—and 16 islands—PAR-16.

Figure 8.6 shows the RLDs of the aforementioned parallel schemes for both instances and a maximum execution time of 6 hours. Speedup factors with respect to PAR-4 were calculated for different success rates ranging from 25% to 75%. In the case of the artificial instance, the speedup factors corresponding to PAR-8 ranged from 1.57 to 1.88. In the case of the homogeneous island-based model PAR-16, the speedup factors ranged from 1.62 to 3.57. Although these speedup factors show

the benefits of adding a larger amount of islands, some unusual scalability issues were detected for the PAR-16 scheme. In fact, PAR-8 and PAR-16 provided similar speedup factors when considering success rates ranging from 40% to 60%. However, when other success rates were used, the PAR-16 approach achieved the highest speedup factors, thus demonstrating its advantages. Some scalability problems were also exhibited by the Malaga instance. For example, no benefit was obtained by executing the homogeneous island-based model with 8 islands—PAR-8. In fact, we can see that the RLDs belonging to PAR-4 and PAR-8 are quite similar. However, when 16 islands were used—PAR-16—the speedup factors increased significantly. In particular, the speedup factors given by PAR-16 in comparison to PAR-4 ranged from 1.42 to 1.9.

Frequency Assignment Problem

The *Frequency Assignment Problem (FAP)* is a well-known *NP*-hard combinatorial optimisation problem of great importance to the design of communication networks [1]. Several variants of the FAP have emerged based on different wireless technologies. Specifically, the FAP is one of the crucial issues in the design of Global System for Mobile Communications (GSM) networks [254]. This problem is also known as the *Automatic Frequency Planning (AFP) problem* and the *Channel Assignment Problem (CAP)* in the literature. Although the FAP has led to many different mathematical and engineering models, all of them share two common features:

- A set of antennas must be assigned frequencies such that data transmissions between the two end points of each connection are possible.
- Depending on the frequencies assigned to the antennas, they might interfere with one another, resulting in loss of signal quality.

This dissertation focuses on the FAP that arises in the design of GSM networks. In this particular case, the available frequency band is slotted into channels that have to be allocated to the elementary transceivers installed in the base stations of the network. In GSM networks, the assignment of frequencies is a difficult design task because the usable radio spectrum is very scarce and frequencies have to be reused throughout the network, thus resulting in some degree of interference. The main aim of the designer is therefore to minimise the interferences in the network, i.e. to minimise the loss of signal quality.

The FAP emerges in different network environments and involves different objectives, features, and constraints. Therefore, several mathematical formulations have

been defined for dealing with this problem [1]. In recent years, the basic FAP formulation has been widely extended in order to address real-world issues [195]. Most of the FAP models differ in the way that the interference is measured. Computing the level of interference is an arduous task that depends on the channels, the radio signals and many other features of the environment. The quantification of the interference results in an *interference matrix*, which is usually denoted by M . Some theoretical methods for computing M have been proposed [1]. Theoretical methods offer the advantage of allowing for new instances to be computed with less effort. However, these methods ignore some features of the environment, which makes it difficult to determine the consequences that their use might involve. As a result, in other research works [195], extensive network measurements are performed in order to calculate the interference matrix, which relies on more accurate values, resulting in more realistic frequency plans. Using this method to calculate the interference matrix associated with a new network is, however, very expensive.

The FAP can be classified based on different criteria, such as the size of the frequency spectrum, its goals, and the specific technological constraints, among others. In [1], three main FAP models were described: the *Minimum Order Frequency Assignment Problem (MOFAP)*, the *Minimum Span Frequency Assignment Problem (MSFAP)*, and the *Minimum Interference Frequency Assignment Problem (MIFAP)*. These models have appeared in the literature over time as demanded by the working conditions imposed by technology, national regulations, and markets. The MOFAP is aimed at reducing the number of frequencies used in a given network. It assumes that different frequencies do not interfere with each other. The MSFAP focuses on searching for an assignment of frequencies that minimises the difference between the largest and the smallest frequencies assigned, i.e. the frequency span. It assumes that frequencies are assigned by regulators in continuous slots. Finally, the MIFAP tries to minimise a measure of the overall interference in the network. The MIFAP is the model that has been most frequently addressed in the recent literature, mainly due to its direct applicability to the resolution of large instances corresponding to real-world GSM networks [38]. The version of the FAP considered in this dissertation [222] follows the MIFAP model. In this particular case, the interference matrix includes the interference among cells by giving the entire probability distribution of the *Carrier-to-Interference (C/I) ratio* [342]. This information is directly imported from real world GSM frequency planning as is currently conducted in the industry, instead of being generated in a computer by sampling random variables. As a result, this MIFAP variant allows not only the computation of high performance frequency plans, but also the prediction of the Quality of Service (QoS). Indeed, both the definition of the interference matrix and the subsequent computations car-

ried out to obtain the cost values are motivated by real-world GSM networks, since they are related to the computation of the *Bit Error Rate (BER)* performance of *Gaussian Minimum Shift Keying (GMSK)*, the modulation scheme used in GSM networks [307]. Finally, it is important to remark that besides these three main models of the FAP, other variants have been proposed [1].

Several optimisation methods have been applied to solve the various aforementioned versions of the FAP. Among them, some exact algorithms have been proposed [12, 118, 229]. However, since most of the FAP models involve the resolution of an *NP*-hard optimisation problem [148], approximate algorithms are mandatory when considering instances that represent vast networks [1]. Meta-heuristics have been shown to yield very accurate solutions to the FAP [7]. Specifically, MAs have been successfully applied to this problem [164, 184, 238].

In [222], an ACO algorithm was adapted to the FAP variant considered herein. Furthermore, a comparative study using a large set of meta-heuristics, including this ACO algorithm, was conducted in [223]. It included population-based and trajectory-based meta-heuristics. This research revealed the good performance of a MA that includes a method to increase the population size when stagnation is detected. This algorithm, which is called *Evolutionary Algorithm with Increasing Population Size (EAIPS)*, is a modified version of an EA combined with a (1+1) survivor selection operator and an individual learning procedure specifically designed to deal with the FAP formulation addressed in this thesis. This optimisation scheme made it possible to obtain high-quality frequency plans for different network instances. However, before solving a particular instance, the algorithm's parameters must first be set. In order to more quickly obtain high-quality solutions, several parallel approaches have also been applied. In [298], a parallel hyper-heuristic that makes use of the EAIPS was proposed. The hyper-heuristic was applied with the aim of avoiding the parameter setting of the memetic approach. It made it possible to obtain high-quality frequency plans in a single run. However, the quality of the frequency plans obtained was not as high as that provided by the grid-based GA proposed in [224]. Note that this approach used approximately 300 processors.

The rest of this chapter is organised as follows. In Section 9.1 the mathematical formulation of the FAP variant considered herein is described. Then, the different optimisation schemes defined for dealing with this problem are presented in Section 9.2. Afterwards, the control approaches that are proposed to adapt the parameters of said optimisation schemes are introduced in Section 9.3. Finally, Section 9.4 details the experimental evaluation conducted using the proposed optimisation schemes and parameter control approaches on several instances of the FAP.

9.1 Formal Definition

The FAP formulation applied in this dissertation was proposed in [222]. Let $T = \{t_1, t_2, \dots, t_n\}$ be a set of n *transceivers*, and let $F_i = \{f_{i1}, \dots, f_{ik_i}\} \subset \mathbb{N}$ be the set of valid frequencies that can be assigned to a transceiver $t_i \in T$, $i = 1, \dots, n$. Note that k_i —the cardinality of F_i —is not necessarily the same for every transceiver. Furthermore, let $S = \{s_1, s_2, \dots, s_m\}$ be a set of given sectors—or cells—of cardinality m . Each transceiver $t_i \in T$ is installed in exactly one of the m sectors. From now on we denote the sector in which a transceiver t_i is installed by $s(t_i) \in S$. Finally, the matrix $M = \{(\mu_{ij}, \sigma_{ij})\}_{m \times m}$ is denoted as the interference matrix. The two elements μ_{ij} and σ_{ij} of a matrix entry $M(i, j) = (\mu_{ij}, \sigma_{ij})$ are numeric values greater than or equal to zero. The values of μ_{ij} and σ_{ij} represent the mean and the standard deviation, respectively, of a Gaussian probability distribution describing the C/I ratio when sectors i and j operate on the same frequency. The higher the mean value, the lower the interference and thus the better the communication quality. Note that the interference matrix is defined at the sector—or cell—level because the transceivers installed in each sector serve the same area.

A solution is obtained by assigning to each transceiver $t_i \in T$ one of the frequencies from F_i . Consequently, a candidate solution—or frequency plan—is denoted by $p \in F_1 \times F_2 \times \dots \times F_n$, where $p(t_i) \in F_i$ is the frequency assigned to the transceiver t_i . The objective is to find a solution p that minimises the following cost—objective—function:

$$C(p) = \sum_{t \in T} \sum_{u \in T, u \neq t} C_{\text{sig}}(p, t, u) \quad (9.1)$$

In order to define the function $C_{\text{sig}}(p, t, u)$ —Equation 9.2—let s_t and s_u be the sectors in which the transceivers t and u are installed, i.e. $s_t = s(t)$ and $s_u = s(u)$, respectively. Moreover, let $\mu_{s_t s_u}$ and $\sigma_{s_t s_u}$ be the two elements of the entry $M(s_t, s_u)$ of the interference matrix with respect to sectors s_t and s_u .

$$C_{\text{sig}}(p, t, u) = \begin{cases} K & \text{if } s_t = s_u, |p(t) - p(u)| < 2 \\ C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{if } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t) - p(u)| = 0 \\ C_{\text{adj}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{if } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t) - p(u)| = 1 \\ 0 & \text{otherwise.} \end{cases} \quad (9.2)$$

The parameter K in Equation 9.2 represents the cost associated with the use of the same or adjacent frequencies in the same sector. In real networks, it is unfeasible

to operate with more than one transceiver with the same or adjacent frequencies serving the same sector. Thus, K is defined as a very large constant. Function $C_{\text{co}}(\mu, \sigma)$ is defined as follows:

$$C_{\text{co}}(\mu, \sigma) = 100 \left(1.0 - Q \left(\frac{c_{\text{SH}} - \mu}{\sigma} \right) \right) \quad (9.3)$$

where

$$Q(z) = \int_z^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \quad (9.4)$$

is the tail integral of a Gaussian probability distribution function with zero mean and unit variance, and c_{SH} is a minimum quality signalling threshold. The Q function is widely used in digital communication systems because it characterises the error probability performance of digital signals [307]. This means that $Q\left(\frac{c_{\text{SH}} - \mu}{\sigma}\right)$ is the probability of the C/I ratio being greater than c_{SH} , and therefore $C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ computes the probability of the C/I ratio in the service area of sector s_t being below the quality threshold due to the interference caused by sector s_u . If this probability is low, the C/I ratio in sector s_t is not likely to be degraded by the interfering signal coming from sector s_u , and thus the resulting communication quality is high. Note that this is compliant with the definition of a minimisation problem. In contrast, a high probability—and consequently a high cost—mostly causes the C/I ratio to be below the minimum threshold c_{SH} , and thus results in low-quality communications.

Since function Q has no closed integral form, it has to be evaluated numerically. To do so, the complementary error function E is used:

$$Q(z) = \frac{1}{2} E \left(\frac{z}{\sqrt{2}} \right) \quad (9.5)$$

In [279], a numerical method was presented that allows computing the value of E with a fractional error smaller than $1.2 \cdot 10^{-7}$. Analogously, function $C_{\text{adj}}(\mu, \sigma)$ is defined as:

$$\begin{aligned} C_{\text{adj}}(\mu, \sigma) &= 100 \left(1.0 - Q \left(\frac{c_{\text{SH}} - c_{\text{ACR}} - \mu}{\sigma} \right) \right) \\ &= 100 \left(1.0 - \frac{1}{2} E \left(\frac{c_{\text{SH}} - c_{\text{ACR}} - \mu}{\sigma \sqrt{2}} \right) \right) \end{aligned} \quad (9.6)$$

The only difference between functions C_{co} and C_{adj} is the additional constant $c_{\text{ACR}} > 0$ (*Adjacent Channel Rejection*) in the definition of function C_{adj} . This hardware specific constant measures the receiver's ability to receive the desired signal in the presence of an unwanted signal in an adjacent channel. The effect of constant c_{ACR} is that $C_{\text{adj}}(\mu, \sigma) < C_{\text{co}}(\mu, \sigma)$. This is to be expected since using adjacent frequencies does not result in interference as strong as when the same frequency is used.

9.2 Optimisation Schemes

This section focuses on defining the different optimisation schemes that are applied in this thesis to solve the FAP. Particularly, the EAIPS proposed in [223] is considered. Moreover, several diversity-based objectives, as well as a multi-objectivisation method, are used together with a novel multi-objective MA. All the above optimisation schemes incorporate an individual learning procedure that was specifically designed to deal with the FAP.

9.2.1 Evolutionary Algorithm with Increasing Population Size

The EAIPS is a single-objective MA that combines an EA with a $(1 + 1)$ survivor selection operator and the individual learning procedure detailed in Section 9.2.4. The algorithm starts its execution as a trajectory-based approach. Nevertheless, it increases the population size to escape from local optima when stagnation is detected, thus behaving as a population-based algorithm.

Algorithm 9 shows the pseudocode of the EAIPS. During the initialisation stage—line 1—the initial parent population P is filled with N_0 randomly generated individuals, and the individual learning procedure is applied to each of them—line 2. Then, all individuals in P are evaluated—line 3—by computing the objective function defined in Equation 9.1, so that a fitness value is assigned to every individual. Afterwards, until the stopping criterion of the algorithm is satisfied—line 4—a set of steps is repeated for each generation. First, a variation stage is applied to the parent population—line 5—so as to produce the offspring population CP with $M = N$ new individuals. We should note that at this point, a new offspring is generated for each parent. The variation phase considered herein is described in Section 9.2.3. Once the variation stage is complete, the individual learning procedure is applied to every individual in CP —line 6—all of which are subsequently evaluated using the

Algorithm 9 Pseudocode of the Evolutionary Algorithm with Increasing Population Size

```

1: Initialisation. Randomly generate the initial parent population  $P$  with  $N_0$  individuals. Assign  $N = N_0$ .
2: Learning process. Apply the individual learning process to every individual in  $P$ .
3: Evaluation. Evaluate all individuals in  $P$  by computing the objective function in order to assign a fitness value to every individual.
4: while (not stopping criterion) do
5:   Variation. Apply the variation operators to the parent population so as to create the offspring population  $CP$  with  $M = N$  individuals.
6:   Learning process. Apply the individual learning process to every individual in  $CP$ .
7:   Evaluation. Evaluate all individuals in  $CP$  using the objective function in order to assign a fitness value to every offspring.
8:   for  $i = 1$  to  $N$  do
9:     if  $P(i)$  has been blocked for SoftBloq generations then
10:       $P(i) \leftarrow CP(i)$ 
11:     else
12:       $P(i) \leftarrow \text{bestFitness}(P(i), CP(i))$ 
13:     end if
14:   end for
15:   if  $P$  has been blocked for HardBloq generations then
16:     if  $N < N_{max}$  then
17:        $N = N + 1$ 
18:     end if
19:   end if
20: end while

```

objective function—line 7—so as to assign a fitness value to every offspring. The $(1+1)$ selection operator is deterministic and selects the fittest individual between a certain parent and its corresponding offspring. In order to better handle stagnation at local optima, two main features are incorporated into the algorithm. First, if after *SoftBloq* generations the fitness value of a given parent has not been improved by its corresponding offspring, a $(1, 1)$ survivor selection operator is applied during the current generation, i.e. the offspring is selected to survive independently of its fitness value—lines 8–14. Additionally, if after *HardBloq* generations the fitness value of none of the individuals in the parent population has been improved by their corresponding offspring, an additional individual is introduced into the parent population—lines 15–19—thus increasing the population size N for the next generation. So as to avoid the uncontrolled growth of the population, its maximum

size is limited to N_{max} individuals. Finally, to completely define the EAIPS, the individuals were encoded using arrays with n integer values (p_1, p_2, \dots, p_n) , where n is the total number of transceivers in the network, and p_i is the frequency assigned to the transceiver i .

9.2.2 Diversity-based Multi-objective Memetic Algorithm and Multi-objectivisation by Aggregation

The novel diversity-based multi-objective MA used herein to address the FAP is based on the NSGA-II—Section 2.4.1. The main difference with respect to the original NSGA-II is that the individual learning strategy detailed in Section 9.2.4 is applied to every generation after the variation phase. This is evident in the Algorithm 10. Since a diversity-based multi-objective MA is used, an additional diversity-based objective function must be considered together with the original objective function of the FAP defined in Equation 9.1. Different encoding-independent and genotypic diversity-based objectives were taken into account. In particular, the diversity-based objectives tested were *random*, *inversion*, ADI, DBI, and DCN—Section 3.1. Additionally, the diversity-based objective with parameters DBI-THR—Section 5.1—was also applied. So as to completely define the diversity-based multi-objective MA, individuals were encoded by means of arrays with n integer values (p_1, p_2, \dots, p_n) , where n is the number of transceivers deployed in the network and p_i is the frequency assigned to the transceiver i . Finally, the genetic operators applied during the variation stage of the diversity-based multi-objective MA are described in Section 9.2.3.

In addition to the application of the diversity-based multi-objective MA described above, multi-objectivisation by aggregation of helper-objectives is also employed herein to deal with the FAP. To do so, the multi-objective MA described in Algorithm 10 is combined together with a helper-objective, instead of using a diversity-based objective function. This helper-objective makes use of problem-dependent information and it is called *Dependent*. In order to calculate this helper-objective, the original objective function of the FAP, denoted by f , is decomposed into two independent functions, f_0 and f_1 , such that $f = f_0 + f_1$. The decomposition is performed as follows. First, a table containing all possible interferences between each pair of transceivers is generated. Then, this table is sorted based on the cost of the appearance of each pair ρ . The resultant position of each pair ρ in the sorted table is denoted by i_ρ . The cost associated with each pair ρ is taken into account in the function f_{obj} where $obj = i_\rho \bmod 2$. Hence, f_0 is used as the helper-objective.

Algorithm 10 Pseudocode of the memetic algorithm based on the Non-Dominated Sorting Genetic Algorithm II

- 1: **Initialisation.** Randomly generate the initial parent population P_0 with N individuals. Assign $t = 0$.
 - 2: **Evaluation.** Evaluate all the individuals in the initial parent population by calculating the objective functions.
 - 3: **while** (not stopping criterion) **do**
 - 4: **Fitness assignment.** Calculate the fitness values of individuals in P_t . Use the non-domination rank in the first generation, and the crowded comparison operator in remaining generations.
 - 5: **Parent selection.** Apply deterministic binary tournament selection with replacement to P_t in order to fill the mating pool with N parents.
 - 6: **Variation.** Apply the crossover and mutation operators with probabilities p_c and p_m , respectively, to the individuals in the mating pool in order to create the offspring population CP with $M = N$ new individuals.
 - 7: **Learning process.** Apply the individual learning process to every individual in the offspring population CP .
 - 8: **Evaluation.** Evaluate every offspring in CP by computing the objective functions.
 - 9: **Survivor selection.** Select the N fittest individuals from among N parents and M offspring by using the crowded comparison operator so as to constitute P_{t+1} .
 - 10: $t = t + 1$
 - 11: **end while**
-

Likewise, f_1 could have been used as the helper-objective. Lastly, all the remaining components, such as the encoding of the individuals, the genetic operators, and the individual learning procedure, are the same as those applied with the diversity-based multi-objective MA.

9.2.3 Genetic Operators

A variation phase that involves applying a crossover operator, and then a mutation operator, is performed in every generation of the MAs described in the previous sections. Recall that these operators are applied with probabilities p_c and p_m , respectively. Two different crossover operators, a random one and a novel proposal that takes into account problem-dependent information, are tested. The first one is the UX operator—Section 2.2.2—while the second one is called *Interference-based Crossover (IX)*. It operates as follows. First, a transceiver t_x is randomly selected. Every gene associated with a transceiver that interferes with t_x or is interfered with by t_x , including the gene corresponding to t_x , is inherited from the first parent.

The offspring's remaining genes are inherited from the second parent. The second offspring is generated by using an inverse mapping.

Once one of the above crossover operators is applied, the *Neighbour-based Mutation (NM)* is always applied as the mutation operator, since it yielded the best performance in [298]. Its operation is as follows. First, a random transceiver t_x is randomly reassigned. Then, the transceivers that interfere with t_x , or are interfered with by t_x , are included in a list called *affected* and are mutated with a probability p_m . The previous step is repeated R times, but in the subsequent iterations the transceiver is selected from among those that are included in the list *affected*. Hence, the NM operator focuses on one area of the network.

9.2.4 Individual Learning Strategy

The individual learning procedure described in this section, which is based on a stochastic hill climbing local search, was specifically designed to deal with the FAP variant addressed herein. It is used in every generation of the MAs introduced in Sections 9.2.1 and 9.2.2. Given its importance, a considerable effort was made to make this procedure as efficient as possible. The application of local search methods allows for admissible solutions to be obtained in relatively short times. This is a typical requirement within commercial tools, this being the context within which the FAP resides.

The operation of the local search—Algorithm 11—is based on optimising the assignment of frequencies to transceivers in a given sector, without modifying the remaining network assignments. Every neighbour of a candidate solution is obtained by replacing the frequencies in a sector's transceivers. Therefore, the neighbourhood size is the number of sectors in the network. The frequencies within a sector are reassigned as follows. First, the available frequencies for the sector are sorted by their corresponding cost. Then, two possibilities are considered. Either assign the frequency with the lowest associated cost to a transceiver that is permitted to use that frequency, or assign its two adjacent frequencies to two different transceivers that are allowed to use these frequencies. For each of the newly generated partial solutions, the same process is repeated until every transceiver in the sector is assigned a frequency. The complete solution with the lowest associated cost becomes the new neighbour, while the other ones are discarded.

Figure 9.1 illustrates the generation of a new neighbour. In this example, the sector is assumed to contain three transceivers, where each transceiver can use any frequency slot. For every node, the cost associated with each slot is shown. The children of a

Algorithm 11 Pseudocode of the individual learning strategy designed for the Frequency Assignment Problem

```

1: Input: current solution  $S$ 
2:  $nextSectors \leftarrow \{1, \dots, numberOfSectors\}$ 
3: while ( $nextSectors \neq \emptyset$ ) do
4:    $currentSectors \leftarrow nextSectors$ 
5:    $nextSectors \leftarrow \emptyset$ 
6:   while ( $currentSectors \neq \emptyset$ ) do
7:      $sec \leftarrow$  extract a random sector from  $currentSectors$ 
8:      $neighbour \leftarrow$  reassign frequencies of  $S$  in sector  $sec$ 
9:     if ( $neighbour$  improves  $S$ ) then
10:       $S \leftarrow neighbour$ 
11:       $nextSectors \leftarrow nextSectors \cup \{\text{sectors interfered with by } sec\}$ 
12:       $nextSectors \leftarrow nextSectors \cup \{\text{sectors that interfere with } sec\}$ 
13:    end if
14:  end while
15: end while
16: return  $S$ 

```

node are generated in accordance with the rules detailed earlier. The slots assigned to the transceivers are bolded. The nodes with three slots assigned are complete solutions, while the other ones are partial solutions. The complete solution identified by the number 3 is the new neighbour because it is the one with the lowest cost. All other solutions generated are discarded.

The neighbours are analysed in random order—line 7 of Algorithm 11—while trying to avoid the generation of neighbours that do not improve the current solution. This is done by using a set called *currentSectors* containing the sectors that might improve the current solution. Initially, all the sectors are input into *currentSector*—lines 2 and 4. In order to generate a new neighbour, a sector *sec* is randomly extracted from *currentSectors*—line 7—and its frequencies reassigned as discussed above—line 8. The local search moves to the first new generated neighbour that improves the current solution—lines 9–10—adding all the sectors that interfere with or are interfered with by *sec* to the set of the next sectors—*nextSectors*—to be considered—lines 11–12. When the set *currentSectors* becomes empty—line 6—sectors in *nextSectors* are transferred to the current set—line 4—and the set *nextSectors* is cleared—line 5. The local search stops—line 3—when no neighbour improves the current solution.

In cases where the network satisfies a set of properties, the neighbour generation

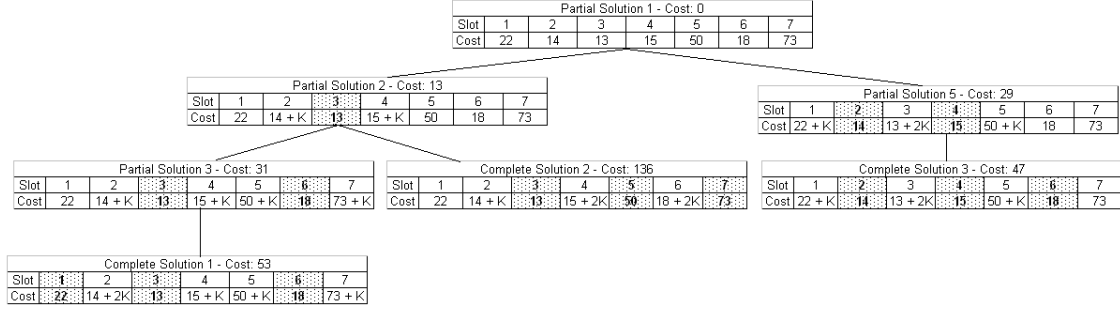


Figure 9.1: Generating a new neighbour by reassigning the frequencies belonging to a certain sector

process ensures that the frequency assignment inside the analysed sector will be optimal while keeping the rest of the network intact. An sketch of the proof was presented in [298]. These properties are:

1. All transceivers in a given sector are allowed to use the same frequency ranges.
2. It is possible to make assignments that do not use the same frequency or adjacent frequencies in any two transceivers serving the same area.
3. The optimal assignment does not use the same frequency or adjacent frequencies in any two transceivers located in the same sector.

9.3 Parameter Control Schemes

This section is devoted to describing the different parameter control approaches that are used herein to adapt some of the parameters contained in the optimisation schemes detailed in preceding sections. First, FLCs and hyper-heuristics are used to control certain parameters of the diversity-based multi-objective MA described in Section 9.2.2. In particular, the parameters of the NM operator, which was introduced in Section 9.2.3, are dynamically adjusted during the course of a run. Second, the DYN model is applied herein to control some of the components and parameters corresponding to the diversity-based multi-objective MA and the multi-objective MA based on multi-objectivisation by aggregation presented in Section 9.2.2. In addition, by applying the DYN model, both of the aforementioned memetic approaches are enabled for use in parallel environments.

9.3.1 Fuzzy Logic Controllers and Hyper-heuristics

The novel FLCs proposed in Section 6.1, as well as the hyper-heuristics described in Section 6.2, are used herein to control the parameters of the NM operator. The main novelty of the FLCs proposed lies in the incorporation of a set of different rule bases that are enabled depending on historical information extracted from the optimisation process. Recall that two different variants of the Mamdani-type and TSK-type FLCs were defined. Particularly, the FUZZY-A and FUZZY-B Mamdani-type FLCs, as well as the FUZZY-A-TSK and FUZZY-B-TSK TSK-type FLCs, are considered when carrying out the experimental evaluation. Additionally, the HH-PROB and HH-ELI hyper-heuristics approaches are used for comparison purposes. It is important to remark that this is the first application of parameter control techniques based on fuzzy logic and hyper-heuristics that adapts the parameters of a mutation operator specifically designed to address the FAP.

One of the main drawbacks of applying this operator is that two different parameters must be set. One of these parameters— p_m —is continuous and the other one— R —is discrete. In addition, the most suitable values for these parameters could depend on the problem and/or instance being solved or even on the current stage of the optimisation process, and therefore modifying them during the execution might be beneficial. Consequently, the application of parameter control techniques to automatically adapt these parameters ought to significantly improve both the behaviour and the robustness of the diversity-based multi-objective MA proposed to deal with the FAP. This idea seems to be very promising and is addressed in detail herein. Finally, we should note that only one of the above parameters is controlled during the execution, while the other remains constant, as a result of which two independent studies are carried out, one for each parameter of the NM operator.

9.3.2 Dynamic-mapped Island-based Model

The DYN model introduced in Section 6.3 is used herein as a parameter control approach. Recall that in the DYN model a hyper-heuristic is combined together with a parallel island-based model in order to dynamically map the low-level configurations involved to the islands, rather than performing a static mapping as is the case with standard island-based models. In this case, the DYN model is based on the HH-PROB hyper-heuristic, which was detailed in Section 6.2. In order to deal with the FAP, the set of low-level configurations was defined starting from the

diversity based multi-objective MA and the multi-objective MA based on multi-objectivisation by aggregation, which were described in Section 9.2.2. Hence, the aim is to control certain components and parameters of these memetic approaches while enabling their use in parallel environments.

Furthermore, several migration stages were tested with the DYN model. In particular, four different migration stages were defined by combining two different replacement schemes with two separate migration topologies. The replacement schemes were HAM-R and ELI-R, whereas the migration topologies were ALL and RING—Section 2.6.1. The ELI-M migration scheme—Section 2.6.1—was used to define the four migration stages. In order to identify the different migration stages, the *Replacement-Topology* nomenclature is used. For example, the migration stage that uses the ELI-R replacement scheme and the RING topology is called *ELI-R-RING*.

9.4 Experimental Evaluation and Discussion

In this section the different experiments performed with the aforementioned optimisation schemes and parameter control approaches on different instances of the FAP are presented.

Experimental Method. The different optimisation schemes, as well as the parameter control approaches, were implemented using METCO. The tests were run on a Debian GNU/Linux computer with four AMD ® OpteronTM processors (model number 6164HE) at 1.7 GHz and 64 Gb RAM. The compiler was the GCC 4.7.2, while the FLCs were implemented using the *fuzzylite* 3.1 library. Communications among different islands of the DYN model were implemented asynchronously using the MPICH library. Since every experiment applied stochastic algorithms, every execution was repeated 32 times. As a result, comparisons were performed by applying the statistical analysis detailed in Section 1.2.6.

FAP Instances. The studies were conducted considering two different instances representing two real cities in the USA: Seattle and Denver. The Seattle instance has $n = 970$ transceivers and 15 different frequencies to be assigned. The Denver instance is larger, consisting of $n = 2,612$ transceivers and 18 frequencies. In both cases, the constants used in the formal definition of the FAP presented in Section 9.1 were set to $K = 1 \cdot 10^5$, $c_{SH} = 6 \text{ dB}$, and $c_{ACR} = 18 \text{ dB}$. The matrix M contains

59,169 items in the Seattle network, while it contains 20,638 items for the Denver instance.

9.4.1 On the Comparison of Sequential Memetic Algorithms

In this first experiment the aim is to discover whether the use of the proposed diversity-based multi-objective MA, as well as the use of the proposed multi-objective MA based on multi-objectivisation by aggregation *Dependent*, offers any benefits over using the best-behaved single-objective MA available in the literature, i.e. the EAIPS. To do this, the following configurations of the three optimisation schemes above were compared:

- 3 configurations of the EAIPS, which were constituted by defining three different configurations for the variation stage.
- 18 configurations of the diversity-based multi-objective MA, which were constituted by combining six diversity-based objectives with three different configurations of the variation stage.
- 3 configurations of the *Dependent* approach, which were constituted by defining three different configurations for the variation phase.

The three configurations of the variation stage were defined as follows. In the first configuration, the UX operator with a probability $p_c = 1$ was used. In the second one, the IX operator with a probability $p_c = 1$ was applied. Finally, in the third configuration, the crossover operator was disabled, i.e. $p_c = 0$. All the above configurations of the variation stage applied the NM mutation operator with a probability $p_m = 0.9$ and $R = 7$.

The values for the remaining parameters of the EAIPS were set as follows:

- Initial population size $N_0 = 1$.
- Maximum population size $N_{max} = 5$.
- *SoftBloq* = 50.
- *HardBloq* = 300.
- Stopping criterion fixed to 4 hours.

In the case of the diversity-based multi-objective MA, the following parameterisation was considered:

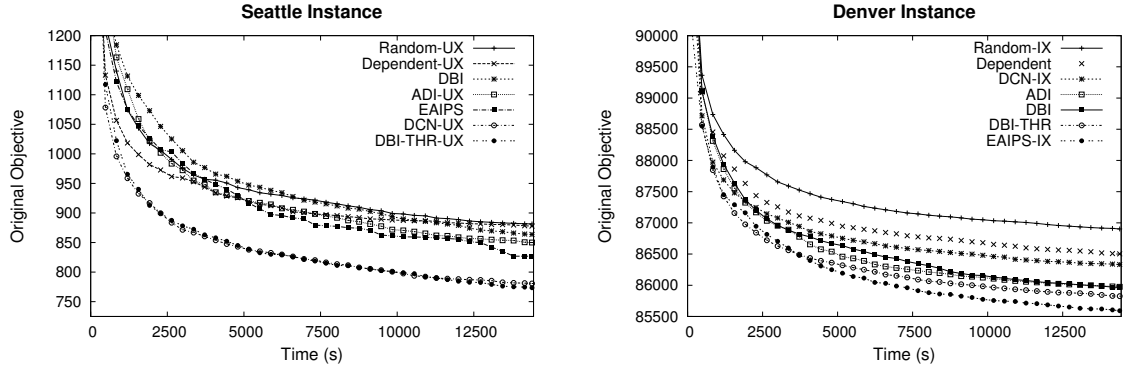


Figure 9.2: Evolution of the mean original objective value for the different memetic approaches

- Population size $N = 10$.
- Diversity-based objectives *random*, *inversion*, ADI, DBI, DCN, and DBI-THR.
- Threshold value $th = 0.9$ for the diversity-based objective DBI-THR.
- Stopping criterion fixed to 4 hours.

Finally, the *Dependent* scheme was applied with a population size $N = 10$ and a stopping criterion equal to 4 hours.

This analysis was carried out in terms of the performance achieved in the original objective function. Hence, Figure 9.2 shows, for both instances, the evolution of the mean original objective value achieved by the memetic optimisation schemes considered. In order to differentiate the variants of these memetic approaches, the *approach-crossover* nomenclature is used. For instance, the configuration of the diversity-based multi-objective MA *Random-UX* applies the diversity-based objective *Random* and the UX operator in the variation stage, while the EAIPS-IX configuration uses the IX operator in the variation phase of the EAIPS. In cases where only the name of the approach is shown, such as *Dependent*, the crossover operator is disabled, and only the NM operator is applied in the variation stage. Since the *Inversion* diversity-based objective obtained poor results, its data is not shown. For the remaining optimisation schemes, the data corresponding to the configuration that yielded the best-behaved variation stage, in terms of the mean original objective value achieved at the end of the executions, is shown.

It can be thus observed that for the Seattle instance, the DBI-THR-UX and DCN-UX approaches obtained the lowest mean of the original objective value at the end

Table 9.1: Statistical comparison among different memetic approaches for the Seattle instance

	DBI-THR-UX	DCN-UX	EAIPS	ADI-UX	DBI	Dependent-UX	Random-UX
DBI-THR-UX	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow
DCN-UX	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow
EAIPS	\downarrow	\downarrow	\leftrightarrow	\uparrow	\uparrow	\leftrightarrow	\uparrow
ADI-UX	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
DBI	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
Dependent-UX	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
Random-UX	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 9.2: Statistical comparison among different memetic approaches for the Denver instance

	EAIPS-IX	DBI-THR	DBI	ADI	DCN-IX	Dependent	Random-IX
EAIPS-IX	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
DBI-THR	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
DBI	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow
ADI	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
DCN-IX	\downarrow	\downarrow	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow
Dependent	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
Random-IX	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow

of the executions. The worst results, however, were provided by the *Random-UX* and *Dependent-UX* schemes. Furthermore, for this instance, note that in general the best-behaved variation stage applied the UX operator. In the case of Denver, the EAIPS-IX approach was able to attain the lowest mean of the original objective value at the end of the executions, followed by the DBI-THR scheme, whereas the worst results were given by the *Random-IX* approach. Generally, for this instance, the best-behaved variation stage did not use any crossover operator.

In order to perform a statistical comparison, Table 9.1 shows, for the Seattle instance, the statistical differences among the different memetic approaches. Particularly, the table shows whether the scheme located in a given row is statistically better (\uparrow), not different (\leftrightarrow), or worse (\downarrow) than the corresponding scheme situated in a certain column. Table 9.2 shows the same information for the Denver instance. It is worth mentioning that in the case of Seattle, the DBI-THR-UX and DCN-UX approaches, which obtained the lowest mean of the original objective value at the end of the executions, were able to statistically outperform the EAIPS scheme, thus demonstrating the advantages provided by the diversity-based multi-objective MA with respect to the best single-objective MA available in the literature.

For the Denver instance, the lowest mean of the original objective value at the end of the executions was achieved by the EAIPS-IX approach, followed by the DBI-THR scheme. However, the differences between the EAIPS-IX method and the DBI-THR,

Table 9.3: Statistical comparison among different configurations of the variation stage

	Seattle			Denver		
	UX	No Crossover	IX	UX	No Crossover	IX
UX	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\downarrow	\downarrow
No Crossover	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow	\leftrightarrow
IX	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow	\leftrightarrow

DBI, and ADI approaches were not statistically significant. It can thus be concluded that both the single-objective MA and the diversity-based multi-objective MA were able to provide solutions of similar quality for this particular instance. As a result, it is important to remark that there exist test cases for which the diversity-based multi-objective MA is able to provide similar results to those given by the single-objective MA, but there are other test cases where the former yields even better results than the latter.

Finally, if the diversity-based multi-objective MA and the multi-objective MA based on multi-objectivisation by aggregation are taken into consideration, we should note that the results given by the former clearly outperform those obtained by the latter. Consequently, for the particular case of the FAP, the use of a helper-objective that takes into account problem-dependent information provided no benefit over more general problem-independent diversity-based objectives.

Taking into account the configuration of the diversity-based multi-objective MA that showed the best performance for both instances, i.e. the DBI-THR approach, Table 9.3 shows a statistical comparison among the three variation stages tested for Seattle and Denver. For the Seattle instance, there were no significant differences between the different variation stages. Consequently, for this particular case, modifying the variation stage does not alter the performance of the diversity-based multi-objective MA. In the case of the Denver instance, the IX operator was statistically better than the UX operator but not statistically different from the variation scheme in which the crossover operator was not applied. As a result, a more in-depth analysis with other instances should be carried out in order to shed light on the potential benefits that the IX operator is able to provide.

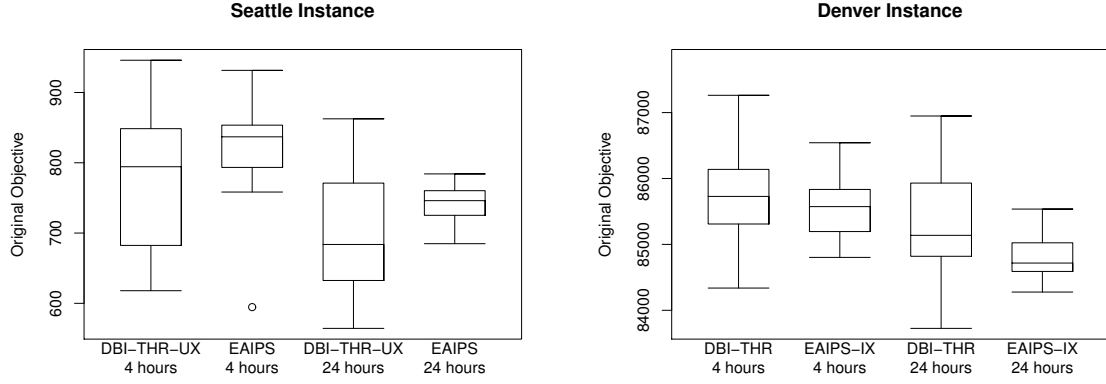


Figure 9.3: Box plots for the best configurations of the memetic approaches

9.4.2 Long-term Analysis of the Sequential Memetic Algorithms

The goal of the second experiment is to check the behaviour of the diversity-based multi-objective MA and the EAIPS in the long term. To do so, the configurations of the diversity-based multi-objective MA that provided the best results in the previous experiment—DBI-THR-UX for Seattle and DBI-THR for Denver—were executed using a stopping criterion of 24 hours. The configurations of the EAIPS that were able to achieve the best results in the previous experiment—EAIPS for Seattle and EAIPS-IX for Denver—were also run during 24 hours. Both optimisation schemes applied the same parameter values than those used in the first experiment.

Figure 9.3 shows, for both instances, the box plots of the original objective values achieved by the different executions in 4 and 24 hours. In the case of Seattle, the DBI-THR-UX approach was statistically better than the EAIPS scheme considering both stopping criteria. For the particular case of Denver, and taking into account 4 hours of execution, the DBI-THR and EAIPS-IX approaches provided similar results. In fact, differences between them were not statistically significant. Regarding the stopping criterion of 24 hours, most of the executions belonging to the EAIPS-IX scheme obtained better results than those given by the executions of the DBI-THR approach. Nevertheless, it is worth noting that some executions of the latter were able to better deal with premature convergence issues. In fact, the diversity-based multi-objective MA was able to obtain the best frequency plans for both instances, as it can be observed in the box plots.

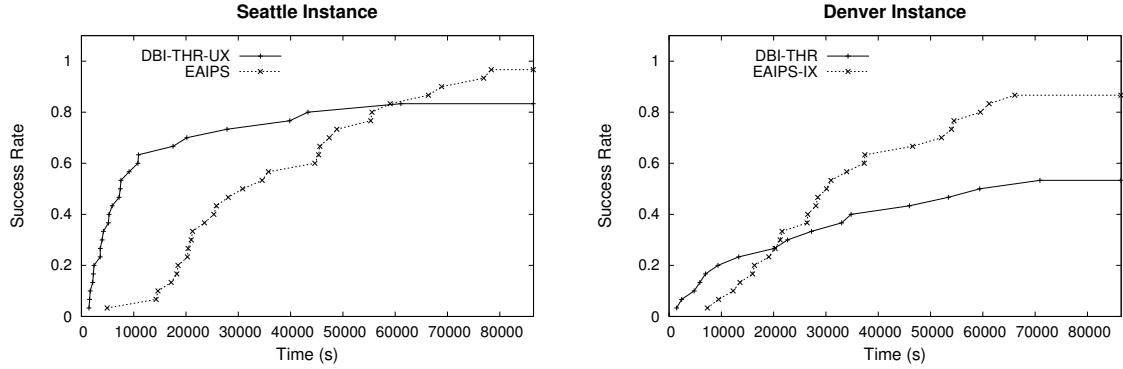


Figure 9.4: Run-length distributions for the best configurations of the memetic approaches

In order to quantify the run-time behaviour of the approaches considered in this experiment, the RLDs described in Section 1.2.6 were applied. The quality level was set as the mean original objective value achieved by the EAIPS configuration for Seattle and the EAIPS-IX configuration for Denver over an 8-hour execution. Figure 9.4 shows the RLDs for both instances. In the case of Seattle, note that 50% of the executions performed using the DBI-THR-UX approach reached the specified quality level in less than 7,440 seconds. For the EAIPS configuration, 50% of its executions invested 30,840 seconds in order to achieve the same quality level. Hence, the EAIPS configuration required 4.15 times more resources—time—than the DBI-THR-UX approach to achieve the pre-determined quality level. Nevertheless, the EAIPS scheme was able to attain a higher success rate than the DBI-THR-UX configuration after a 24-hour execution.

Taking into account the Denver instance and a 50% success rate, the EAIPS-IX approach was 1.97 times faster than the DBI-THR scheme in reaching the specified quality level. Additionally, considering the results after 24 hours of execution, the EAIPS-IX configuration obtained a higher success rate than the DBI-THR approach. However, the latter was able to attain higher success rates than the former when execution times under 5.5 hours were considered. As a result, in terms of the quality level specified for this analysis, the most suitable optimisation scheme depends on the time constraints.

Finally, note that the best frequency plans for the instances considered were reported in [298, 224]. However, the configurations of the diversity-based multi-objective MA applied in this experiment were able to improve on said frequency plans. In the case of Seattle, the original objective value decreased from 654.5 to 564.3, whereas

the original objective value for Denver decreased from 83,991.3 to 83,725.6. The configurations of the EAIPS used herein yielded the values 594.6 and 84,276.8 for Seattle and Denver, respectively.

9.4.3 Analysing the Robustness of the Dynamic-mapped Island-based Model

The diversity-based multi-objective MA was able to provide the best solutions in the previous experiment. However, as was stated, the main drawback of this approach is that several components and parameters have to be tested in order to obtain high-quality solutions, which also requires very long execution times. Since the optimal parameterisation depends on the instance being solved, the parameter setting involves a large computational and user effort. Thus, the aim of the current experiment is threefold. First, to adaptively adjust some of the components and parameters of the multi-objective MAs applied in previous experiments, as well as to enable their use in parallel environments, by applying the DYN model. Second, to analyse the robustness of the DYN model in terms of the migration stage used. Finally, to study the ability of the DYN model to obtain high-quality results in a single run while using a low number of processors.

To do so, the DYN model was executed with the four migration stages described in Section 9.3.2. The migration rate for all was set to 1 individual, whereas the migration probability was set to 0.01. A total amount of $n_p = 4$ islands was considered. The global stopping criterion was set to 11.5 hours of execution, while the local stopping criterion was set to 10 minutes. The HH-PROB hyper-heuristic of the DYN model was applied with an adaptation level $k = 10$, and the value of β was set such that 10% of the decisions made by the hyper-heuristic used a uniform distribution, i.e. $\beta \cdot n_h = 0.1$. Moreover, $n_h = 21$ low-level configurations of the multi-objective MAs depicted in Section 9.2.2 were used as the low-level configurations. These low-level approaches were the 18 configurations of the diversity-based multi-objective MA and the 3 configurations of the multi-objective MA based on multi-objectivisation by aggregation *Dependent* applied in the first experiment—Section 9.4.1. Their parameter values were also the same as those used in said experiment. So as to compare the results obtained by the DYN model, the sequential versions of the above 21 low-level configurations were also executed for 11.5 hours. They were sorted based on the mean original objective value achieved at the end of their executions, and an index based on this order was assigned to each one. The best sequential low-level configuration, i.e. the one that achieved the lowest

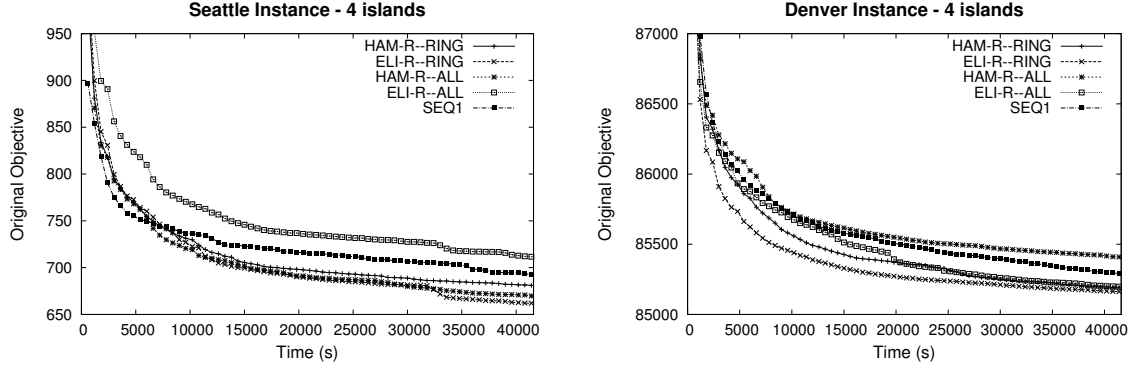


Figure 9.5: Evolution of the mean original objective value for the dynamic-mapped island-based model executed with 4 islands

mean original objective value at the end of the executions, is referred to as SEQ1, whereas the worst one is referred to as SEQ21.

The executions conducted as part of this experiment were performed on the HECToR machine [102], the UK’s National Supercomputing Service. The Phase 3 (Cray XE6) system of HECToR is contained in 30 cabinets and comprises of a total of 704 compute blades. Each blade contains four compute nodes, each with two 16-core AMD ® Opteron™ 2.3 GHz Interlagos processors. This amounts to a total of 90,112 cores, offering a theoretical peak performance of over 800 Tflops. Each 16-core socket is coupled with a Cray Gemini routing and communications chip, and shares 16 Gb of memory. Finally, due to restrictions in the computational resources available, 24 executions, and not 32, were performed for each of the aforementioned optimisation schemes.

Figure 9.5 shows, for both instances, the evolution of the mean original objective value obtained by the DYN model combined together with the four migration stages, and by the best-behaved low-level configuration. In the case of the Seattle instance, note that although the DYN model used a larger amount of computational resources than the best sequential configuration, three parallel models were able to obtain better results than SEQ1, in terms of the original objective value achieved at the end of the executions. As a result, it is worth pointing out that by using the DYN model, the requirement to test each of the 21 low-level configurations under consideration can be avoided, thus considerably reducing the amount of computational resources invested. Executing a single run of the DYN model yields better frequency plans than those provided by a significant number of low-level configurations executed independently. In addition, since the parameter setting stage is alleviated, the effort

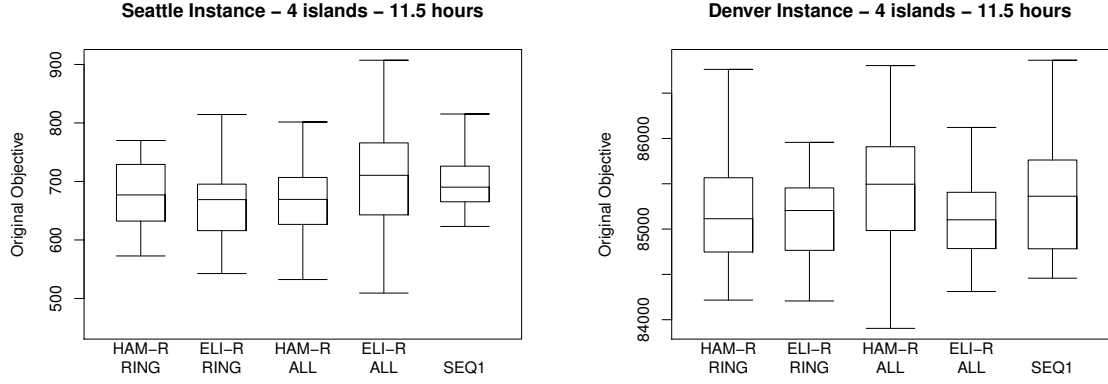


Figure 9.6: Box plots for the dynamic-mapped island-based model executed with 4 islands

required by the user to solve a new network instance is also reduced. For the Denver instance, three parallel models were also able to improve on the results achieved by SEQ1, and consequently the same conclusions as those given for the Seattle instance can also be drawn for this particular case. Finally, we should note that the lowest mean of the original objective value was achieved by the ELI-R–RING scheme for both instances considered.

In order to better analyse the results, the box plots of the original objective values obtained by every model at the end of the runs are shown in Figure 9.6 for both instances. Note that the results yielded by all of the approaches were very similar for both test cases. Table 9.4 shows, for the Seattle instance, whether the scheme located in a given row is statistically better (\uparrow), not different (\leftrightarrow), or worse (\downarrow) than the corresponding scheme situated in a certain column after 11.5 hours of execution. This table reveals that the ELI-R–RING scheme was statistically better than the ELI-R–ALL approach, while the differences for the remaining schemes were not statistically significant. The same information is shown in Table 9.5 for the Denver instance. In this case, no statistically significant differences appeared among the different approaches. Therefore, the DYN model was able to yield competitive frequency plans for the instances considered, regardless of the migration stage applied, thus demonstrating its robustness.

This analysis compared the different parallel models in terms of the quality achieved after fixed execution times. It is important, however, to quantify the improvement achieved by these parallel approaches in terms of the amount of time saved. To

Table 9.4: Statistical comparison among different migration stages for the Seattle instance – 4 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL	SEQ1
HAM-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow
HAM-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
SEQ1	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 9.5: Statistical comparison among different migration stages for the Denver instance – 4 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL	SEQ1
HAM-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
HAM-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
SEQ1	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

do so, RLDs were calculated for the different optimisation schemes considered in this experiment. For each instance, the quality level was set as the median of the original objective value obtained by SEQ5 at the end of its executions, i.e. at 11.5 hours. Figure 9.7 shows, for both instances, the RLDs for the parallel schemes, as well as for the SEQ1, SEQ3, and SEQ5 sequential configurations. Note the existing similarities between the parallel models and SEQ1, thus validating the conclusions drawn by the statistical comparison carried out in the previous study. Some parallel models, however, were able to achieve higher success rates than the SEQ1 approach. Additionally, the RLDs show the clear superiority of the parallel models with respect to the remaining sequential configurations.

Table 9.6 shows the speedup factors obtained by the parallel models required to attain a 50% success rate for the Seattle instance in comparison to the SEQ1, SEQ3, and SEQ5 approaches. The ELI-R-ALL parallel scheme was clearly outperformed by the remaining approaches. In fact, considering a 50% success rate, it took four times longer than SEQ1 and SEQ3 to achieve the pre-set quality level. The remaining parallel schemes, however, behaved noticeably better, even yielding super-linear speedup factors with respect to SEQ5. In these cases, the hyper-heuristic probably granted more computational resources to the best-behaved low-level configurations, meaning the decision space was explored more efficiently. Table 9.7 shows the speedup factors for the Denver instance. Similar conclusions can be drawn for this

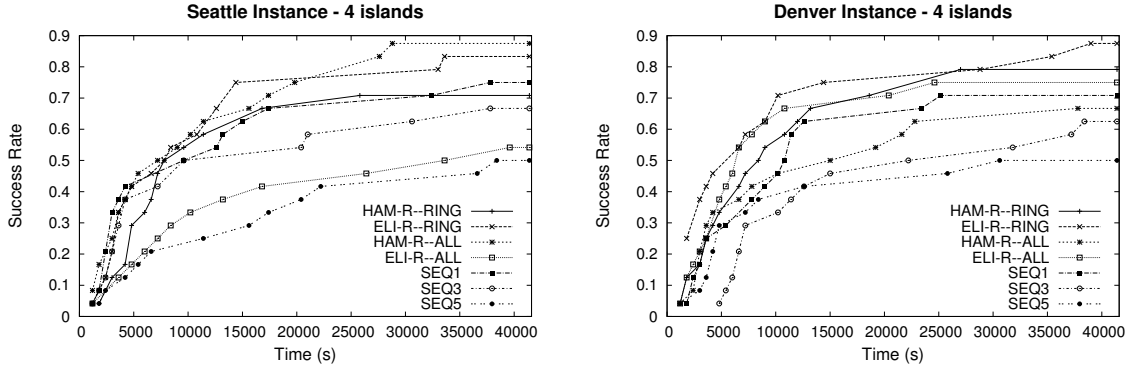


Figure 9.7: Run-length distributions for the dynamic-mapped island-based model executed with 4 islands

Table 9.6: Speedup factors achieved by the dynamic-mapped island-based model for the Seattle instance – 4 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
SEQ1	1.23	1.23	1.33	0.28
SEQ3	1.23	1.23	1.33	0.28
SEQ5	4.92	4.92	5.33	1.14

Table 9.7: Speedup factors achieved by the dynamic-mapped island-based model for the Denver instance – 4 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
SEQ1	1.28	1.63	0.72	1.63
SEQ3	2.64	3.36	1.48	3.36
SEQ5	3.64	4.63	2.04	4.63

particular case. The HAM-R-ALL approach obtained the worst results. Considering a 50% success rate, it took longer than SEQ1 to reach the pre-set quality level. However, the ELI-R-RING and ELI-R-ALL parallel models were able to provide super-linear speedup factors in comparison to SEQ5.

9.4.4 Analysing the Scalability of the Dynamic-mapped Island-based Model

In the previous experiment the DYN model was executed assuming $n_p = 4$ islands. As was stated, the differences among the results obtained by the different migration

Table 9.8: Speedup factors achieved by the dynamic-mapped island-based model for the Seattle instance – 8, 16, and 32 islands

	8 Islands	16 Islands	32 Islands
HAM-R-RING	6.37	12.75	17.00
ELI-R-RING	5.66	10.20	12.75
HAM-R-ALL	0.75	5.66	7.28
ELI-R-ALL	1.02	4.25	2.68

Table 9.9: Speedup factors achieved by the dynamic-mapped island-based model for the Denver instance – 8, 16, and 32 islands

	8 Islands	16 Islands	32 Islands
HAM-R-RING	11.50	13.80	17.25
ELI-R-RING	11.50	17.25	23.00
HAM-R-ALL	8.62	13.80	5.75
ELI-R-ALL	6.90	7.66	9.85

stages considered were not statistically significant. However, for a larger number of islands, the differences might be more evident. We should note that in the RING migration topology a certain island sends its solutions to another island, whereas in the ALL migration topology a given island sends its solutions to $n_p - 1$ islands. As n_p increases, so do the differences between the two migration topologies. Hence, the goal of this fourth experiment is to analyse the scalability of the DYN model, in terms of the effect on performance caused by the different migration stages, as the number of islands grows. To do this, the DYN model was executed with the same parameterisation and four migration stages as those used in the previous experiment, but considering 8, 16, and 32 islands. The experiments, consisting of 24 repetitions, were also carried on the HECToR machine.

The speedup factors for the different parallel schemes, using SEQ1 as the reference approach, are shown in Table 9.8 for the Seattle instance. They were calculated considering the time required to attain a 50% success rate, with the quality level set as the lowest median of the original objective value achieved by any of the parallel models at the end of their executions, i.e. after 11.5 hours. The differences among the different parallel approaches were noticeable. For instance, using 32 islands, the speedup values ranged from 2.68 to 17.00. Additionally, the parallel models that used the RING migration topology achieved the pre-set quality level faster than the models that applied the ALL migration topology. In fact, specifying 32 islands and a stopping criterion of 11.5 hours, the statistical tests shown in Table 9.10 confirm the superiority of the parallel models based on the RING topology. The same

Table 9.10: Statistical comparison among different migration stages for the Seattle instance – 32 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL	SEQ1
HAM-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
HAM-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow
ELI-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
SEQ1	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow

Table 9.11: Statistical comparison among different migration stages for the Denver instance – 32 islands

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL	SEQ1
HAM-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
HAM-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow
ELI-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow
SEQ1	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow

analysis was carried out for the Denver instance. The speedup factors are shown in Table 9.9 for this test case. Differences among the different parallel approaches were also obvious. For instance, for 32 islands, the speedup values ranged from 5.75 to 23.00. Furthermore, as was also the case with Seattle, the superiority of the parallel models that applied the RING migration topology was also clear with respect to the parallel models that used the ALL migration topology. The statistical tests shown in Table 9.11 confirm this last fact. In summary, the scalability analysis revealed that as the number of islands increases, the performance of the DYN model is more sensitive to the setting of the migration stage, and consequently the practitioner must carefully select its components.

In this experiment, the quality of the frequency plans achieved by the parallel models using a high number of islands was also analysed. For the Seattle instance, the box plots of the best and worst parallel models, in terms of the speedup values obtained with 32 islands, are shown in Figure 9.8. In the case of the DYN model executed with the HAM-R-RING migration stage, the addition of extra islands into the scheme produced a clear decrease in the original objective value. In contrast, when the ELI-R-ALL migration stage was incorporated, the DYN model was unable to profit from the addition of more islands. In the case of Denver, the box plots for the best and worst parallel models are shown in Figure 9.9. For this network, the same conclusions can be drawn. However, the best and worst migration stages were

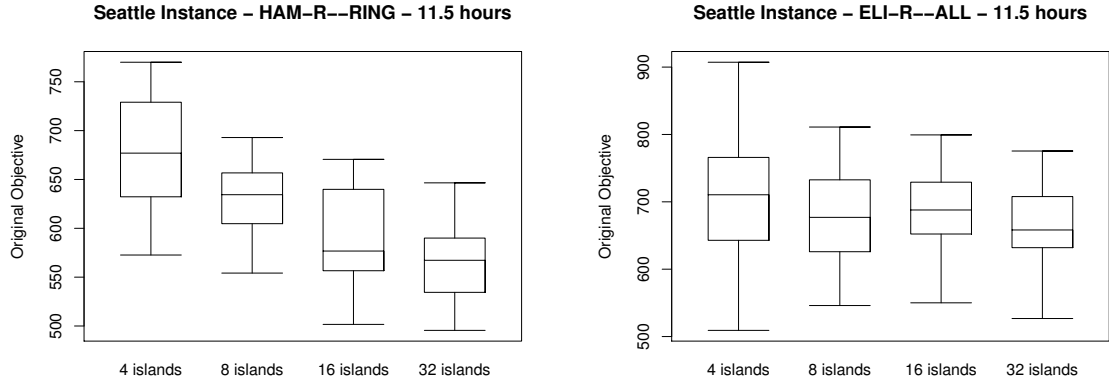


Figure 9.8: Box plots for the best (left-hand side) and worst (right-hand side) migration stages for the Seattle instance – 4, 8, 16, and 32 islands

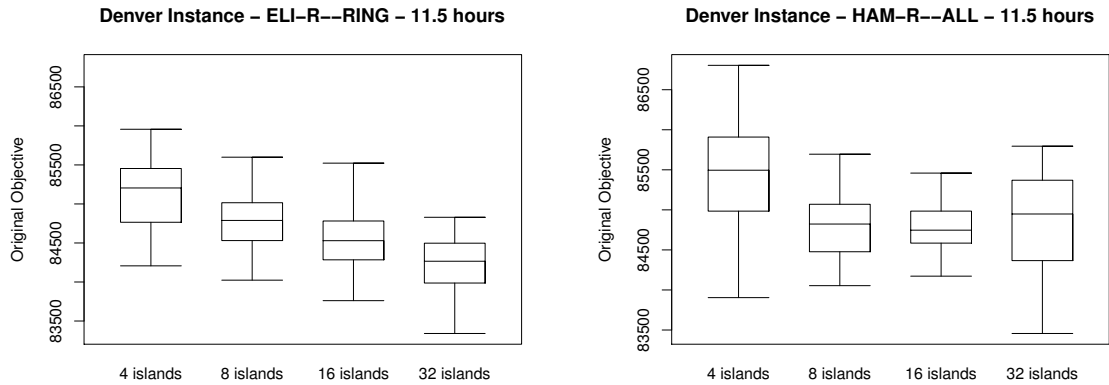


Figure 9.9: Box plots for the best (left-hand side) and worst (right-hand side) migration stages for the Denver instance – 4, 8, 16, and 32 islands

ELI-R–RING and HAM-R–ALL, respectively.

Finally, we should note that the best frequency plans for both instances, which were previously obtained in the second experiment described in Section 9.4.2, were improved upon in this experiment. Said best frequency plans were obtained by the DYN model executed with the ELI-R–RING migration stage and 32 islands. In the case of Seattle the original objective value decreased from 564.3 to 486.6, while the

Table 9.12: Parameterisation of the diversity-based multi-objective memetic algorithm for different values of the parameter p_m

Parameter	Value	Parameter	Value
Stopping criterion	$1.5 \cdot 10^5$ evals.	Crossover rate (p_c)	1
Population size (N)	10 individuals	Mutation rate (p_m)	0, 0.1, 0.2, ..., 0.9, 1
Crossover operator	UX (Seattle), IX (Denver)	NM operator steps (R)	7
Auxiliary objective	DCN (Seattle), ADI (Denver)		

Table 9.13: Parameterisation of the hyper-heuristics HH-ELI and HH-PROB to control the parameter p_m

Parameter	Value	Parameter	Value
Local stopping criterion	$1.5 \cdot 10^3, 3 \cdot 10^3$ evals.	Minimum selection rate (β)	0.1
Number of low-level configs. (n_h)	11 configs.	Historical knowledge (k)	2, 5

original objective value for Denver decreased from 83,725.6 to 83,340.2.

9.4.5 Adapting the parameter p_m of the Neighbour-based Mutation Operator

In this experiment the parameter control approaches based on FLCs and hyper-heuristics described in Section 9.3.1 were applied to the parameter p_m of the NM operator to solve both of the FAP instances considered. The diversity-based multi-objective MA presented in Section 9.2.2 was also executed with different values for the parameter p_m , while the value of R was kept constant. The main aim was to analyse the performance of the different parameter control approaches and to study whether parameter control offers any benefits in regard to tuning the parameter p_m .

A common parameterisation was used for the diversity-based multi-objective MA and the different parameter control schemes. Table 9.12 shows the parameterisation of the diversity-based multi-objective MA. Different configurations—exactly 11—were defined by modifying the value of the parameter p_m . Moreover, the diversity-based objectives and the crossover operator considered for each of the two instances were different. This is because, depending on the instance, the most appropriate values for these components change.

The parameterisations of the different parameter control approaches are shown in Tables 9.13 and 9.14 for the hyper-heuristics and the FLCs, respectively. Note that four different configurations for the HH-ELI and HH-PROB hyper-heuristics were applied by combining different values for the local stopping criterion and the parameter k . Similarly, four configurations of the FUZZY-A, FUZZY-B, FUZZY-

Table 9.14: Parameterisation of the fuzzy logic controllers FUZZY-A, FUZZY-B, FUZZY-A-TSK, and FUZZY-B-TSK to control the parameter p_m

Parameter	Value	Parameter	Value
Number of generations (<i>numGen</i>)	$1.5 \cdot 10^2, 3 \cdot 10^2$	Difference among samples (Δ)	0.1
Number of linguistic terms (<i>numTerms</i>)	7	Historical knowledge (k)	2, 5
Range of the parameter p_m	$[0, 1]$		

A-TSK, and FUZZY-B-TSK FLCs were defined by setting different values for the number of generations and the parameter k . Finally, the hyper-heuristics were applied with $n_h = 11$ low-level configurations. Low-level configurations used the parameterisation shown in Table 9.12, with each one using a different value for the parameter p_m .

Tables 9.15 and 9.16 show the statistics for the different configurations of the HH-ELI and HH-PROB hyper-heuristics, the FUZZY-A, FUZZY-B, FUZZY-A-TSK, and FUZZY-B-TSK FLCs and the diversity-based multi-objective MA—FIXED—for the Seattle and Denver instances, respectively, including dispersion measures like the Standard Deviation (SD) and the Coefficient of Variation (CV). The data in bold show, for each method, the configuration that achieved the lowest mean of the original objective value at the end of the executions. Moreover, the remaining configurations of a given method that are shown in bold did not exhibit statistically significant differences in comparison to the method that achieved the lowest mean for the original objective value. In contrast, the configurations of a given method which do not appear in bold presented statistically significant differences from the configuration with the lowest mean of the original objective value. In order to identify a given approach’s particular configuration, the values of its parameters reflect the name of the approach. For example:

- *HH-PROB_1500_5* is a configuration of the HH-PROB hyper-heuristic with a local stopping criterion equal to 1,500 evaluations and historical knowledge k equal to 5 decisions.
- *FUZZY-A_300_2* is a configuration of the FUZZY-A FLC that uses 300 generations as the local stopping criterion and historical knowledge equal to 2 decisions.
- *FIXED_0.9* is a configuration of the diversity-based multi-objective MA which applies the NM operator with probability p_m equal to 0.9.

We should note the following observations. With regard to parameter tuning, in the case of the Seattle instance, the configuration of the FIXED approach that obtained the lowest mean for the original objective value used the value 0.5—FIXED_0.5—

Table 9.15: Control and tuning of the parameter p_m – Seattle instance

Approach	Min.	First Qu.	Median	Mean	Third Qu.	Max.	SD	CV
HH-ELI_1500_2	547.1	589.1	624.3	644.5	675.3	889.2	84.4	13.1
HH-ELI_3000_2	506.0	594.9	655.1	651.7	698.8	870.8	87.2	13.4
HH-ELI_1500_5	511.0	610.9	672.4	662.8	726.4	799.6	72.7	11.0
HH-ELI_3000_5	525.9	595.8	637.3	646.2	686.8	896.8	79.6	12.3
HH-PROB_1500_2	530.9	629.2	668.2	669.8	708.0	855.4	74.7	11.1
HH-PROB_3000_2	523.6	578.2	650.3	644.0	695.5	775.1	71.4	11.1
HH-PROB_1500_5	515.6	608.2	665.4	675.2	749.5	888.4	88.8	13.1
HH-PROB_3000_5	534.5	590.1	642.9	647.2	686.7	790.8	65.5	10.1
FUZZY-A_150_2	529.7	610.2	664.1	669.5	720.8	785.2	68.8	10.3
FUZZY-A_300_2	504.1	564.5	645.2	643.1	680.0	882.8	89.4	13.9
FUZZY-A_150_5	517.4	584.6	636.7	643.0	691.0	780.0	69.0	10.7
FUZZY-A_300_5	505.3	602.2	646.3	658.7	687.8	851.9	81.3	12.3
FUZZY-B_150_2	463.0	609.2	651.1	659.2	721.4	814.1	87.6	13.3
FUZZY-B_300_2	471.7	619.0	678.2	669.8	712.6	807.4	77.4	11.6
FUZZY-B_150_5	519.9	616.7	658.3	660.7	701.9	804.5	70.0	10.6
FUZZY-B_300_5	497.6	616.3	673.4	663.8	704.1	795.5	68.5	10.3
FUZZY-A-TSK_150_2	468.8	603.9	653.1	669.7	729.3	939.0	97.9	14.6
FUZZY-A-TSK_300_2	475.8	569.7	641.9	636.4	691.7	810.2	78.9	12.4
FUZZY-A-TSK_150_5	536.4	620.1	662.1	685.0	741.0	854.6	88.5	12.9
FUZZY-A-TSK_300_5	502.0	629.8	668.7	667.1	706.7	834.6	78.6	11.8
FUZZY-B-TSK_150_2	526.5	605.4	645.6	655.3	691.0	876.0	78.0	11.9
FUZZY-B-TSK_300_2	474.0	587.0	649.4	645.0	701.1	815.9	88.2	13.7
FUZZY-B-TSK_150_5	533.1	622.6	672.8	677.6	736.2	862.7	87.4	12.9
FUZZY-B-TSK_300_5	452.7	613.4	668.7	668.6	713.2	855.7	86.3	12.9
FIXED_0	587.9	684.6	736.3	740.4	775.3	974.4	84.4	11.4
FIXED_0.1	525.5	660.7	712.7	716.2	785.0	878.6	92.2	12.9
FIXED_0.2	547.9	631.1	687.4	684.8	727.3	797.8	62.4	9.1
FIXED_0.3	562.7	613.0	664.9	688.7	748.5	941.1	90.4	13.1
FIXED_0.4	515.2	631.8	680.9	679.3	736.2	809.2	72.1	10.6
FIXED_0.5	521.3	607.7	667.4	667.3	714.3	842.6	76.7	11.5
FIXED_0.6	557.9	650.9	676.6	694.1	723.6	834.7	65.4	9.4
FIXED_0.7	551.8	642.1	686.6	678.0	717.4	813.2	62.2	9.2
FIXED_0.8	504.1	649.2	676.9	679.7	716.1	806.0	60.8	8.9
FIXED_0.9	541.5	638.0	695.4	697.0	751.5	893.5	78.1	11.2
FIXED_1.0	546.6	683.5	705.9	713.6	750.8	864.6	62.9	8.8

for the parameter p_m of the NM operator, while in the case of the Denver instance this value was equal to 0.8—FIXED_0.8. This fact confirms that the most suitable value for a parameter changes depending on the problem and/or instance being solved. Moreover, these configurations exhibited statistically significant differences as compared to others. In the case of the Seattle instance, there were statistically significant differences with 3 configurations, while for the Denver instance, there were differences with 5 configurations. We can therefore observe that the parameter p_m is more sensitive to changes in its value when the NM operator is applied to the Denver instance, so it is even more important to select the appropriate values in this particular case.

Considering the control methods applied to the Seattle instance, the only statistically significant difference appeared among the configurations of the FUZZY-A-TSK

Table 9.16: Control and tuning of the parameter p_m – Denver instance

Approach	Min.	First Qu.	Median	Mean	Third Qu.	Max.	SD	CV
HH-ELI_1500_2	84228.4	84811.6	85099.4	85285.8	85798.8	86742.7	673.9	0.8
HH-ELI_3000_2	84019.6	84840.6	85113.6	85137.3	85373.8	86446.6	579.7	0.7
HH-ELI_1500_5	83996.6	84792.3	85170.2	85251.3	85531.2	87590.5	790.0	0.9
HH-ELI_3000_5	83933.5	84840.5	85246.2	85229.9	85565.8	86405.4	625.6	0.7
HH-PROB_1500_2	83833.4	84918.7	85306.4	85397.2	85818.7	87083.6	805.7	0.9
HH-PROB_3000_2	84274.6	84916.2	85455.6	85560.5	86213.6	87661.3	874.5	1.0
HH-PROB_1500_5	83789.9	84599.5	85013.2	85058.1	85380.7	87005.8	714.7	0.8
HH-PROB_3000_5	84257.9	84929.5	85660.6	85529.5	85965.6	87215.7	761.1	0.9
FUZZY-A_150_2	84442.6	84884.3	85218.1	85364.2	85783.4	86857.3	701.7	0.8
FUZZY-A_300_2	84277.6	84877.2	85139.4	85323.3	85812.7	88066.0	814.3	1.0
FUZZY-A_150_5	84194.4	84992.8	85356.4	85413.8	85690.5	86863.9	647.3	0.8
FUZZY-A_300_5	83594.4	84508.5	84979.9	85136.9	85445.8	87149.7	803.8	0.9
FUZZY-B_150_2	84000.7	84937.9	85567.4	85509.3	85995.9	87174.2	808.1	0.9
FUZZY-B_300_2	84245.1	84774.4	85066.2	85194.8	85625.7	87106.1	622.5	0.7
FUZZY-B_150_5	84118.2	84627.1	84975.1	85190.6	85559.6	87213.2	841.0	1.0
FUZZY-B_300_5	84004.6	84493.2	85035.4	84986.1	85407.6	85794.5	536.4	0.6
FUZZY-A-TSK_150_2	84190.2	85019.9	85342.9	85609.9	86099.7	87926.5	897.2	1.0
FUZZY-A-TSK_300_2	84455.0	84938.8	85432.2	85403.1	85611.6	87481.4	646.2	0.8
FUZZY-A-TSK_150_5	84098.0	84787.6	85245.9	85375.6	85921.6	87115.1	790.0	0.9
FUZZY-A-TSK_300_5	84149.1	84764.2	84975.3	85196.7	85440.2	87088.2	681.3	0.8
FUZZY-B-TSK_150_2	84342.0	84844.9	85157.6	85487.2	85776.0	88404.1	1006.3	1.2
FUZZY-B-TSK_300_2	83993.9	84755.4	85113.0	85169.1	85471.8	86655.7	661.8	0.8
FUZZY-B-TSK_150_5	83774.2	84690.4	85140.4	85087.2	85392.3	86347.5	553.3	0.7
FUZZY-B-TSK_300_5	84104.5	84830.8	85250.9	85426.2	86104.6	86974.6	759.2	0.9
FIXED_0	85243.4	86554.4	87230.3	87071.1	87572.1	88744.8	838.7	1.0
FIXED_0.1	84765.0	85682.4	86444.1	86475.8	87050.8	88501.1	980.1	1.1
FIXED_0.2	84552.3	85955.6	86276.4	86392.5	86867.1	89207.3	952.2	1.1
FIXED_0.3	84432.4	85293.9	85796.1	85946.2	86550.9	88072.8	924.0	1.1
FIXED_0.4	84447.7	85236.0	85824.9	85809.0	86155.5	87545.7	771.5	0.9
FIXED_0.5	84055.0	84924.4	85552.8	85704.0	86210.9	87405.7	961.4	1.1
FIXED_0.6	83969.0	84940.1	85429.2	85541.8	86073.9	88281.3	977.2	1.1
FIXED_0.7	84075.3	84836.5	85478.0	85481.3	85950.7	87432.7	748.9	0.9
FIXED_0.8	83400.7	84820.1	85295.2	85367.0	85750.7	87628.2	829.2	1.0
FIXED_0.9	84441.6	85233.9	85569.0	85603.4	85816.2	87884.8	673.6	0.8
FIXED_1.0	84292.5	85178.5	85409.2	85418.6	85683.7	87152.4	498.5	0.6

FLC. For the Denver instance, the only statistically significant differences appeared among the configurations of the HH-PROB hyper-heuristic and the FUZZY-B and FUZZY-A-TSK FLCs. This means that both the hyper-heuristics and the FLCs are robust enough from the point of view of their parameters. If these parameters are modified, these changes are not going to greatly determine the performance of the control strategy. Thus, the parameters of the control methods do not add more burdens to the configuration of the diversity-based multi-objective MA.

If for each of the six control methods exposed herein the corresponding configuration that achieved the lowest mean of the original objective value is considered, no statistically significant differences are evident among them. This was the case for both the Seattle and Denver instances. As a result, hyper-heuristics or FLCs can be applied indistinctly to control the parameter p_m without drastically affecting the

Table 9.17: Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter p_m – Seattle instance

Approach	Number of configurations
HH-ELI	10
HH-PROB	9
FUZZY-A	10
FUZZY-B	3
FUZZY-A-TSK	10
FUZZY-B-TSK	6

Table 9.18: Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter p_m – Denver instance

Approach	Number of configurations
HH-ELI	10
HH-PROB	10
FUZZY-A	8
FUZZY-B	11
FUZZY-A-TSK	8
FUZZY-B-TSK	10

quality of either city’s frequency plans.

In order to compare parameter control and parameter tuning, Tables 9.17 and 9.18 show the number of configurations for the FIXED approach that exhibited statistically significant differences with each of the six control schemes for Seattle and Denver, respectively. To carry out the statistical comparison, the configuration for each control method that obtained the lowest mean of the original objective value at the end of the executions was used.

For the Seattle instance, the HH-ELI hyper-heuristic and the FUZZY-A and FUZZY-A-TSK FLCs were able to outperform 10 configurations of the FIXED approach, while in the case of Denver, both hyper-heuristics and the FUZZY-B-TSK FLC outperformed 10 configurations, whereas the FUZZY-B FLC was able to outperform 11 configurations. Moreover, no configuration of the FIXED scheme was able to statistically outperform any control method for either instance. Consequently, the advantages of using parameter control instead of parameter tuning are clear. Just one execution of the control schemes was needed to yield similar or even better solutions than those obtained using the best-behaved configuration of the diversity-based multi-objective MA. It is important to note that in order to find the best-behaved

Table 9.19: Parameterisation of the diversity-based multi-objective memetic algorithm for different values of the parameter R

Parameter	Value	Parameter	Value
Stopping criterion	$1.5 \cdot 10^5$ evals.	Crossover rate (p_c)	1
Population size (N)	10 individuals	Mutation rate (p_m)	0.5 (Seattle), 0.8 (Denver)
Crossover operator	UX (Seattle), IX (Denver)	NM operator steps (R)	1, 2, 3, ..., 14, 15
Auxiliary objective	DCN (Seattle), ADI (Denver)		

Table 9.20: Parameterisation of the hyper-heuristics HH-ELI and HH-PROB to control the parameter R

Parameter	Value	Parameter	Value
Local stopping criterion	$1.5 \cdot 10^3, 3 \cdot 10^3$ evals.	Minimum selection rate (β)	0.1
Number of low-level configs. (n_h)	15 configs.	Historical knowledge (k)	2, 5

configuration of the diversity-based multi-objective MA, 11 different parameterisations were tested by varying the value of the parameter p_m . With this in mind, the benefits of parameter control over parameter tuning are even higher.

Finally, it is worth mentioning that the best-known frequency plan for Seattle, which was obtained during the fourth experiment performed in Section 9.4.4, was improved upon by the FUZZY-B-TSK control scheme when the parameter p_m was adapted, with the original objective value decreasing from 486.6 to 452.7, as shown in Table 9.15.

9.4.6 Adapting the parameter R of the Neighbour-based Mutation Operator

This experiment was based on the application of the control approaches based on FLCs and hyper-heuristics detailed in Section 9.3.1 to the parameter R of the NM operator in order to solve the FAP. As was the case with the previous experiment, the diversity-based multi-objective MA presented in Section 9.2.2 was also run. Nevertheless, its configurations were obtained by varying the parameter R while holding the value of p_m constant. The main objective was to analyse the behaviour of the different control schemes when adapting the parameter R . These control techniques were also compared to parameter tuning.

The same parameterisations from the previous experiment were used, though in this case the value of parameter p_m was held constant. Several values for the parameter R were also tested. Table 9.19 shows the parameterisation of the diversity-based multi-objective MA for this experiment. In this case, 15 different configurations of

Table 9.21: Parameterisation of the fuzzy logic controllers FUZZY-A, FUZZY-B, FUZZY-A-TSK, and FUZZY-B-TSK to control the parameter R

Parameter	Value	Parameter	Value
Number of generations (<i>numGen</i>)	$1.5 \cdot 10^2, 3 \cdot 10^2$	Difference among samples (Δ)	1
Number of linguistic terms (<i>numTerms</i>)	7	Historical knowledge (k)	2, 5
Range of the parameter R	[1, 15]		

this approach were executed using different values for the parameter R .

The parameterisations of the control approaches are shown in Tables 9.20 and 9.21 for the hyper-heuristics and the FLCs, respectively. As in the previous experiment, four different configurations of each control method were applied. In the case of the hyper-heuristics, they were defined with $n_h = 15$ low-level configurations, with each one taking on a different value for the parameter R and using the parameterisation shown in Table 9.19.

Tables 9.22 and 9.23 show the statistics obtained for the Seattle and Denver instances, respectively, by the different configurations of the hyper-heuristics, the FLCs, and the diversity-based multi-objective MA. Note the following regarding the setting of parameter R : In terms of parameter tuning, the configuration of the FIXED approach that yielded the lowest mean for the original objective value for the Seattle instance used the NM operator with the parameter R equal to 7—FIXED_7. In the case of Denver, FIXED_6 was the most suitable configuration of the diversity-based multi-objective MA. Both configurations exhibited statistically significant differences as compared to other configurations of the FIXED scheme. In the case of Seattle, there were differences with 7 configurations, while for Denver, the number of statistically significant differences was equal to 6.

Similar conclusions to those extracted for parameter p_m can be drawn for parameter R . The study involving parameter tuning reveals that the most appropriate value for R depends on the problem and/or instance being solved. A statistical comparison shows that for this particular parameter, the number of statistical differences between the FIXED scheme configuration that obtained the lowest mean for the original objective value and other FIXED configurations is noticeable for both instances. As a result, we can conclude that the parameter R is also sensitive to changes in its value, as was the case with p_m .

With regard to parameter control, and considering the Seattle instance, the configurations did not exhibit statistically significant differences among them, while in the case of Denver, only statistically significant differences appeared between the configurations of the HH-ELI hyper-heuristic and the FUZZY-A-TSK FLC. Once

more, both hyper-heuristics and FLCs demonstrated their robustness, with R being adapted in this case, since their performance was not significantly affected by changes in their parameter values, as occurred when p_m was adapted.

No statistically significant differences appeared among the configurations that obtained the lowest mean for the original objective value for each of the six control schemes. This was the case for both instances. Consequently, not only can the parameter p_m be controlled by hyper-heuristics or FLCs indistinctly, but so can the parameter R . Thus, we can confirm the generality of both control methods, which can adapt continuous and discrete numeric parameters successfully.

In order to compare parameter control and parameter tuning, Tables 9.24 and 9.25 show, for Seattle and Denver respectively, the number of FIXED scheme configurations that exhibited statistically significant differences with each of the six control techniques. To statistically compare each of the six control methods, the configuration that obtained the lowest mean for the original objective value was selected. For Seattle, the HH-ELI hyper-heuristic was able to outperform 13 FIXED scheme configurations, followed by the FUZZY-B-TSK approach, which was able to outperform 12 configurations. In the case of Denver, the superiority of the control techniques is clear as they were able to outperform every FIXED scheme configuration. As in the previous experiment, no configuration of the FIXED scheme was able to statistically outperform any control method for either instance. Consequently, as was the case with the parameter p_m , the benefits of adapting the parameter R instead of tuning it are also clear. A single execution of the hyper-heuristics or the FLCs yielded frequency plans for the two cities in question that were similar to or even better than those provided by the best-behaved configurations of the diversity-based multi-objective MA. To find the best-behaved configuration, 15 different parameterisations of the diversity-based multi-objective MA were tested by modifying the value of R . Taking this fact into consideration, the benefits of parameter control over parameter tuning are even greater.

Finally, we would like to note that the best-known frequency plan for Denver, which was obtained during the fourth experiment performed in Section 9.4.4, was improved upon by the FUZZY-B control scheme when the parameter R was adapted, with the original objective value decreasing from 83,340.2 to 83,280.9, as shown in Table 9.23.

Table 9.22: Control and tuning of the parameter R – Seattle instance

Approach	Min.	First Qu.	Median	Mean	Third Qu.	Max.	SD	CV
HH-ELI_1500_2	496.5	591.6	636.1	634.8	675.8	738.4	61.2	9.6
HH-ELI_3000_2	558.2	593.5	646.4	656.3	711.1	825.0	70.7	10.8
HH-ELI_1500_5	534.3	588.7	635.6	644.2	680.7	813.5	72.1	11.2
HH-ELI_3000_5	501.8	590.0	635.2	650.7	704.8	845.8	81.7	12.6
HH-PROB_1500_2	527.1	603.4	629.4	645.1	697.6	775.3	64.2	10.0
HH-PROB_3000_2	540.6	611.8	649.5	674.5	750.2	819.4	80.5	11.9
HH-PROB_1500_5	514.5	620.8	634.1	646.6	697.7	764.4	63.9	9.9
HH-PROB_3000_5	522.6	597.9	646.1	653.9	704.6	854.6	80.4	12.3
FUZZY-A_150_2	486.5	587.1	627.2	641.4	694.9	821.1	77.4	12.1
FUZZY-A_300_2	503.6	596.3	644.5	649.4	696.7	794.5	74.6	11.5
FUZZY-A_150_5	508.8	585.2	627.4	647.7	697.0	820.5	88.0	13.6
FUZZY-A_300_5	543.0	597.1	635.2	655.9	698.2	880.3	79.7	12.2
FUZZY-B_150_2	473.9	563.4	617.8	638.7	701.7	832.2	95.4	14.9
FUZZY-B_300_2	516.0	582.8	654.8	650.6	706.4	834.0	82.3	12.7
FUZZY-B_150_5	496.2	607.2	642.4	639.1	683.6	809.1	65.9	10.3
FUZZY-B_300_5	520.9	590.3	623.9	642.0	701.2	794.4	73.6	11.5
FUZZY-A-TSK_150_2	494.4	596.1	634.1	657.9	715.4	881.0	91.5	13.9
FUZZY-A-TSK_300_2	522.6	614.8	647.9	659.9	722.1	781.2	70.5	10.7
FUZZY-A-TSK_150_5	514.4	605.3	641.9	645.5	706.4	787.0	76.9	11.9
FUZZY-A-TSK_300_5	550.7	622.8	662.2	660.7	701.7	757.3	52.2	7.9
FUZZY-B-TSK_150_2	452.9	589.2	621.2	635.9	689.4	775.9	74.3	11.7
FUZZY-B-TSK_300_2	484.0	594.3	642.8	646.6	692.9	813.7	72.4	11.2
FUZZY-B-TSK_150_5	504.7	609.2	663.9	662.6	702.1	841.8	70.9	10.7
FUZZY-B-TSK_300_5	466.8	560.9	632.5	643.7	705.2	795.8	94.9	14.7
FIXED_1	549.1	673.6	728.6	744.6	814.0	949.0	97.9	13.2
FIXED_2	592.5	675.1	726.7	727.8	778.7	887.7	74.4	10.2
FIXED_3	550.6	667.7	707.7	725.3	767.8	968.0	83.0	11.4
FIXED_4	566.6	644.0	687.8	689.5	722.7	893.6	73.8	10.7
FIXED_5	566.8	640.5	681.7	693.9	721.0	908.3	80.4	11.6
FIXED_6	547.0	635.4	670.0	675.8	709.5	806.9	60.6	9.0
FIXED_7	580.3	624.1	641.6	658.4	683.4	830.4	57.3	8.7
FIXED_8	529.8	637.1	682.5	668.2	713.5	782.7	69.9	10.5
FIXED_9	555.0	619.4	667.2	662.1	709.5	771.1	59.5	9.0
FIXED_10	533.3	629.1	682.8	684.2	723.3	843.7	74.4	10.9
FIXED_11	543.8	634.9	667.3	678.9	732.4	863.0	76.2	11.2
FIXED_12	525.9	647.6	683.0	681.6	728.1	833.6	73.2	10.7
FIXED_13	597.7	665.3	698.9	712.2	737.5	982.5	74.2	10.4
FIXED_14	621.5	674.8	731.2	738.5	795.2	894.2	80.0	10.8
FIXED_15	613.7	709.6	737.3	739.8	780.6	879.2	63.8	8.6

Table 9.23: Control and tuning of the parameter R – Denver instance

Approach	Min.	First Qu.	Median	Mean	Third Qu.	Max.	SD	CV
HH-ELI_1500_2	83780.7	84658.2	85041.6	85064.2	85568.1	86324.1	609.5	0.7
HH-ELI_3000_2	83876.2	84446.8	84856.4	84846.1	85196.2	86552.9	638.7	0.8
HH-ELI_1500_5	84037.3	84593.2	84899.0	85090.5	85412.7	87154.3	731.3	0.9
HH-ELI_3000_5	83740.6	84784.8	84974.7	85184.5	85610.9	86783.9	654.8	0.8
HH-PROB_1500_2	83914.2	84568.8	85127.7	85240.3	85715.4	87159.4	844.1	1.0
HH-PROB_3000_2	84352.3	84667.8	84962.4	85177.4	85660.6	86766.0	691.9	0.8
HH-PROB_1500_5	83796.0	84441.5	84876.1	84973.9	85335.6	86323.2	614.8	0.7
HH-PROB_3000_5	83548.0	84639.6	85026.2	84965.3	85269.4	86455.1	619.5	0.7
FUZZY-A_150_2	83829.4	84615.1	84956.8	85081.4	85354.6	86533.0	634.9	0.7
FUZZY-A_300_2	83750.1	84533.9	84988.5	85044.9	85432.5	87192.5	750.9	0.9
FUZZY-A_150_5	83852.6	84534.0	84962.4	84967.1	85313.5	86897.6	696.0	0.8
FUZZY-A_300_5	83335.4	84545.0	85102.9	85032.9	85524.4	86584.4	792.8	0.9
FUZZY-B_150_2	83280.9	84446.4	85033.5	84881.4	85380.0	86181.9	694.2	0.8
FUZZY-B_300_2	83377.4	84515.4	84888.5	84915.7	85359.1	86106.8	592.1	0.7
FUZZY-B_150_5	83955.2	84549.1	85145.9	85228.5	85749.9	87100.4	831.0	1.0
FUZZY-B_300_5	83727.2	84481.1	84939.1	84942.2	85313.6	86433.9	660.5	0.8
FUZZY-A-TSK_150_2	83506.5	84567.6	84930.4	84932.3	85338.2	85976.4	583.9	0.7
FUZZY-A-TSK_300_2	83445.6	84407.0	84753.3	84898.6	85291.9	87418.1	773.8	0.9
FUZZY-A-TSK_150_5	84195.8	84687.3	84973.2	85117.3	85357.1	86971.2	652.1	0.8
FUZZY-A-TSK_300_5	83442.2	84748.0	85083.4	85220.6	85577.7	87000.9	768.0	0.9
FUZZY-B-TSK_150_2	83402.6	84488.6	84843.3	84936.9	85290.2	86678.3	725.2	0.9
FUZZY-B-TSK_300_2	83878.6	84478.1	84962.6	85034.4	85423.3	86530.3	737.4	0.9
FUZZY-B-TSK_150_5	83672.0	84485.3	84937.9	85029.2	85678.6	86965.0	785.8	0.9
FUZZY-B-TSK_300_5	83702.2	84376.8	84888.1	84923.2	85416.3	86759.9	728.7	0.9
FIXED_1	84885.9	85695.1	86231.9	86206.7	86667.4	87291.9	681.1	0.8
FIXED_2	83747.4	85352.1	85747.4	85716.0	86282.5	86894.6	718.9	0.8
FIXED_3	84558.3	84876.7	85325.8	85610.8	86334.2	87403.0	872.4	1.0
FIXED_4	84001.3	85083.5	85225.4	85423.6	85848.8	87109.2	768.8	0.9
FIXED_5	84760.0	85151.6	85631.0	85588.6	85981.8	86745.8	503.2	0.6
FIXED_6	84483.9	84888.0	85254.9	85333.7	85751.2	86899.4	606.2	0.7
FIXED_7	84051.4	85088.9	85528.4	85505.9	85938.8	87282.4	726.6	0.8
FIXED_8	84296.1	85052.9	85382.1	85449.0	85820.9	87139.2	574.2	0.7
FIXED_9	84357.2	85129.6	85339.9	85428.5	85655.3	87110.0	595.5	0.7
FIXED_10	84693.2	85277.0	85434.1	85652.1	86066.5	87222.2	585.1	0.7
FIXED_11	84771.0	85388.2	85560.2	85671.2	86077.8	86910.9	558.9	0.7
FIXED_12	84345.7	85203.1	85500.4	85538.7	85819.2	86688.2	590.3	0.7
FIXED_13	84587.0	85268.0	85516.4	85621.5	85869.9	86981.9	606.5	0.7
FIXED_14	84627.3	85159.4	85740.0	85703.3	85979.4	87857.6	676.3	0.8
FIXED_15	84955.6	85768.8	86036.3	86154.7	86550.7	87522.7	649.9	0.8

Table 9.24: Number of fixed configurations outperformed by the parameter control approaches adapting the parameter R – Seattle instance

Approach	Number of configurations
HH-ELI	13
HH-PROB	10
FUZZY-A	10
FUZZY-B	10
FUZZY-A-TSK	9
FUZZY-B-TSK	12

Table 9.25: Number of fixed configurations outperformed by the parameter control approaches by adapting the parameter R – Denver instance

Approach	Number of configurations
HH-ELI	15
HH-PROB	15
FUZZY-A	15
FUZZY-B	15
FUZZY-A-TSK	15
FUZZY-B-TSK	15

Two-dimensional Packing Problem

Packing problems are a class of optimisation problems that involve packing a set of objects together as densely as possible. They are highly related to cutting problems, whose main goal is to cut large stock sheets into a set of smaller pieces. In many cases, both problems have been analysed together, being referred to as cutting and packing problems. Both problems have been shown to be combinatorial *NP*-hard problems. Therefore, obtaining high-quality solutions is a complex task. However, there is considerable interest in solving them because they are related to real-life packaging, storage, and transportation issues. Therefore, they have many applications and are widely used within more complex systems, such as filling containers and trucks, loading pallets, and optimising the layout of electrical circuits, among others. Cutting and packing problems can be classified [216, 346] according to several characteristics, including: the number of dimensions—1D, 2D, 3D—, the number of available patterns, the shape of the patterns—regular or irregular—, the orientation, and the objective to be optimised. Depending on these features, several variants of the problem can be defined. Some of the most popular ones are 2D strip packing [93], constrained 2D cutting stock [340], knapsack problems [233], packing with cost [55], and online packing [304]. Within each category there are also several different formulations. In the GECCO 2008 competition session¹, a variant of a *Two-Dimensional Packing Problem (2DPP)* was proposed. It was a reformulation of a packing problem designed with the aim of hindering the achievement of optimal values and increasing the size of the search space. While it may be difficult to imagine direct practical applications of this particular variant of a 2D packing problem, it is hard and complex enough that it can be used to check the advan-

¹<http://www.isgec.org/gecco-2008/competitions.html>

tages and drawbacks of a given optimisation scheme. This was the main reason for considering this formulation of the problem in this dissertation. Moreover, previous results obtained using different optimisation methods can be used for comparison purposes [205].

There are many proposals designed to deal with packing problems. Among them, several exact approaches have been proposed [232]. Usually, the time associated with these algorithms is very large. Therefore, in order to reduce the execution time, some parallel exact approaches have also been designed [35, 204]. However, since packing problems usually involve a large search space, exact approaches are practically unaffordable for many real-world instances. In order to handle large instances, a wide variety of approximate algorithms have been tested. For instance, an approach based on an ACO algorithm was used to deal with a multi-objective version of a packing problem in [197], whereas in [236] a GA was applied to a single-objective variant. MAs have also yielded very promising results for packing problems [358]. Regarding the 2DPP defined in the GECCO 2008 competition session, several proposals have also been tested. During the contest, the two best-behaved approaches were based on MAs. Prior to the proposals defined herein, the approach that had yielded the best results for the competition session instance was the EAIPS, which was described in Section 9.2.1. However, the individual learning process considered was specifically designed to deal with the 2DPP. In [205], the DYN model was executed considering different low-level configurations of the EAIPS in an effort to obtain high-quality results faster. It was able to attain the best-known solution for the competition instance.

The rest of the chapter is organised as follows. In Section 10.1 the mathematical formulation of the 2DPP is described. Then, the different optimisation schemes defined for dealing with this problem are presented in Section 10.2. Afterwards, the control approaches that are proposed to adapt the parameters of said optimisation schemes are introduced in Section 10.3. Finally, Section 10.4 details the experimental evaluation conducted using the proposed optimisation schemes and parameter control approaches on several instances of the 2DPP, including the competition instance.

10.1 Formal Definition

The 2DPP is a two-dimensional variant of a packing problem. Since the problem has been tackled using many different approaches and its search space is vast, it can be used as a benchmark problem. Problem instances are described by:

Pairs Table (v)		
a	b	$v(a, b)$
2	1	100
1	2	150
1	3	200
3	2	175
1	1	25
...

Grid (G)	
1	2
3	1

Objective Value = $v(1,2) + v(2,1) + v(2,3) + v(3,2) + v(3,1) + v(1,3) + v(1,1)$

Figure 10.1: Assignment of the objective function value for the Two-Dimensional Packing Problem

- The sizes of a rectangular grid: X, Y .
- The maximum number which can be assigned to a grid position: M . The value assigned to each grid location is an integer in the range $[0, M]$.
- The score or value associated with the appearance of each pair (a, b) where $a, b \in [0, M]$: $v(a, b)$. Note that $v(a, b)$ is not necessarily equal to $v(b, a)$.

A candidate solution is obtained by assigning a number to each grid position. Thus, the search space consists of $(M + 1)^{X \cdot Y}$ candidate solutions. The aim of the problem proposed is to best pack a grid so that the sum of the point scores for every pair of adjacent numbers is maximised. Two positions are considered to be adjacent if they are neighbours in the same row, column, or diagonal of the grid. Once a particular pair is collected, it cannot be collected a second time in the same grid.

Mathematically, the goal of the problem is to find the grid G which maximises the following objective function f :

$$f = \sum_{a=0}^M \sum_{b=0}^M v_2(a, b) \quad (10.1)$$

where

$$v_2(a, b) = \begin{cases} 0 & \text{if } (a, b) \text{ are not adjacent in } G \\ v(a, b) & \text{if } (a, b) \text{ are adjacent in } G \end{cases} \quad (10.2)$$

Figure 10.1 illustrates the assignment of the objective function value for a candidate solution of a 2×2 grid. Note that although the pairs $(1, 2)$ and $(2, 1)$ are repeated in the grid, they are only considered once when computing the objective function.

10.2 Optimisation Schemes

This section is devoted to defining the different optimisation schemes that are applied in this thesis to solve the 2DPP. Particularly, the EAIPS is considered. Moreover, several diversity-based objectives, as well as a multi-objectivisation method, are used together with novel multi-objective MAs based on the well-known NSGA-II and SPEA2. All the above optimisation schemes incorporate an individual learning procedure that was specifically designed to deal with the 2DPP. Lastly, different parallel homogeneous island-based models are also taken into account.

10.2.1 Evolutionary Algorithm with Increasing Population Size

The EAIPS detailed in Section 9.2.1, which was applied as an optimisation scheme to deal with the FAP in the previous chapter, is considered herein to address the 2DPP. Recall that the EAIPS is a single-objective MA that combines an EA with a $(1 + 1)$ survivor selection operator. Additionally, it applies the variation phase and the individual learning procedure described in Sections 10.2.3 and 10.2.4, respectively. The algorithm starts its execution as a trajectory-based approach, though it increases the population size to escape from local optima when stagnation is detected, thus behaving as a population-based algorithm. In order to complete the definition of the EAIPS, the individuals were encoded as two-dimensional arrays of integer values G , where $G(x, y)$ is the number assigned to the grid position (x, y) .

10.2.2 Diversity-based Multi-objective Memetic Algorithms and Multi-objectivisation by Aggregation

The novel diversity-based multi-objective MAs used herein to address the 2DPP are based on the NSGA-II and the SPEA2, which were described in Section 2.4.1. The main difference with respect to the original MOEAs is that the individual learning strategy detailed in Section 10.2.4 is applied to every generation after the variation

Algorithm 12 Pseudocode of the memetic algorithm based on the Strength Pareto Evolutionary Algorithm 2

- 1: **Initialisation.** Generate the initial parent population P_0 with N individuals, and create the empty archive \overline{P}_0 . Assign $t = 0$.
 - 2: **while** (not stopping criterion) **do**
 - 3: **Evaluation.** Evaluate all individuals in the parent population by calculating the objective functions.
 - 4: **Fitness assignment.** Calculate the fitness values of individuals in P_t and \overline{P}_t . For each individual i , calculate the raw fitness raw_i and the density estimate $density_i$.
 - 5: **Environmental Selection.** Copy non-dominated individuals which belong to P_t and \overline{P}_t to \overline{P}_{t+1} . If $|\overline{P}_{t+1}| > \overline{N}$ reduce \overline{P}_{t+1} by means of the truncation operator. Otherwise, if $|\overline{P}_{t+1}| < \overline{N}$, fill \overline{P}_{t+1} with dominated individuals belonging to P_t and \overline{P}_t , considering their fitness.
 - 6: **Parent selection.** Perform deterministic binary tournament selection with replacement on \overline{P}_{t+1} to fill the mating pool with N parents.
 - 7: **Variation.** Apply crossover and mutation operators, with probabilities p_c and p_m , to the mating pool so as to obtain $M = N$ offspring.
 - 8: **Learning process.** Apply the individual learning process to every individual in P_{t+1} .
 - 9: **Survivor selection.** Set P_{t+1} to the offspring population.
 - 10: $t = t + 1$
 - 11: **end while**
-

phase. This is evident in the Algorithms 10 and 12, which show the operation of the MAs based on the NSGA-II and the SPEA2, respectively. Since diversity-based multi-objective MAs are used, an additional diversity-based objective function must be considered together with the original objective function of the 2DPP defined in Equation 10.1. Different encoding-independent and genotypic diversity-based objectives were taken into account for both MAs. In particular, the diversity-based objectives tested were *random*, *inversion*, ADI, DBI, and DCN—Section 3.1. Additionally, the diversity-based objectives with parameters DBI-THR and DCN-THR—Section 5.1—were also applied. We should note that in the case of the NSGA-II, the diversity-based objectives are calculated using the individuals in the population. When the SPEA2 is applied, however, the individuals in the population and the archive are taken into account. So as to completely define the diversity-based multi-objective MAs, individuals were encoded by means of two-dimensional arrays of integer values G , where $G(x, y)$ is the number assigned to the grid position (x, y) . Finally, the genetic operators applied during the variation stage of the diversity-based multi-objective MAs are described in Section 10.2.3.

In addition to the application of the diversity-based multi-objective MAs described above, multi-objectivisation by aggregation of helper-objectives is also employed herein to solve the 2DPP. To do so, the multi-objective MAs described in the Algorithms 10 and 12 are combined together with a helper-objective, instead of using a diversity-based objective function. This helper-objective makes use of problem-dependent information and is called *Dependent*. In order to calculate this helper-objective, the original objective function of the 2DPP, denoted by f , is decomposed into two separate functions, f_0 and f_1 , such that $f = f_0 + f_1$. The decomposition is performed as follows. First, a table containing all possible pairs whose score is not equal to zero is calculated. Then, this table is sorted based on the score of the appearance of each pair ρ . The resultant position of each pair ρ in the sorted table is denoted by i_ρ . The value associated with each pair ρ is taken into account in the function f_{obj} , where $obj = i_\rho \bmod 2$. Hence, f_0 is used as the helper-objective. Likewise, f_1 could have been used as the helper-objective. Finally, all the remaining components, such as the encoding of the individuals, the genetic operators, and the individual learning procedure, are the same as those applied with the diversity-based multi-objective MAs.

10.2.3 Genetic Operators

A variation phase that involves applying a crossover operator and then a mutation operator is performed in every generation of the MAs described in the previous sections. Several variation operators were tested in [205]. The best-behaved ones are considered herein. On the one hand is the crossover operator SSX, whose operation was explained in Section 2.2.2. Recall that this operator is applied with probability p_c . On the other hand is the mutation operator *Uniform Mutation with Domain Information* (UMD). Every gene is mutated with a probability between min_p_m and max_p_m . Additionally, in order to make a new assignment to a certain gene, a random value is selected from among those that produce a non-zero increase in the original objective value.

10.2.4 Individual Learning Strategy

The individual learning procedure described in this section, which is based on a stochastic hill climbing local search, was specifically designed to deal with the 2DPP [205]. It is used in every generation of the MAs introduced in Sections 10.2.1

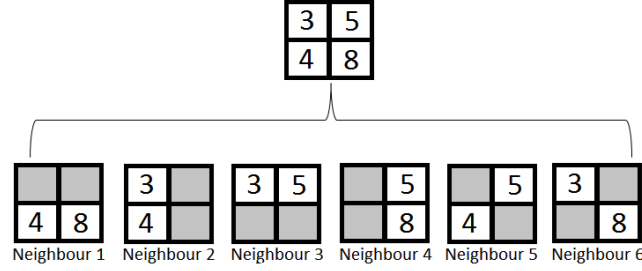


Figure 10.2: Generation of new neighbours by the learning process

and 10.2.2. As a result, a significant effort was made to make this procedure as efficient as possible.

The operation of this individual learning strategy is as follows. For each pair of adjacent grid positions (i, j) and (k, l) , a neighbour is considered. This is illustrated in Figure 10.2. Every neighbour is constituted by assigning the best possible values to the positions (i, j) and (k, l) —shaded positions in Figure 10.2—while leaving intact the assignments in any other grid location. In order to assign the best values to both locations, the trivial solution consists of enumerating all possible pairs so that the best one can subsequently be chosen. This approach is computationally too expensive, so a mechanism is used to prune the values explored. First, all the possible assignments $n \in [0, M]$ to the grid position (i, j) are considered, and the contribution of each assignment $v_{ij}(n)$, assuming position (k, l) is unassigned, is calculated. The same process is performed for position (k, l) , assuming position (i, j) is unassigned, which thus calculates $v_{kl}(n)$. The contribution to the original objective function obtained by assigning a value a to position (i, j) , and a value b to position (k, l) , is given by:

$$v_{ij}(a) + v_{kl}(b) + v'(a, b) - v_{rep} \quad (10.3)$$

where $v'(a, b) = v(a, b) + v(b, a)$ if the pair (a, b) was not already in the grid, or 0 if it was, and v_{rep} is the value associated with pairs that are constituted by both the assignment of the value a to (i, j) and the assignment of the value b to (k, l) , which must be considered only once. An upper bound for this contribution is given by:

$$v_{ij}(a) + v_{kl}(b) + \min(bestV(a), bestV(b)) \quad (10.4)$$

where $bestV(n)$ is the maximum value associated with any pair (n, m) , $m \in [0, M]$, i.e.:

$$bestV(n) = \max\{v(n, m) + v(m, n)\} \quad (10.5)$$

If $bestObj$ is the best original objective value currently achieved for an assignment of the positions (i, j) and (k, l) , the only values a' , b' that must be considered are those that satisfy the following inequality:

$$v_{ij}(a') + v_{kl}(b') + \min(bestV(a'), bestV(b')) > bestObj \quad (10.6)$$

Omitting the values at which the above inequality is not satisfied drastically reduces the neighbourhood to be considered, resulting in significant time savings.

Since stochastic hill climbing is used, the order in which neighbours are analysed is determined randomly. The local search moves to the first newly generated neighbour that improves the current solution. Finally, the learning process stops when none of the neighbours improves the current solution.

10.2.5 Parallel Homogeneous Island-based Models

The parallelisation of the EAIPS described in Section 10.2.1, as well as the parallelisation of the diversity-based multi-objective MA built upon the NSGA-II introduced in Section 10.2.2, is also considered herein through the use of homogeneous island-based models—Section 2.6.1. Recall that in a homogeneous island-based model, every island executes the same algorithm with the same parameterisation.

As a result, two different types of homogeneous island-based models are taken into account. The first one, in which every island executes the same parameterisation of the EAIPS, is referred to as *Single-Island*. In this case, the migration stage applies the ALL migration topology, whereas the ELI-M migration scheme is used. Additionally, a replacement only takes place when the migrated individual is fitter than every individual in the destination island. The second homogeneous island-based model, in which every island executes the same configuration of the diversity-based multi-objective MA, is called *Multi-Island*. The migration stage is the same as that applied by the *Single-Island* approach. The replacement scheme selected, however, is the ELI-R approach.

10.3 Parameter Control Schemes

This section focuses on describing the parameter control approach that is used herein to adapt some of the parameters contained in the optimisation schemes detailed in preceding sections. Particularly, the DYN model is applied to control some of the components and parameters corresponding to the diversity-based multi-objective MAs and the multi-objective MAs based on multi-objectivisation by aggregation presented in Section 10.2.2. Moreover, by applying the DYN model, the aforementioned memetic approaches are enabled for use in parallel environments.

10.3.1 Dynamic-mapped Island-based Model

The DYN model introduced in Section 6.3 is also used herein as a parameter control approach. Recall that in the DYN model a hyper-heuristic is combined together with a parallel island-based model in order to dynamically map the low-level configurations involved to the islands, rather than performing a static mapping as is the case with standard island-based models. In this case, the DYN model is based on the HH-PROB hyper-heuristic, which was detailed in Section 6.2. In order to solve the 2DPP, the set of low-level configurations was defined starting from the diversity based multi-objective MAs and the multi-objective MAs based on multi-objectivisation by aggregation, which were described in Section 10.2.2. Thus, the goal is to control certain components and parameters of these memetic approaches while enabling their use in parallel environments.

In addition, several migration stages were tested with the DYN model. In particular, four different migration stages were defined by combining two different replacement schemes with two separate migration topologies. The replacement schemes were HAM-R and ELI-R, whereas the migration topologies were ALL and RING—Section 2.6.1. The ELI-M migration scheme—Section 2.6.1—was used to define the four migration stages. In order to identify the different migration stages, the *Replacement–Topology* nomenclature is used. For example, the migration stage that uses the HAM-R replacement scheme and the ALL topology is called *HAM-R–ALL*.

10.4 Experimental Evaluation and Discussion

In this section, the set of experiments performed with the above optimisation schemes and parameter control approaches on several instances of the 2DPP is presented.

Experimental Method. The different optimisation schemes, as well as the parameter control approaches, were implemented using METCO. The tests were run on a Debian GNU/Linux computer with four AMD ® Opteron™ processors (model number 6164HE) at 1.7 GHz and 64 Gb RAM. The compiler was the GCC 4.7.2. Communications among different islands of the homogeneous island-based models, as well as the DYN model, were implemented asynchronously using the MPICH library. Since every experiment applied stochastic algorithms, every execution was repeated 32 times. As a result, comparisons were performed by applying the statistical analysis detailed in Section 1.2.6.

2DPP Instances. The analyses were conducted considering two different instances of the 2DPP. The first one is characterised by the following parameters: $X = 10$, $Y = 10$, $M = 99$, and 9,032 possible pair scores. The second one is the instance proposed in the competition session. Its parameters are the following: $X = 20$, $Y = 20$, $M = 399$, and 15,962 possible pair scores.

10.4.1 Comparison of the Sequential Memetic Algorithms

In this first experiment the goal is to discover whether the use of the proposed diversity-based multi-objective MA, as well as the use of the proposed multi-objective MA based on multi-objectivisation by aggregation *Dependent*, offers any benefits over using the best-behaved single-objective MA available in the literature, i.e. the EAIPS. We should note that the multi-objective MAs are based on the NSGA-II. To carry out this experiment, the following configurations of the three optimisation schemes above were compared:

- 1 configuration of the EAIPS.
- 6 configurations of the diversity-based multi-objective MA, which were constituted by the use of six different diversity-based objective functions.
- 1 configuration of the *Dependent* approach.

The values for the parameters of the EAIPS were set as follows:

- Initial population size $N_0 = 2$.
- Maximum population size $N_{max} = 10$.
- *SoftBloq* = 50.

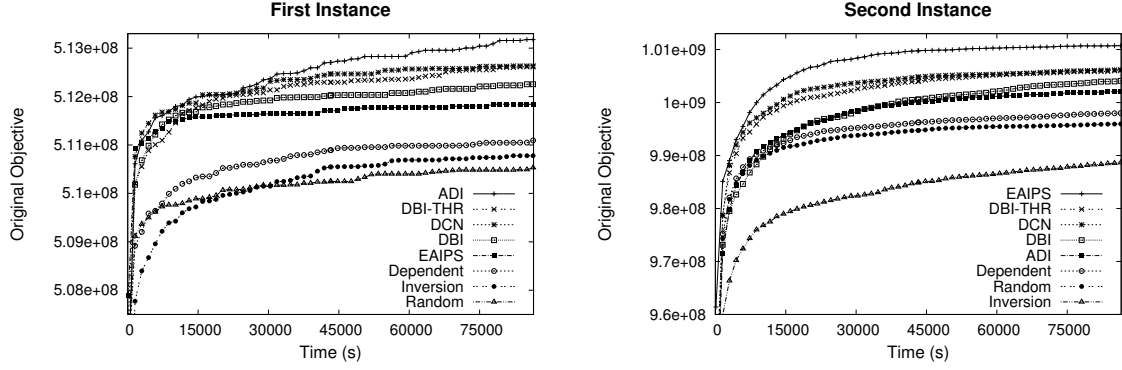


Figure 10.3: Evolution of the mean original objective value for the different memetic approaches

- $HardBloq = 300$.
- The UMD operator was applied with $min_p_m = 0.1$ and $max_p_m = 0.15$.
- Crossover rate $pc = 1$.
- Stopping criterion fixed to 24 hours.

In the case of the diversity-based multi-objective MA, the following parameterisation was considered:

- Population size $N = 10$.
- Diversity-based objectives *random*, *inversion*, ADI, DBI, DCN, and DBI-THR.
- Threshold value $th = 0.99$ for the diversity-based objective DBI-THR.
- The UMD operator was applied with $min_p_m = 0.1$ and $max_p_m = 0.15$.
- Crossover rate $pc = 1$.
- Stopping criterion fixed to 24 hours.

Finally, the *Dependent* scheme was applied with the same parameterisation as that used with the diversity-based multi-objective MA.

This analysis was carried out in terms of the performance achieved in the original objective function. Thus, Figure 10.3 shows, for both instances, the evolution of the mean original objective value achieved by the memetic schemes considered. In the case of the first instance, the ADI approach obtained the highest mean of the original objective value at the end of the executions. In addition, four configurations of the

Table 10.1: Statistical comparison among different memetic approaches for the first instance

	ADI	DBI-THR	DCN	DBI	EAIPS	Dependent	Inversion	Random
ADI	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow
DBI-THR	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
DCN	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
DBI	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
EAIPS	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
Dependent	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
Inversion	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
Random	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 10.2: Statistical comparison among different memetic approaches for the second instance

	EAIPS	DBI-THR	DCN	DBI	ADI	Dependent	Random	Inversion
EAIPS	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow
DBI-THR	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
DCN	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow	\uparrow
DBI	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
ADI	\downarrow	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow	\uparrow
Dependent	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\uparrow	\uparrow
Random	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow	\uparrow
Inversion	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\leftrightarrow

diversity-based multi-objective MA achieved a higher mean of the original objective value than the EAIPS. The worst results, however, were given by the *Dependent*, *Inversion*, and *Random* schemes. Considering the second instance, the EAIPS was able to attain the highest mean of the original objective value at the end of the executions, followed by the DBI-THR scheme, while the worst results were given once more by the *Dependent*, *Random*, and *Inversion* approaches.

In order to carry out a statistical comparison, Table 10.1 shows, for the first instance, the statistical differences among the different memetic approaches. Particularly, the table shows whether the scheme located in a given row is statistically better (\uparrow), not different (\leftrightarrow), or worse (\downarrow) than the corresponding scheme situated in a certain column. Table 10.2 shows the same information for the second instance. It is important to remark that in the case of the first instance, the ADI, DBI-THR, and DCN approaches, which obtained the highest mean of the original objective value at the end of the executions, were able to statistically outperform the EAIPS, thus demonstrating the advantages provided by the diversity-based multi-objective MA with respect to the best single-objective MA available in the literature.

For the second instance, the highest mean of the original objective value at the end of the executions was achieved by the EAIPS, which was also able to statistically outperform all the remaining memetic approaches. As a result, there exist test cases

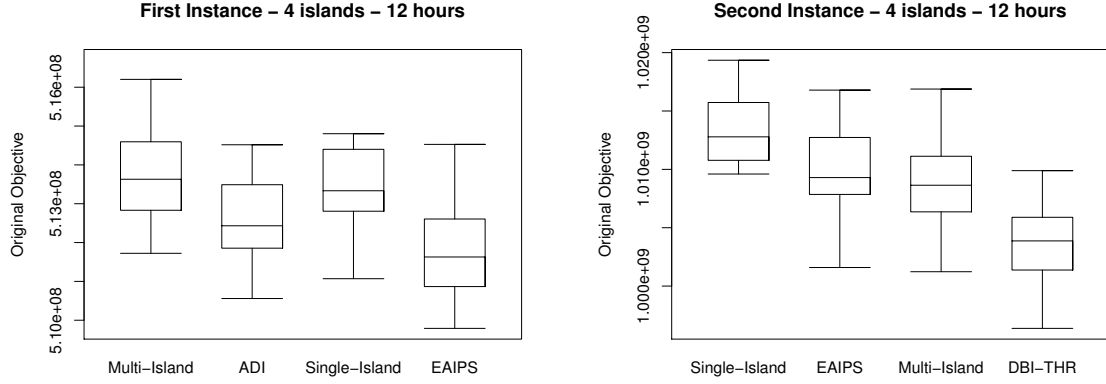


Figure 10.4: Box plots for the homogeneous island-based models executed with 4 islands for 12 hours

for which the diversity-based multi-objective MA is able to provide better results than those given by the single-objective MA, but there are other test cases where the latter yields better results than the former. Hence, the most suitable optimisation scheme depends on the 2DPP instance being solved.

Finally, if the diversity-based multi-objective MA and the multi-objective MA based on multi-objectivisation by aggregation *Dependent* are taken into consideration, then the results given by the former clearly outperform those obtained by the latter for both instances. Consequently, as was the case with the FAP in the previous chapter, for the case of the 2DPP, the use of a helper-objective that considers problem-dependent information provided no benefit over more general problem-independent diversity-based objectives.

10.4.2 Analysis of the Parallel Homogeneous Island-based Models

The main aim of the second experiment is twofold. First, to study the behaviour of different homogeneous island-based models when they are compared against the sequential memetic approaches that obtained the best performance in the previous section. Second, to analyse whether the application of a diversity-based multi-objective MA offers any benefits over the application of the EAIPS when both schemes are enabled for use in parallel environments. To do so, the two homogeneous

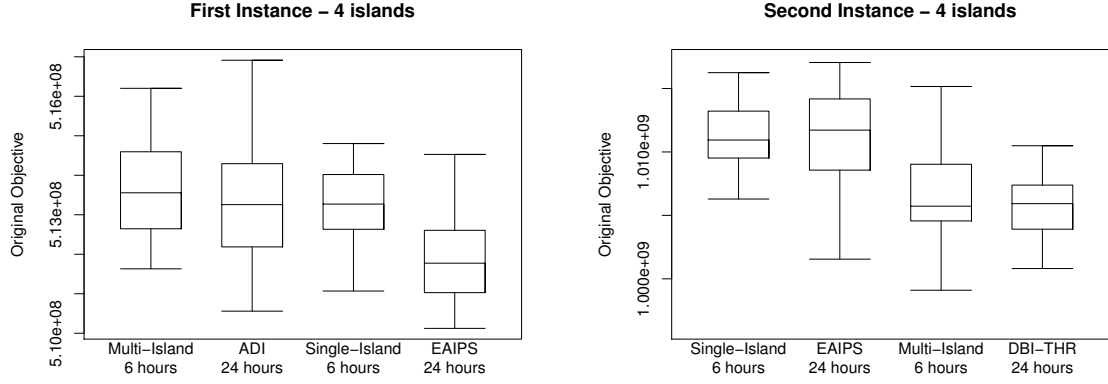


Figure 10.5: Box plots for the homogeneous island-based models executed with 4 islands and considering a fixed computational effort

island-based models proposed in Section 10.2.5 were executed with 4 islands and considering a stopping criterion of 12 hours. For both models, the migration rate was set to 1 individual, whereas the migration probability was set to 0.01. Every island in the *Single-Island* model executed the same configuration of the EAIPS. Every island in the *Multi-Island* model, however, applied the same configuration of the diversity-based multi-objective MA that provided the best results in the previous experiment, i.e. the ADI approach for the first instance and the DBI-THR scheme for the second instance. The parameterisation of the EAIPS and the diversity-based multi-objective MA was the same as that considered in the previous experiment.

Figure 10.4 shows, for both instances, the box plots of the original objective values achieved by the different sequential and parallel models in 12 hours. In the case of the first instance, note that both parallel models clearly outperformed their corresponding sequential variants. Furthermore, for this particular test case, the *Multi-Island* parallel model improved the results achieved by the remaining optimisation schemes considered. Hence, taking into account this first instance, the use of the diversity-based multi-objective MA, and in particular its parallelisation through the *Multi-Island* scheme, provided a clear advantage with respect to the application of the approaches based on the EAIPS. Finally, considering both sequential schemes, differences between them were more noticeable than in the case of both parallel models. For the second instance, both parallel models were also able to outperform their corresponding sequential versions. Nevertheless, the *Single-Island* parallel model provided better results than all the remaining optimisation schemes

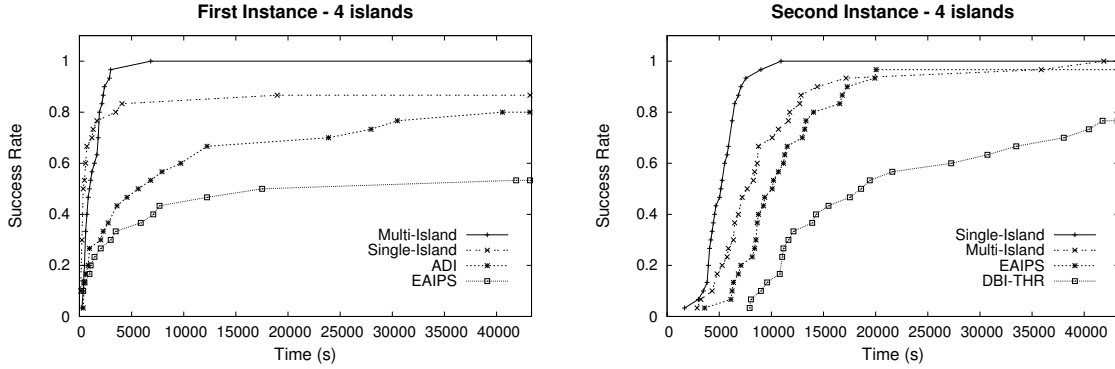


Figure 10.6: Run-length distributions for the homogeneous island-based models executed with 4 islands for 12 hours

for this test case. Consequently, as was stated in the first experiment, the use of the diversity-based multi-objective MA—and its parallelisation—did not yield any benefits over the use of the schemes based on the EAIPS for this test case.

The previous analysis showed that the parallel models yielded better solutions than those obtained by their corresponding sequential schemes when considering a stopping criterion equal to 12 hours for all the approaches. In order to check whether the parallel models make a more appropriate use of the computational resources than the sequential approaches, Figure 10.5 shows, for both instances, the box plots of the original objective values attained by the different sequential and parallel schemes considering the same amount of computational effort for all them. Since the parallel models were executed with 4 islands, the results obtained by the sequential schemes are shown assuming a stopping criterion equal to 24 hours, whereas for the parallel approaches the stopping criterion was fixed to 6 hours. Note that the parallel models and their corresponding sequential approaches made a similar use of the computational resources, since similar solutions were obtained by them. In the case of the first instance, the *Single-Island* scheme made even better use of the computational resources than the EAIPS, since the former was able to provide solutions of better quality than the latter.

We should note that the homogeneous island-based models use the computational resources in a parallel way. Although the parallel models considered a stopping criterion equal to 6 hours, they were able to provide similar or even better solutions than their corresponding sequential versions executed for 24 hours. This is because the parallel schemes were executed with 4 islands. Therefore, the validity of the homogeneous island-based models, in terms of the proper exploitation of the

computational resources available, is demonstrated.

So as to quantify the improvement of the parallel models with respect to the sequential schemes, the RLDs described in Section 1.2.6 were applied. In order to establish a sufficiently high quality level for each instance, it was set as the lowest mean original objective value achieved by any of the sequential and parallel models taken into account after 6 hours of execution. Figure 10.6 shows the RLDs calculated with the above quality level for both instances considered. Note the superiority of the parallel schemes in the case of the first instance. In fact, considering the time required to attain a 50% success rate, super-linear speedups were obtained by the parallel schemes using their corresponding sequential models as the reference approaches. The main reason for the super-linear speedups is probably the use of a larger population in the case of the parallel schemes, thus allowing stagnation in local optima to be avoided more efficiently. In addition, since the EAIPS essentially behaves as a trajectory-based algorithm, its parallelisation through the *Single-Island* model had an even greater impact on performance. This outcome was expected, since for the same computational effort, the *Single-Island* parallel model was able to provide better solutions than those given by the EAIPS, as was shown in Figure 10.5.

In the case of the second instance, the benefits of the parallel schemes can also be appreciated. Considering the time needed to achieve a 50% success rate, the speedup factor for the *Single-Island* parallel scheme was equal to 1.95, taking the EAIPS as the reference model. The speedup factor for the *Multi-Island* parallel approach was equal to 2.42, taking DBI-THR as the reference scheme. Finally, it is worth mentioning that the speedup factors changed when the success rates ranged from 25% to 75%. In the case of the schemes based on the EAIPS, the speedup factors ranged from 1.95 to 2.13. The speedup factors ranged from 1.89 to 3.58 for the approaches built upon the diversity-based multi-objective MA.

10.4.3 On the comparison of the Memetic Algorithms based on the Non-Dominated Sorting Genetic Algorithm II and the Strength Pareto Evolutionary Algorithm 2

The objective of this third experiment is to compare the behaviour of the novel MAs based on the NSGA-II and the SPEA2, which were proposed in Section 10.2.2, from the point of view of robustness. Particularly, the goal is to study whether the quality of the solutions provided depends on the MA applied and/or on the auxiliary objective—diversity-based objective or helper-objective—considered. Furthermore, additional analyses are also conducted with the aim of determining whether the

most suitable approach depends on the instance of the 2DPP considered. To this end, 16 different configurations of the MAs were defined:

- 14 configurations of the diversity-based multi-objective MAs, obtained by combining the NSGA-II and the SPEA2 with seven diversity-based objectives.
- 2 configurations of the multi-objective MAs based on multi-objectivisation by aggregation, where the first one is based on the NSGA-II and the second one is based on the SPEA2.

Every configuration of the diversity-based multi-objective MAs applied the following parameterisation:

- Population size $N = 10$ and archive size $\overline{N} = 10$.
- Diversity-based objectives *random*, *inversion*, ADI, DBI, DCN, DBI-THR, and DCN-THR.
- Threshold value $th = 0.99$ for the diversity-based objectives DBI-THR and DCN-THR.
- The UMD operator was applied with $min_p_m = 0.1$ and $max_p_m = 0.15$.
- Crossover rate $pc = 1$.
- In the case of the first instance, the stopping criterion was set to 5 hours, while for the second one a stopping criterion of 11.5 hours was used.

Both configurations of the multi-objective MAs based on multi-objectivisation by aggregation were applied with the above parameterisation, but using the *Dependent* helper-objective instead of considering a diversity-based objective. The executions carried out as part of this experiment were performed on the HECToR machine. Due to restrictions in the computational resources available, 24 executions, and not 32, were performed for each of the aforementioned configurations.

Table 10.3 shows, for the first instance and for each configuration of the MAs tested, the mean, the median, and the maximum of the original objective value attained at the end of the executions. The configurations of the MAs were sorted in terms of the mean original objective value achieved at the end of the executions. An index based on this order was assigned to every configuration. Thus, the first configuration, i.e. the one which achieved the highest mean of the original objective value, is referred to as SEQ1, while the last one is referred to as SEQ16. The differences among the configurations were noticeable and revealed the importance of correctly selecting the appropriate parameterisation. In fact, the differences between SEQ1 and the

Table 10.3: Original objective function for the memetic algorithms considering the first instance

Name	MA	Aux. Obj.	Mean	Median	Max
SEQ1	SPEA2	DBI	$5.130 \cdot 10^8$	$5.134 \cdot 10^8$	$5.152 \cdot 10^8$
SEQ2	SPEA2	DBI-THR	$5.129 \cdot 10^8$	$5.129 \cdot 10^8$	$5.157 \cdot 10^8$
SEQ3	NSGA2	ADI	$5.126 \cdot 10^8$	$5.125 \cdot 10^8$	$5.152 \cdot 10^8$
SEQ4	NSGA2	DCN	$5.124 \cdot 10^8$	$5.127 \cdot 10^8$	$5.142 \cdot 10^8$
SEQ5	SPEA2	DCN	$5.120 \cdot 10^8$	$5.121 \cdot 10^8$	$5.145 \cdot 10^8$
SEQ6	SPEA2	DCN-THR	$5.120 \cdot 10^8$	$5.118 \cdot 10^8$	$5.137 \cdot 10^8$
SEQ7	NSGA2	DBI-THR	$5.118 \cdot 10^8$	$5.119 \cdot 10^8$	$5.143 \cdot 10^8$
SEQ8	NSGA2	DCN-THR	$5.118 \cdot 10^8$	$5.115 \cdot 10^8$	$5.146 \cdot 10^8$
SEQ9	SPEA2	ADI	$5.117 \cdot 10^8$	$5.112 \cdot 10^8$	$5.144 \cdot 10^8$
SEQ10	NSGA2	DBI	$5.117 \cdot 10^8$	$5.119 \cdot 10^8$	$5.139 \cdot 10^8$
SEQ11	NSGA2	Dependent	$5.105 \cdot 10^8$	$5.102 \cdot 10^8$	$5.149 \cdot 10^8$
SEQ12	SPEA2	Dependent	$5.104 \cdot 10^8$	$5.105 \cdot 10^8$	$5.133 \cdot 10^8$
SEQ13	NSGA2	Random	$5.103 \cdot 10^8$	$5.103 \cdot 10^8$	$5.131 \cdot 10^8$
SEQ14	SPEA2	Random	$5.099 \cdot 10^8$	$5.097 \cdot 10^8$	$5.126 \cdot 10^8$
SEQ15	NSGA2	Inversion	$5.095 \cdot 10^8$	$5.093 \cdot 10^8$	$5.127 \cdot 10^8$
SEQ16	SPEA2	Inversion	$5.095 \cdot 10^8$	$5.097 \cdot 10^8$	$5.117 \cdot 10^8$

remaining configurations were statistically significant, except for the configurations SEQ2–SEQ4. Statistical tests also confirmed that both the auxiliary objective function and the MA used affected the quality of the solutions. For instance, SEQ1 was significantly different from SEQ10. These configurations are based on the same diversity-based objective—DBI—but they consider a different MA. Therefore, properly selecting the MA affects the quality of the solutions. Similarly, SEQ1 and SEQ5, which are both based on the SPEA2, were statistically different. Since they only differ in the diversity-based objective used, the importance of properly selecting this component is also demonstrated. Finally, the incorporation of a threshold value in the diversity-based objectives tested did not affect the quality of the results. The configurations that used the DBI-THR and DCN-THR diversity-based objectives were not statistically different from their non-threshold counterparts, i.e. DBI and DCN, for this particular instance.

Table 10.4 shows the same information for the second instance. In this case, differences among the configurations considered were also noticeable. The results obtained by SEQ1 were statistically different from those obtained by the other configurations, apart from the SEQ2 and SEQ3 configurations. In addition, changing the MA used did not yield significant differences in the results. For example, the differences between SEQ1 and SEQ2 were not statistically significant. In this case, both con-

Table 10.4: Original objective function for the memetic algorithms considering the second instance

Name	MA	Aux. Obj.	Mean	Median	Max
SEQ1	NSGA2	DCN-THR	$1.008 \cdot 10^9$	$1.007 \cdot 10^9$	$1.017 \cdot 10^9$
SEQ2	SPEA2	DCN-THR	$1.007 \cdot 10^9$	$1.006 \cdot 10^9$	$1.015 \cdot 10^9$
SEQ3	SPEA2	DBI-THR	$1.006 \cdot 10^9$	$1.007 \cdot 10^9$	$1.015 \cdot 10^9$
SEQ4	NSGA2	DBI-THR	$1.005 \cdot 10^9$	$1.005 \cdot 10^9$	$1.015 \cdot 10^9$
SEQ5	SPEA2	DCN	$1.004 \cdot 10^9$	$1.005 \cdot 10^9$	$1.013 \cdot 10^9$
SEQ6	NSGA2	DCN	$1.004 \cdot 10^9$	$1.005 \cdot 10^9$	$1.015 \cdot 10^9$
SEQ7	NSGA2	ADI	$1.004 \cdot 10^9$	$1.002 \cdot 10^9$	$1.015 \cdot 10^9$
SEQ8	SPEA2	ADI	$1.002 \cdot 10^9$	$1.002 \cdot 10^9$	$1.013 \cdot 10^9$
SEQ9	NSGA2	DBI	$1.001 \cdot 10^9$	$1.001 \cdot 10^9$	$1.011 \cdot 10^9$
SEQ10	SPEA2	DBI	$9.997 \cdot 10^8$	$9.992 \cdot 10^8$	$1.009 \cdot 10^9$
SEQ11	SPEA2	Random	$9.961 \cdot 10^8$	$9.995 \cdot 10^8$	$1.004 \cdot 10^9$
SEQ12	NSGA2	Random	$9.950 \cdot 10^8$	$9.945 \cdot 10^8$	$1.004 \cdot 10^9$
SEQ13	NSGA2	Dependent	$9.946 \cdot 10^8$	$9.952 \cdot 10^8$	$1.001 \cdot 10^9$
SEQ14	SPEA2	Dependent	$9.946 \cdot 10^8$	$9.956 \cdot 10^8$	$1.001 \cdot 10^9$
SEQ15	SPEA2	Inversion	$9.864 \cdot 10^8$	$9.861 \cdot 10^8$	$9.938 \cdot 10^8$
SEQ16	NSGA2	Inversion	$9.851 \cdot 10^8$	$9.850 \cdot 10^8$	$9.923 \cdot 10^8$

figurations applied the same diversity-based objective but used different MAs. This happened for every pair of configurations in which the auxiliary objective function applied was the same and the MA applied was different. However, the auxiliary objective function applied did affect the quality of the solutions. For instance, SEQ1 was statistically different from SEQ4, with the two configurations applying the DCN-THR and the DBI-THR diversity-based objectives, respectively. Finally, we should note that configurations applying a diversity-based objective with threshold were statistically different from their non-threshold counterparts. Consequently, the incorporation of a threshold value in the diversity-based objectives tested improved the quality of the results.

For both instances tested, the most suitable configurations of the MAs were different. For example, the configuration SEQ1 for the first instance was the configuration SEQ10 for the second instance. Similarly, the configuration SEQ1 for the second instance was the configuration SEQ8 for the first one. Hence, the clear conclusion is that the most suitable configurations depend on the features of the instance in question, resulting in some robustness problems. Given a new instance, it is difficult to predict which configuration will provide the best results. In addition, if the number of configurations considered is very large, testing each one of them might not be feasible. Therefore, the application of parameter control approaches seems

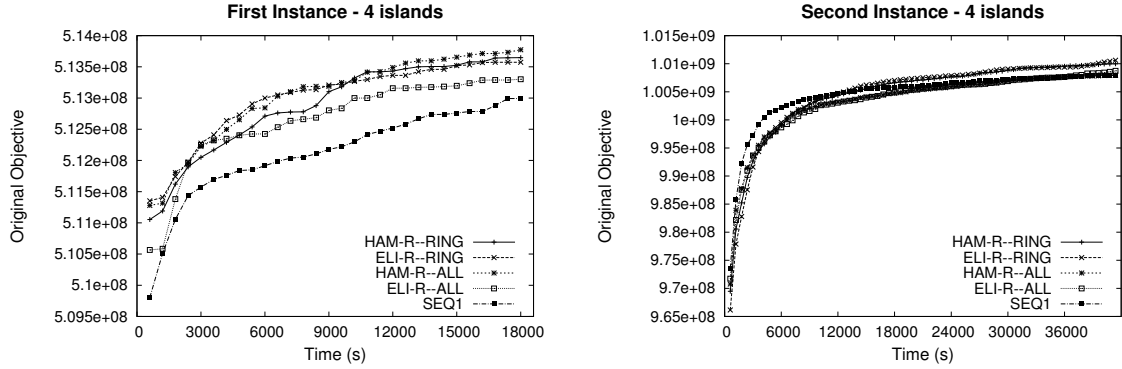


Figure 10.7: Evolution of the mean original objective value for the dynamic-mapped island-based model executed with 4 islands

very promising. Lastly, since the sequential models do not converge even after a very long period of time, the usage of parallel models also seems a promising approach.

10.4.4 Analysing the Robustness of the Dynamic-mapped Island-based Model

The main drawback of the MAs applied in the preceding experiment is that their performance depend on several components and parameters, which have to be tested in order to obtain promising results. Since the optimal parameterisation depends on the instance being solved, the parameter setting requires a large computational and user effort. Thus, the aim of the current experiment is twofold. First, to adaptively adjust some of the components and parameters of the multi-objective MAs applied in the previous experiment, as well as to mitigate their robustness issues, by applying the DYN model. Second, to analyse the robustness of the DYN model in terms of the migration stage used.

To do so, the DYN model was executed with the four migration stages described in Section 10.3.1. The migration rate for every migration stage was set to 1 individual, whereas the migration probability was set to 0.01. A total number of $n_p = 4$ islands was considered. The global stopping criterion was set to 5 hours for the first instance and 11.5 hours for the second one. For both instances, the local stopping criterion was set to 10 minutes. The HH-PROB hyper-heuristic of the DYN model was applied with an adaptation level $k = \infty$, and the value of β was set such that 10% of the decisions made by the hyper-heuristic used a uniform distribution, i.e. $\beta \cdot n_h = 0.1$.

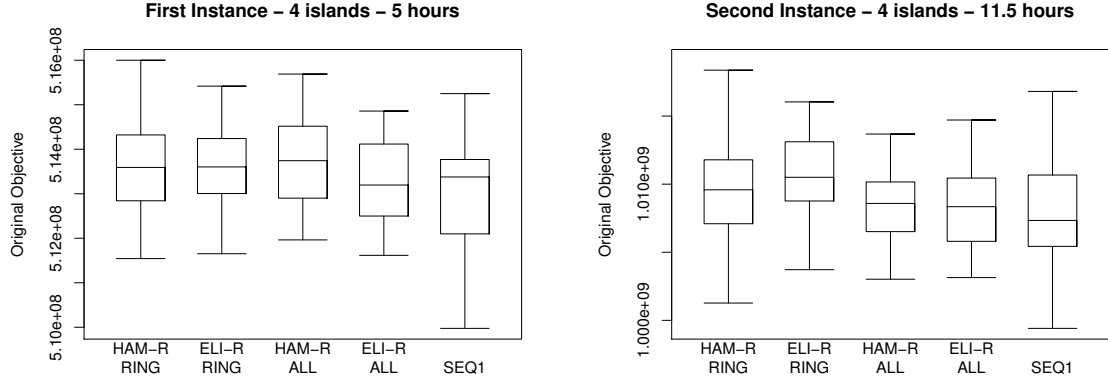


Figure 10.8: Box plots for the dynamic-mapped island-based model executed with 4 islands

The $n_h = 16$ configurations of the MAs applied in the previous experiment were used as the low-level approaches. Finally, we should note that this experiment, consisting of 24 repetitions, was also carried out on the HECToR machine.

Figure 10.7 shows, for the first and second instances, the evolution of the mean original objective value for the DYN model combined together with the four migration stages. In order to compare the results obtained by the parallel approaches, also shown are the data of the best-behaved sequential configuration—SEQ1 in the previous experiment—for each of the two instances considered. For both test cases, the parallel models were able to achieve a higher mean of the original objective value than the corresponding best-behaved sequential approach. Moreover, the differences between the parallel model which yielded the highest mean of the original objective value and the best sequential approach for each instance considered were statistically significant. In the case of the first instance, the best parallel approach relied on the HAM-R–ALL migration stage, whereas for the second instance, the best-behaved parallel model applied the ELI-R–RING migration stage. Consequently, depending on the features of the instance in question, the most suitable migration stage must be properly selected. The box plots of the DYN model executed with the separate migration stages, which are shown in Figure 10.8, confirm the above conclusions.

The DYN model avoided the need to check for the most suitable configuration of an algorithm for a given instance. The solutions obtained from the parallel model applying the best migration stage were of higher quality than that obtained by the best sequential approach. Thus, high quality solutions can be achieved by a single

Table 10.5: Statistical tests for the dynamic-mapped island-based model – 16 islands – 5 hours – First instance

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
HAM-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow
HAM-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 10.6: Statistical tests for the dynamic-mapped island-based model – 32 islands – 5 hours – First instance

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
HAM-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
HAM-R-ALL	\downarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

execution of this parallel model, resulting in lower use of computational resources. This process thus mitigates the robustness problems of the MAs applied in the preceding section. Finally, the DYN model facilitates the application of said MAs in terms of their parameter setting, while enabling their use in parallel environments.

10.4.5 Analysing the Scalability of the Dynamic-mapped Island-based Model

The goal of this fifth experiment is to analyse the scalability of the DYN model in terms of the effect on performance caused by the different migration stages as the number of islands grows. To do this, the DYN model was executed with the same parameterisation and four migration stages as those used in the preceding experiment, but considering 8, 16, and 32 islands. The experiments, consisting of 24 repetitions, were performed on the HECToR machine.

In the first instance, the statistical differences among the different migration stages were not significant when the DYN model was applied with 4 and 8 islands. With 16 and 32 islands, however, statistical differences among the migration stages did appear. This means that the importance of properly selecting the migration stage increases with the number of islands. Tables 10.5 and 10.6 show the statistical significances for the different migration stages considering 16 and 32 islands, respectively. Every cell shows whether the row model is statistically better (\uparrow), not different (\leftrightarrow),

Table 10.7: Statistical tests for the dynamic-mapped island-based model – 4 islands – 11.5 hours – Second instance

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
HAM-R-RING	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\leftrightarrow
HAM-R-ALL	\leftrightarrow	\downarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

Table 10.8: Statistical tests for the dynamic-mapped island-based model – 8, 16, 32 islands – 11.5 hours – Second instance

	HAM-R-RING	ELI-R-RING	HAM-R-ALL	ELI-R-ALL
HAM-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow
ELI-R-RING	\leftrightarrow	\leftrightarrow	\uparrow	\uparrow
HAM-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow
ELI-R-ALL	\downarrow	\downarrow	\leftrightarrow	\leftrightarrow

or worse(\downarrow) than the corresponding column model.

In the case of the second instance, Table 10.7 shows the results of the statistical tests for the different migration stages considering 4 islands. Similarly, Table 10.8 shows the same information when 8, 16, and 32 islands were used. The number of significant statistical differences was larger when 8, 16, and 32 islands were applied. As was the case with the first instance, the importance of selecting the appropriate migration stage increases as a higher number of islands is used.

In order to better quantify the importance of selecting the appropriate migration stage for the DYN model, another analysis was conducted. Considering the mean of the original objective value achieved by the parallel models with 32 islands at the end of the executions, the best and worst models were selected for each instance. Figure 10.9 shows, for the first instance, the box plots of the original objective values achieved by the best and worst parallel models when they were run with up to 32 islands. The same information is shown in Figure 10.10 for the second instance. For both cases, the trend towards obtaining better objective values as the number of islands increases is clear when the best migration stage is considered. However, this did not happen when considering the worst migration stage.

The above analysis compared different parallel models in terms of the quality attained at set intervals. However, it is important to quantify the improvement achieved by these parallel approaches in terms of the amount of time saved. To do so, an additional study that relied on RLDs was conducted. Figure 10.11 shows,

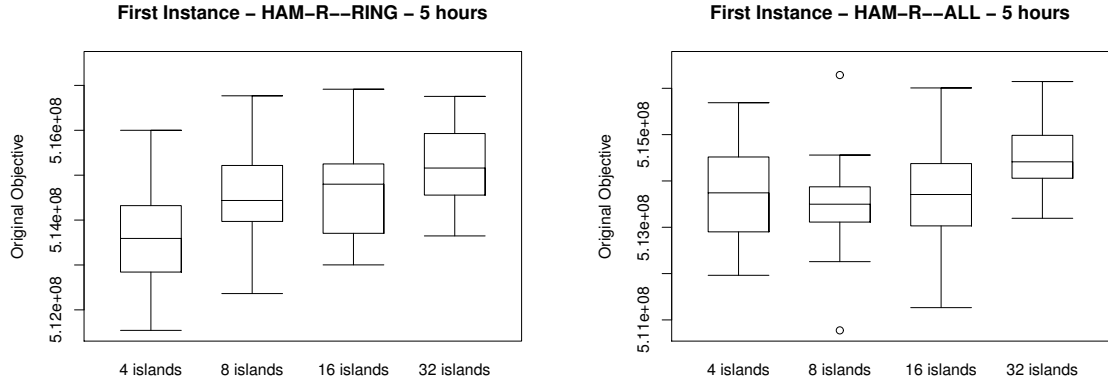


Figure 10.9: Box plots for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the first instance

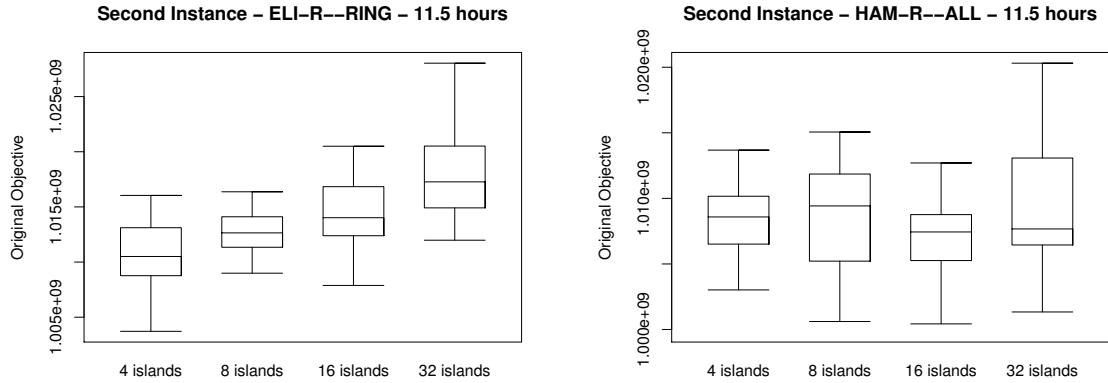


Figure 10.10: Box plots for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the second instance

for the first instance, the RLDs of the best and worst parallel models with up to 32 islands. It also includes the RLDs of the sequential configurations SEQ1 and SEQ3 so as to compare the results obtained by the parallel models. The same information is shown in Figure 10.12 for the second instance. In order to calculate the RLDs for both instances, the quality level was set as the median of the original objective value achieved by the SEQ3 configuration at the end of the executions. In the case of the first instance, the parallel models that used the best migration stage clearly

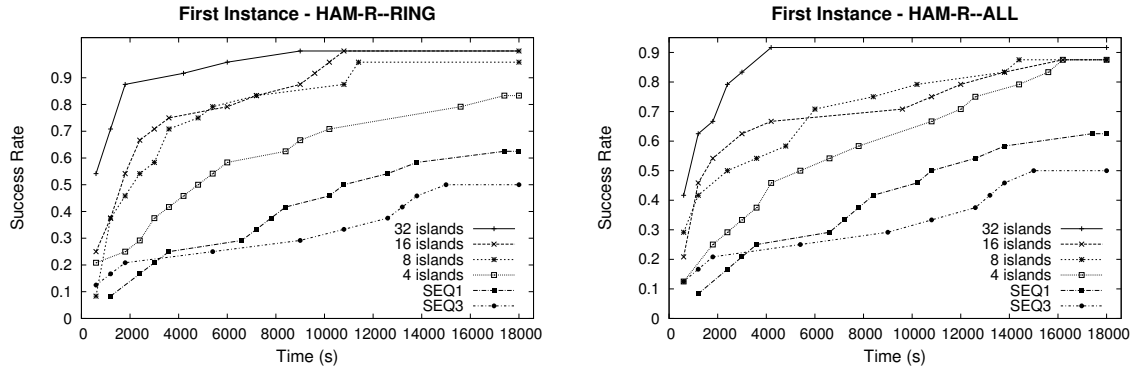


Figure 10.11: Run-length distributions for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the first instance

Table 10.9: Speedup factors for the dynamic-mapped island-based model with the best and worst migration stages – First instance

	Best	Worst
4 islands	1.71	1.61
8 islands	3.98	1.07
16 islands	6.29	1.21
32 islands	15.73	7.26

outperformed the sequential configurations, obtaining the same or higher success rates in less time. In addition, not only were high quality solutions yielded by the best parallel model, but they were obtained in less time when the number of islands increased. For example, the best parallel models with 16 and 32 islands were able to achieve a 100% success rate, i.e. every execution reached the set quality level, although the best parallel model with 32 islands obtained this success rate in less time. The solutions yielded by the parallel models that used the worst-behaved migration stage were of a lower quality than those obtained by the best parallel models. Moreover, none of the worst parallel models was able to reach a 100% success rate.

In the case of the second instance, the same conclusions as for the first instance can be extracted for the parallel model using the best migration stage. The behaviour of the parallel models when using the worst migration stage, however, was poor. For example, the parallel model with 4 islands was able to attain certain success rates in less time than the same model using 32 islands. In summary, for both instances, selecting the appropriate migration stage does not only affect the quality

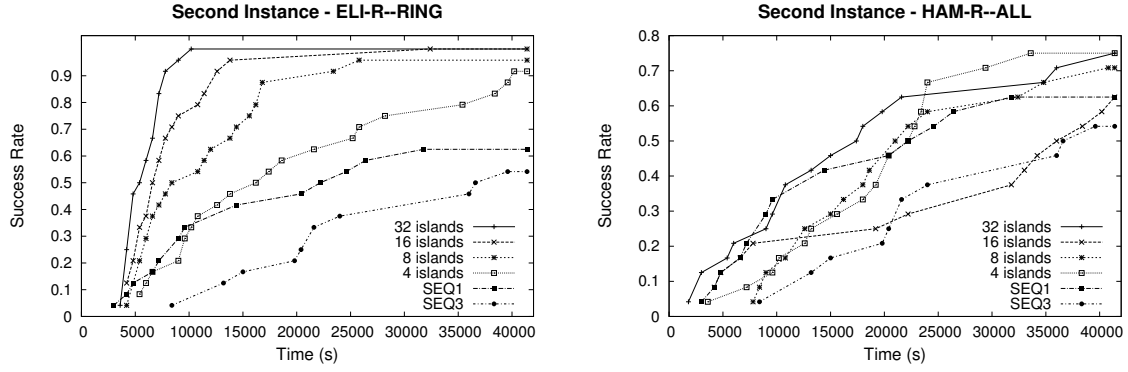


Figure 10.12: Run-length distributions for the dynamic-mapped island-based model with the best (left-hand side) and worst (right-hand side) migration stages for the second instance

Table 10.10: Speedup factors for the dynamic-mapped island-based model with the best and worst migration stages – Second instance

	Best	Worst
4 islands	2.30	1.64
8 islands	4.10	2.57
16 islands	5.70	1.36
32 islands	18.81	2.90

of the solutions provided, but also the total amount of time and processors required to achieve said quality.

In order to quantify the effects that the migration stage has on the scalability of the DYN model, speedup factors were calculated using the data provided by the RLDs. Table 10.9 shows the resulting speedup factors obtained by the DYN model when applied to the first instance with the best and worst migration stages, taking SEQ1 as the reference scheme. In order to calculate these factors the following steps were performed. Firstly, given a model with n_p islands, a relative speedup factor $spr_{[n_p]}$ was calculated with respect to the model with $n_p \div 2$ islands. In the case of the parallel models with $n_p = 4$ islands, the best sequential configuration—SEQ1—was used as the reference approach. For this instance, and for each relative speedup factor, the quality level was set as the lowest median of the original objective value achieved in 5 hours by the two models considered. The relative speedup was calculated by dividing the time invested by the model using a lower number of processors by the time invested by the model using a higher amount of processors.

These times were obtained by considering a 50% success rate. Once the relative speedup factors were calculated, the resulting speedup factor $sp_{[n_p]}$ for the model with n_p processors was calculated as follows:

$$sp_{[n_p]} = \begin{cases} spr_{[n_p]} \cdot sp_{[n_p \div 2]} & \text{if } n_p \neq 4 \\ spr_{[4]} & \text{if } n_p = 4 \end{cases} \quad (10.7)$$

The resulting speedup factors for the second instance are shown in Table 10.10. The aforementioned procedure was also used to calculate these factors, but instead of using 5 hours, 11.5 hours were considered.

For both instances, the speedup factors increased when the best parallel model was applied with a larger amount of islands. For example, in the case of the first instance, the best parallel model with 16 islands obtained a speedup factor equal to 6.29, while the same model considering 32 islands achieved a speedup factor equal to 15.73. In this case, the relative speedup factor calculated for both models was greater than one. This means that 50% of the executions performed by the model with 32 islands achieved the set quality level in less time than the model with 16 islands. However, this was not the case when the corresponding worst parallel model was applied to each instance. For example, in the case of the second instance, the worst parallel model with 8 islands obtained a speedup factor equal to 2.57, while the worst parallel model with 16 islands yielded a speedup factor equal to 1.36. In this case, the relative speedup factor calculated for both models was lower than one. This means that 50% of the executions carried out with the model with a lower number of islands attained the specified quality level in less time than the model which considered a higher number of islands. Therefore, incorporating a larger number of processors to the worst-behaved parallel model for each instance in question did not provide good results.

In order to study the scalability of the DYN model with a larger number of processors, it was executed using the best-behaved migration stage for each instance considering 64 and 128 islands. However, the global stopping criterion was set to 2 hours for both instances due to restrictions on the amount of computational resources available.

Figure 10.13 shows the box plots for the DYN model with up to 128 islands, applying the corresponding best-behaved migration stage for each instance. Even with a large number of islands, the quality of the solutions obtained by the DYN model kept increasing as more resources were considered. In general, the larger the number of islands, the higher the quality of the solutions obtained. Table 10.11 shows

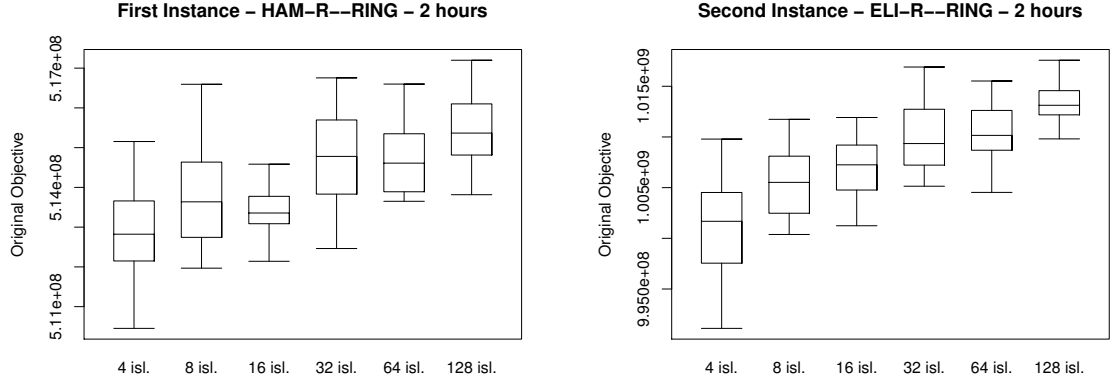


Figure 10.13: Box plots for the dynamic-mapped island-based model with the best migration stage for the first and second instances

Table 10.11: Speedup factors for the dynamic-mapped island-based model with the best migration stage for both instances

	64 islands	128 islands
First instance	12.58	41.90
Second instance	20.51	25.02

the speedup factors for the DYN model using the best migration stage for each instance, applying 64 and 128 islands, and considering SEQ1 as the reference approach. Speedup factors were obtained following the same procedure used above. In order to obtain the relative speedup factors for both instances, the quality level was set at the lowest median of the original objective value obtained by the two models in question in 2 hours. Note that the calculated speedup factors confirm the benefits of adding a larger amount of processors. The benefits were more noticeable in the case of the first instance. Finally, we should note that the use of the DYN model avoids the requirement of independently executing each low-level configuration considered. Hence, the total amount of time that can be saved is greater than that shown by the speedup factors calculated.

The executions carried out with $n_p = 128$ islands were able to provide better results than the best previously known solution for the first instance, which was obtained in the second experiment described in Section 10.4.2. The previous best solution was obtained by the *Multi-Island* homogeneous island-based model. Recall that it was applied considering 4 islands and 12 hours of execution. In order to apply this

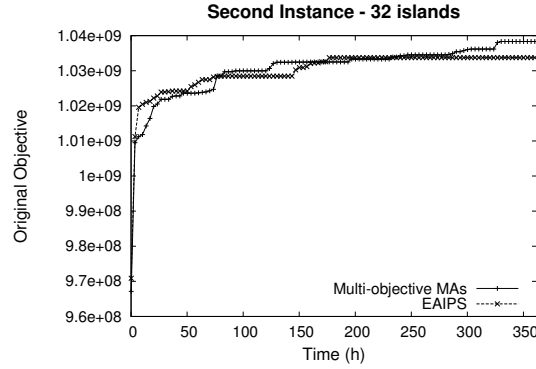


Figure 10.14: Long-term evolution of the mean original objective value for the dynamic-mapped island-based models

homogeneous parallel model, the best low-level configuration for a given instance had to be identified, requiring many computational experiments beforehand. A larger number of islands was used in the current experiment to improve on the best previous solution. Note that, however, the improvement was achieved after only two hours of execution. Specifically, the original objective value rose from 516,202,152 to 517,199,441. Furthermore, the use of a larger number of islands was offset by the reduced time and computational resources needed by the DYN model, since the preliminary analysis to identify the best low-level configuration required by the *Multi-Island* parallel approach was not necessary.

Since the second instance is harder than the first one, larger executions were needed to improve on the best-known solution for this test case. The previous best solution was obtained by the DYN model executed with separate low-level configurations of the single-objective EAIPS [205]. It was applied considering 4 islands and a 6-hour run time. For this particular test case, the use of multi-objective MAs did not provide benefits in the short term, as was stated in Section 10.4.2. However, it would be interesting to determine whether said multi-objective MAs can avoid long-term stagnation. To do so, the DYN model was executed with 32 islands and considering the same parameterisation as that used throughout the current experiment. The ELI-R-RING migration stage was used, while the global stopping criterion was set to 15 days. Due to the availability of resources, a larger number of processors was not considered and only one execution was performed. The DYN model based on the EAIPS was also run for 15 days. Figure 10.14 shows the evolution of the original objective value for both schemes. Note that, in the short term, the DYN model based on the EAIPS was superior. In the medium term, both approaches

behaved similarly. Finally, in the long term, the DYN model based on the multi-objective MAs provided the best solution. Since only one execution was performed, the superiority of the latter can not be ensured. However, the original objective value increased from 1,032,619,547 to 1,038,329,890.

Part IV

Conclusions and Future Lines of Research

Conclusions and Future Lines of Research

Meta-heuristics, and particularly EAs, have shown their ability to provide high-quality solutions to a wide range of complex applications. For some optimisation problems, though, EAs could exhibit the phenomenon called genetic drift—a loss of diversity in a finite population of individuals—which is the main reason for the appearance of premature convergence. In this dissertation, various proposals have been introduced in order to mitigate this issue. First, several novel diversity-based MOEAs, including diversity-based multi-objective MAs, which are based on well-known algorithms, such as the NSGA-II and the SPEA2, have been proposed to deal with the problem of premature convergence when solving single-objective optimisation problems. Recall that in diversity-based MOEAs, a metric of the diversity introduced by each individual is used as an additional objective function, which has to be simultaneously optimised together with the original objective function of the single-objective problem in question. Different encoding-independent and genotypic diversity measures have been taken into account in order to design the diversity-based objective functions considered. In addition, novel genotypic diversity-based objectives that incorporate parameters have been proposed. While they are able to provide better results than those yielded by diversity-based objectives without parameters, their main drawback lies in the fact that said parameters must be properly set depending on the problem and/or instance at hand. Second, multi-objectivisation by aggregation of helper-objectives has also been considered as a technique to prevent premature convergence. Since the helper-objectives are components of the original objective function, they make use of information that depends on the optimisation problem being solved, unlike diversity-based objectives, which are more general problem-independent approaches. Finally, a novel diversity-based survivor selection scheme has also been proposed herein. In this scheme the diversity-based objective is calculated progressively while the individuals to survive for the next generation are selected.

Besides the drawback of premature convergence, finding the appropriate setting for an EA remains one of the persistent challenges for EC. Different approaches have been proposed in this thesis to address this problem. First, a set of novel parameter control schemes based on FLCs has been introduced herein. They incorporate different rule bases and a score function that allows the most promising set of rules to be enabled at every instant during the optimisation process. Additionally, they are able to control different discrete and continuous numeric parameters belonging to different meta-heuristics, including the parameters defined for the novel genotypic diversity-based objective functions. Second, several sequential and parallel hyper-heuristics have also been applied as parameter control methods. The parallel hyper-heuristic—the DYN model—is built upon an island-based model, and allows the candidate low-level configurations to be mapped to the available islands dynamically. Lastly, a novel hybrid control scheme that combines the use of FLCs and hyper-heuristics has also been introduced in this dissertation. It is able to simultaneously adapt symbolic and numeric parameters by combining the benefits of both types of methods while trying to avoid their drawbacks.

A vast experimental evaluation has been performed by not only considering benchmark problems, but also different complex, real-world applications. The results obtained from this extensive experimental evaluation have demonstrated the validity of the aforementioned proposals. In regard to the diversity-based objective functions considered, it was shown that the genotypic diversity-based objectives were able to outperform the schemes based on multi-objectivisation by aggregation, i.e. more general problem-independent approaches provided better performance than problem-dependent schemes. This is a clear advantage, since the more general techniques can be directly applied to different optimisation problems. Moreover, the genotypic diversity-based objectives also outperformed the encoding-independent schemes. With respect to the novel diversity-based objectives with parameters, they were able to attain better solutions than the diversity-based objectives without parameters. However, it was proved that the values for these parameters not only depend on the optimisation problem in question, but also on the current stage of the optimisation process. Therefore, said parameters must be properly set—and changed during the execution—in order to obtain promising results. Finally, in general, the diversity-based MOEAs provided better results than those obtained by the single-objective optimisers used as the comparison approaches, thus demonstrating the advantages of solving single-objective problems by the application of diversity-based schemes, which were able to mitigate the problem of premature convergence. Furthermore, the high-quality results yielded by the diversity-based MOEAs offered significant savings in the computational resources and time invested.

At this point, we should note that maintaining a proper diversity might reduce the convergence speed of the whole optimisation scheme, as was stated with the use of the novel survivor selection approach. In fact, it was shown that by applying this survivor selection operator, the convergence speed in the average case decreased; in exchange, highly sub-optimal results were unavailable in the worst executions. For some cases, a slower convergence was also detected when diversity-based MOEAs were compared against some single-objective optimisers, such as the ILS. For these cases, a homogeneous island-based model was considered in order to speed up the achievement of high-quality solutions by the diversity-based MOEAs, while enabling their use in parallel environments. The robustness analysis of this homogeneous island-based model showed that high-quality results can be obtained regardless of the migration stage considered. In addition, the scalability analysis revealed the benefits of adding a larger number of islands to this parallel approach. Lastly, by applying the homogeneous island-based model, the savings in computational resources and time were noticeable with respect to the corresponding sequential variants of the diversity-based MOEAs.

In regard to the different control approaches proposed to deal with the problem of parameter setting in EAs, we should note that both FLCs and hyper-heuristics can be applied to obtain promising results. They were successfully used to control different parameters in the novel diversity-based MOEAs, including the parameters of the diversity-based objectives presented herein, thus showing their general applicability. Additionally, both types of control schemes can be also used to control the parameters belonging to other meta-heuristics, and not only to the diversity-based MOEAs. For a considerable number of problems, both FLCs and hyper-heuristics did not present statistically significant differences, meaning they can be applied indistinctly. Other control schemes, such as self-adaptation, did not provide any benefits with respect to them. It is worth pointing out that the parameter values of the FLCs and the hyper-heuristics were the same regardless of the optimisation problem considered. This means that these control approaches are robust, since for a wide range of problems promising results were obtained without changing the parameter values. Hence, the parameters of said control methods do not add more burdens to the configuration of the diversity-based MOEAs, thus facilitating the application of the latter.

Regarding the hybrid control scheme based on FLCs and hyper-heuristics, it is important to remark that it was successfully applied to simultaneously adapt several symbolic and numeric parameters of the diversity-based MOEAs. In addition, we should note that this was the first time that FLCs and hyper-heuristics were combined into a hybrid control scheme.

With respect to the use of parallel control schemes, the DYN model was also able to adaptively adjust some of the parameters belonging to the diversity-based MOEAs with success, thus avoiding the need to check for the most suitable configuration for a certain problem and/or instance. The robustness analyses, in terms of the migration stage used, revealed that the DYN model is robust when a low number of islands is considered, since differences among the different migration stages defined were not statistically significant. For a higher number of processors, however, the practitioner must carefully select the components used to define the migration stage. In fact, the scalability studies showed that as the number of islands increases, the performance of the DYN model is more sensitive to the setting of the migration stage. The addition of extra islands yielded the best results in the best-behaved migration stage. In the worst-behaved stage, however, the DYN model did not profit from the addition of more islands.

The advantages of dynamically altering the parameter values of an EA during its run, instead of prefixing them before the run starts, i.e. the advantages of parameter control versus parameter tuning, were demonstrated for most of the problems. For several cases, the above control schemes yielded superior or at least similar results to those given by any of the fixed configurations considered, thus showing the clear superiority of parameter control. Additionally, in every case, a single run of the parameter control schemes was able to provide similar or even better results than those obtained by a considerable number of configurations independently executed with fixed parameter values. Since finding the most suitable fixed values for the parameters is a computationally demanding task, the savings in computational resources and time are evident when using the control approaches proposed throughout this dissertation.

Finally, it is worth pointing out that by applying the diversity-based approaches and parameter control schemes proposed in this thesis, the best previously known solutions for the the FAP and the 2DPP were improved upon.

These results suggest several lines of future work. Only the parameters belonging to diversity-based MOEAs were adapted by the use of the different parameter control approaches introduced herein. As a result, it would be interesting to apply the control schemes to other types of meta-heuristics and/or EAs. Another promising line of research could be the application of the control approaches to “*pure*” multi-objective optimisers. Lastly, enabling the hybrid control scheme based on FLCs and hyper-heuristics for use in parallel environments would be another interesting line of future work.

Part V

Appendices

List of Publications

This appendix introduces the set of publications that emerged from the different topics considered in this dissertation. The listing includes book chapters, articles published in international journals of relevance to the particular research field, and contributions to international conferences reviewed by committees to ensure the quality and validity of the works selected.

Book Chapters

- [1] C. Segura, E. Segredo, and C. León. Analysing the robustness of multiobjectivisation approaches applied to large scale optimisation problems. In E. Tantar, A.-A. Tantar, P. Bouvry, P. Del Moral, P. Legrand, C. A. Coello Coello, and O. Schütze, editors, *EVOLVE- A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, volume 447 of *Studies in Computational Intelligence*, pages 365–391. Springer Berlin Heidelberg, 2013.

International Journals

- [1] E. Segredo, C. Segura, and C. León. Fuzzy logic-controlled diversity-based multi-objective memetic algorithm applied to a frequency assignment problem. *Engineering Applications of Artificial Intelligence*, 30(0):199 – 212, 2014.

- [2] E. Segredo, C. Segura, and C. León. Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. *Journal of Global Optimization*, 58(4):769–794, 2014.
- [3] C. Segura, C. A. Coello Coello, E. Segredo, and C. León. On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, pages 1–10, 2014.
- [4] C. Segura, E. Segredo, and C. León. Scalability and robustness of parallel hyperheuristics applied to a multiobjectivised frequency assignment problem. *Soft Computing*, 17(6):1077–1093, 2013.

International Conferences

- [1] R. Batista, E. Segredo, C. Segura, C. León, and C. Rodríguez. Solving the unknown complexity formula problem with genetic programming. In I. Rojas, G. Joya, and J. Gabestany, editors, *Advances in Computational Intelligence*, volume 7902 of *Lecture Notes in Computer Science*, pages 232–240. Springer Berlin Heidelberg, 2013.
- [2] C. León, G. Miranda, E. Segredo, and C. Segura. Parallel hypervolume-guided hyperheuristic for adapting the multi-objective evolutionary island model. In N. Krasnogor, B. Melián-Batista, J. A. Moreno, J. M. Moreno-Vega, and D. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, volume 236 of *Studies in Computational Intelligence*, pages 261–272. Springer Berlin Heidelberg, 2009.
- [3] C. León, G. Miranda, E. Segredo, and C. Segura. Parallel library of multi-objective evolutionary algorithms. In *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 28–35, Feb 2009.
- [4] E. Segredo, C. Rodríguez, and C. León. Solving the parameter setting in multi-objective evolutionary algorithms using Grid::Cluster. In A. Leon F. de Carvalho, S. Rodríguez-González, J. Paz Santana, and J. Corchado, editors, *Distributed Computing and Artificial Intelligence*, volume 79 of *Advances in Intelligent and Soft Computing*, pages 489–496. Springer Berlin Heidelberg, 2010.
- [5] E. Segredo, C. Segura, and C. León. Analysing the adaptation level of parallel hyperheuristics applied to mono-objective optimisation problems. In D. Pelta,

- N. Krasnogor, D. Dumitrescu, C. Chira, and R. Lung, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*, volume 387 of *Studies in Computational Intelligence*, pages 169–182. Springer Berlin Heidelberg, 2011.
- [6] E. Segredo, C. Segura, and C. León. A multiobjectivised memetic algorithm for the frequency assignment problem. In *2011 IEEE Congress on Evolutionary Computation (CEC)*, pages 1132–1139, June 2011.
- [7] E. Segredo, C. Segura, and C. León. On the comparison of parallel island-based models for the multiobjectivised antenna positioning problem. In A. König, A. Dengel, K. Hinkelmann, K. Kise, R. Howlett, and L. Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6881 of *Lecture Notes in Computer Science*, pages 32–41. Springer Berlin Heidelberg, 2011.
- [8] E. Segredo, C. Segura, and C. León. Analysing the robustness of multiobjectivisation parameters with large scale optimisation problems. In *2012 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, June 2012.
- [9] E. Segredo, C. Segura, and C. León. Control of numeric and symbolic parameters with a hybrid scheme based on fuzzy logic and hyper-heuristics. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, June 2014. In Press.
- [10] C. Segura, A. Cervantes, A. Nebro, M. Jaraíz-Simón, E. Segredo, S. García, F. Luna, J. Gómez-Pulido, G. Miranda, C. Luque, E. Alba, M. Vega-Rodríguez, C. León, and I. Galván. Optimizing the DFCN broadcast protocol with a parallel cooperative strategy of multi-objective evolutionary algorithms. In M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, editors, *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 305–319. Springer Berlin Heidelberg, 2009.
- [11] C. Segura, C. A. Coello Coello, E. Segredo, and C. León. An analysis of the automatic adaptation of the crossover rate in differential evolution. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, June 2014. In Press.
- [12] C. Segura, C. A. Coello Coello, E. Segredo, G. Miranda, and C. León. Improving the diversity preservation of multi-objective approaches used for single-objective optimization. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 3198–3205, June 2013.
- [13] C. Segura, E. Segredo, Y. González, and C. León. Multiobjectivisation of the antenna positioning problem. In A. Abraham, J. Corchado, S. Rodríguez-González, and J. Paz Santana, editors, *International Symposium on Distributed*

Computing and Artificial Intelligence, volume 91 of *Advances in Intelligent and Soft Computing*, pages 319–327. Springer Berlin Heidelberg, 2011.

- [14] C. Segura, E. Segredo, and C. León. Analysing the robustness of multi-objectivisation approaches applied to large scale optimisation problems. In *EVOLVE – A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, pages 1–4, May 2011.
- [15] C. Segura, E. Segredo, and C. León. Parallel island-based multiobjectivised memetic algorithms for a 2D packing problem. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1611–1618, New York, NY, USA, 2011. ACM.
- [16] C. Segura, E. Segredo, and C. León. Analysing the adaptation level of parallel hyperheuristics applied to multiobjectivised benchmark problems. In *2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 138–145, Feb 2012.
- [17] C. Segura, E. Segredo, and C. León. Parallel hyperheuristics for a multiobjectivised 2D packing problem. In *Proceedings of the 9th ESICUP meeting*, pages 20–20, March 2012.

Fuzzy Rule Bases

B.1 Fuzzy Rule Bases for the FUZZY-A and FUZZY-A-TSK Approaches

Table B.1: Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation

Rules	Inputs			Output
ID	P-IN	IMP	VAR	P-OUT
1	L	L	L	PL
2	L	L	M	PL
3	L	L	H	NL
4	L	M	-	Z
5	L	H	-	Z
6	LMB	L	-	NM
7	LMB	M	-	NL
8	LMB	H	-	Z
9	LMA	L	-	NH
10	LMA	M	-	NL
11	LMA	H	-	Z
12	M	L	-	NU
13	M	M	-	NL
14	M	H	-	Z
15	MHA	L	-	NG
16	MHA	M	-	NL
17	MHA	H	-	Z
18	MHB	L	-	NG
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NG
22	H	M	-	NL
23	H	H	-	Z

Rules	Inputs			Output
ID	P-IN	IMP	VAR	P-OUT
1	L	L	-	PM
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	L	PL
5	LMB	L	M	PL
6	LMB	L	H	NL
7	LMB	M	-	Z
8	LMB	H	-	Z
9	LMA	L	-	NM
10	LMA	M	-	NL
11	LMA	H	-	Z
12	M	L	-	NH
13	M	M	-	NL
14	M	H	-	Z
15	MHA	L	-	NU
16	MHA	M	-	NL
17	MHA	H	-	Z
18	MHB	L	-	NG
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NG
22	H	M	-	NL
23	H	H	-	Z

Table B.2: Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PH	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	-	PM	
5	LMB	M	-	PL	
6	LMB	H	-	Z	
7	LMA	L	L	PL	
8	LMA	L	M	PL	
9	LMA	L	H	NL	
10	LMA	M	-	Z	
11	LMA	H	-	Z	
12	M	L	-	NM	
13	M	M	-	NL	
14	M	H	-	Z	
15	MHA	L	-	NH	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NU	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NG	
22	H	M	-	NL	
23	H	H	-	Z	

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PU	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	-	PH	
5	LMB	M	-	PL	
6	LMB	H	-	Z	
7	LMA	L	-	PM	
8	LMA	M	-	PL	
9	LMA	H	-	Z	
10	M	L	L	PL	
11	M	L	M	PL	
12	M	L	H	NL	
13	M	M	-	Z	
14	M	H	-	Z	
15	MHA	L	-	NM	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NH	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NU	
22	H	M	-	NL	
23	H	H	-	Z	

B.1. Fuzzy Rule Bases for the FUZZY-A and FUZZY-A-TSK Approaches

Table B.3: Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation

Rules	Inputs			Output
ID	P-IN	IMP	VAR	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PU
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PH
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PM
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	L	PL
14	MHA	L	M	PL
15	MHA	L	H	NL
16	MHA	M	-	Z
17	MHA	H	-	Z
18	MHB	L	-	NM
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NH
22	H	M	-	NL
23	H	H	-	Z

Rules	Inputs			Output
ID	P-IN	IMP	VAR	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PG
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PU
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PH
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	-	PM
14	MHA	M	-	PL
15	MHA	H	-	Z
16	MHB	L	L	PL
17	MHB	L	M	PL
18	MHB	L	H	NL
19	MHB	M	-	Z
20	MHB	H	-	Z
21	H	L	-	NM
22	H	M	-	NL
23	H	H	-	Z

Table B.4: Rule base number 6 to control the mutation rate p_m , and the parameter R of the Neighbour-based Mutation

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PG	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	-	PG	
5	LMB	M	-	PL	
6	LMB	H	-	Z	
7	LMA	L	-	PG	
8	LMA	M	-	PL	
9	LMA	H	-	Z	
10	M	L	-	PU	
11	M	M	-	PL	
12	M	H	-	Z	
13	MHA	L	-	PH	
14	MHA	M	-	PL	
15	MHA	H	-	Z	
16	MHB	L	-	PM	
17	MHB	M	-	PL	
18	MHB	H	-	Z	
19	H	L	L	PL	
20	H	L	M	PL	
21	H	L	H	NL	
22	H	M	-	Z	
23	H	H	-	Z	

B.1. Fuzzy Rule Bases for the FUZZY-A and FUZZY-A-TSK Approaches

Table B.5: Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	L	NL	
2	L	L	M	NL	
3	L	L	H	PL	
4	L	M	-	Z	
5	L	H	-	Z	
6	LMB	L	-	NM	
7	LMB	M	-	NL	
8	LMB	H	-	Z	
9	LMA	L	-	NH	
10	LMA	M	-	NL	
11	LMA	H	-	Z	
12	M	L	-	NU	
13	M	M	-	NL	
14	M	H	-	Z	
15	MHA	L	-	NG	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NG	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NG	
22	H	M	-	NL	
23	H	H	-	Z	

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PM	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	L	NL	
5	LMB	L	M	NL	
6	LMB	L	H	PL	
7	LMB	M	-	Z	
8	LMB	H	-	Z	
9	LMA	L	-	NM	
10	LMA	M	-	NL	
11	LMA	H	-	Z	
12	M	L	-	NH	
13	M	M	-	NL	
14	M	H	-	Z	
15	MHA	L	-	NU	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NG	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NG	
22	H	M	-	NL	
23	H	H	-	Z	

Table B.6: Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PH	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	-	PM	
5	LMB	M	-	PL	
6	LMB	H	-	Z	
7	LMA	L	L	NL	
8	LMA	L	M	NL	
9	LMA	L	H	PL	
10	LMA	M	-	Z	
11	LMA	H	-	Z	
12	M	L	-	NM	
13	M	M	-	NL	
14	M	H	-	Z	
15	MHA	L	-	NH	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NU	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NG	
22	H	M	-	NL	
23	H	H	-	Z	

Rules		Inputs			Output
ID	P-IN	IMP	VAR	P-OUT	
1	L	L	-	PU	
2	L	M	-	PL	
3	L	H	-	Z	
4	LMB	L	-	PH	
5	LMB	M	-	PL	
6	LMB	H	-	Z	
7	LMA	L	-	PM	
8	LMA	M	-	PL	
9	LMA	H	-	Z	
10	M	L	L	NL	
11	M	L	M	NL	
12	M	L	H	PL	
13	M	M	-	Z	
14	M	H	-	Z	
15	MHA	L	-	NM	
16	MHA	M	-	NL	
17	MHA	H	-	Z	
18	MHB	L	-	NH	
19	MHB	M	-	NL	
20	MHB	H	-	Z	
21	H	L	-	NU	
22	H	M	-	NL	
23	H	H	-	Z	

B.1. Fuzzy Rule Bases for the FUZZY-A and FUZZY-A-TSK Approaches

Table B.7: Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the threshold ratio th of the diversity-based objectives with parameters

Rules					Rules				
		Inputs					Inputs		
ID	P-IN	IMP	VAR	P-OUT	ID	P-IN	IMP	VAR	P-OUT
1	L	L	-	PG	1	L	L	-	PG
2	L	M	-	PL	2	L	M	-	PL
3	L	H	-	Z	3	L	H	-	Z
4	LMB	L	-	PU	4	LMB	L	-	PG
5	LMB	M	-	PL	5	LMB	M	-	PL
6	LMB	H	-	Z	6	LMB	H	-	Z
7	LMA	L	-	PH	7	LMA	L	-	PU
8	LMA	M	-	PL	8	LMA	M	-	PL
9	LMA	H	-	Z	9	LMA	H	-	Z
10	M	L	-	PM	10	M	L	-	PH
11	M	M	-	PL	11	M	M	-	PL
12	M	H	-	Z	12	M	H	-	Z
13	MHA	L	L	NL	13	MHA	L	-	PM
14	MHA	L	M	NL	14	MHA	M	-	PL
15	MHA	L	H	PL	15	MHA	H	-	Z
16	MHA	M	-	Z	16	MHB	L	L	NL
17	MHA	H	-	Z	17	MHB	L	M	NL
18	MHB	L	-	NM	18	MHB	L	H	PL
19	MHB	M	-	NL	19	MHB	M	-	Z
20	MHB	H	-	Z	20	MHB	H	-	Z
21	H	L	-	NH	21	H	L	-	NM
22	H	M	-	NL	22	H	M	-	NL
23	H	H	-	Z	23	H	H	-	Z

Table B.8: Rule base number 6 to control the threshold ratio th of the diversity-based objectives with parameters

Rules	Inputs			Output
ID	P-IN	IMP	VAR	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PG
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PG
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PU
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	-	PH
14	MHA	M	-	PL
15	MHA	H	-	Z
16	MHB	L	-	PM
17	MHB	M	-	PL
18	MHB	H	-	Z
19	H	L	L	NL
20	H	L	M	NL
21	H	L	H	PL
22	H	M	-	Z
23	H	H	-	Z

B.2 Fuzzy Rule Bases for the FUZZY-B and FUZZY-B-TSK Approaches

Table B.9: Rule base number 0 (left-hand side) and number 1 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters

Rules					Rules				
Inputs		Output			Inputs		Output		
ID	P-IN	IMP	BEST-P-IN	P-OUT	ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	L	NL	1	L	L	-	PM
2	L	L	M	PL	2	L	M	-	PL
3	L	L	H	PL	3	L	H	-	Z
4	L	M	-	Z	4	LMB	L	L	NL
5	L	H	-	Z	5	LMB	L	M	PL
6	LMB	L	-	NM	6	LMB	L	H	PL
7	LMB	M	-	NL	7	LMB	M	-	Z
8	LMB	H	-	Z	8	LMB	H	-	Z
9	LMA	L	-	NH	9	LMA	L	-	NM
10	LMA	M	-	NL	10	LMA	M	-	NL
11	LMA	H	-	Z	11	LMA	H	-	Z
12	M	L	-	NU	12	M	L	-	NH
13	M	M	-	NL	13	M	M	-	NL
14	M	H	-	Z	14	M	H	-	Z
15	MHA	L	-	NG	15	MHA	L	-	NU
16	MHA	M	-	NL	16	MHA	M	-	NL
17	MHA	H	-	Z	17	MHA	H	-	Z
18	MHB	L	-	NG	18	MHB	L	-	NG
19	MHB	M	-	NL	19	MHB	M	-	NL
20	MHB	H	-	Z	20	MHB	H	-	Z
21	H	L	-	NG	21	H	L	-	NG
22	H	M	-	NL	22	H	M	-	NL
23	H	H	-	Z	23	H	H	-	Z

Table B.10: Rule base number 2 (left-hand side) and number 3 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters

Rules		Inputs		Output
ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PH
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PM
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	L	NL
8	LMA	L	M	PL
9	LMA	L	H	PL
10	LMA	M	-	Z
11	LMA	H	-	Z
12	M	L	-	NM
13	M	M	-	NL
14	M	H	-	Z
15	MHA	L	-	NH
16	MHA	M	-	NL
17	MHA	H	-	Z
18	MHB	L	-	NU
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NG
22	H	M	-	NL
23	H	H	-	Z

Rules		Inputs		Output
ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PU
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PH
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PM
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	L	NL
11	M	L	M	Z
12	M	L	H	PL
13	M	M	-	Z
14	M	H	-	Z
15	MHA	L	-	NM
16	MHA	M	-	NL
17	MHA	H	-	Z
18	MHB	L	-	NH
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NU
22	H	M	-	NL
23	H	H	-	Z

Table B.11: Rule base number 4 (left-hand side) and number 5 (right-hand side) to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters

Rules		Inputs		Output
ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PU
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PH
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PM
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	L	NL
14	MHA	L	M	NL
15	MHA	L	H	PL
16	MHA	M	-	Z
17	MHA	H	-	Z
18	MHB	L	-	NM
19	MHB	M	-	NL
20	MHB	H	-	Z
21	H	L	-	NH
22	H	M	-	NL
23	H	H	-	Z

Rules		Inputs		Output
ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PG
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PU
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PH
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	-	PM
14	MHA	M	-	PL
15	MHA	H	-	Z
16	MHB	L	L	NL
17	MHB	L	M	NL
18	MHB	L	H	PL
19	MHB	M	-	Z
20	MHB	H	-	Z
21	H	L	-	NM
22	H	M	-	NL
23	H	H	-	Z

Table B.12: Rule base number 6 to control the mutation rate p_m , the parameter R of the Neighbour-based Mutation, and the threshold ratio th of the diversity-based objectives with parameters

Rules		Inputs		Output
ID	P-IN	IMP	BEST-P-IN	P-OUT
1	L	L	-	PG
2	L	M	-	PL
3	L	H	-	Z
4	LMB	L	-	PG
5	LMB	M	-	PL
6	LMB	H	-	Z
7	LMA	L	-	PG
8	LMA	M	-	PL
9	LMA	H	-	Z
10	M	L	-	PU
11	M	M	-	PL
12	M	H	-	Z
13	MHA	L	-	PH
14	MHA	M	-	PL
15	MHA	H	-	Z
16	MHB	L	-	PM
17	MHB	M	-	PL
18	MHB	H	-	Z
19	H	L	L	NL
20	H	L	M	NL
21	H	L	H	PL
22	H	M	-	Z
23	H	H	-	Z

Bibliography

- [1] K. I. Aardal, S. P. M. V. Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007.
- [2] H. A. Abbass and K. Deb. Searching under multi-evolutionary pressures. In *Proceedings of the Fourth Conference on Evolutionary Multi-Criterion Optimization*, pages 391–404. Springer-Verlag, 2003.
- [3] K. Abdi, M. Fathian, and E. Safari. A novel algorithm based on hybridization of artificial immune system and simulated annealing for clustering problem. *International Journal of Advanced Manufacturing Technology*, 60(5-8):723–732, 2012.
- [4] B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.*, 54(1):99–114, Jan. 2006.
- [5] E. Alba. Evolutionary algorithms for optimal placement of antennae in radio network design. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, page 168, April 2004.
- [6] A. Alsheddy and M. Kampouridis. Off-line parameter tuning for guided local search using genetic programming. In *2012 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–5, June 2012.
- [7] E. Amaldi, A. Capone, F. Malucelli, and C. Mannino. Optimization problems and models for planning cellular networks. In *Handbook of Optimization in Telecommunication*, pages 917–939. Springer, 2006.
- [8] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

BIBLIOGRAPHY

- [9] P. J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152 – 163. IEEE Press, 1995.
- [10] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS-a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence*, pages 73–78 vol.1, Jun 1994.
- [11] Argonne National Laboratory. *MPICH User's Guide. Version 3.0.4*, 2013. <http://www.mpich.org/static/downloads/3.0.4/mpich-3.0.4-userguide.pdf>.
- [12] A. Avenali, C. Mannino, and A. Sassano. Minimizing the span of d-walks to compute optimum frequency assignments. *Mathematical Programming*, 91(2):357–374, 2002.
- [13] T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992.
- [14] T. Bäck. Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press, 1992.
- [15] T. Bäck. Optimal mutation rates in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann, 1993.
- [16] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.
- [17] T. Bäck. Self-adaptation. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, chapter 21, pages 188 – 211. Institute of Physics Publishing, Bristol, 2000.
- [18] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An empirical study on gas "without parameters". In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 315–324, London, UK, UK, 2000. Springer-Verlag.
- [19] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing, Bristol, UK, 1st edition, 1997.

- [20] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, Mar. 1993.
- [21] M. Bader-El-Den and R. Poli. Generating SAT Local-Search Heuristics Using a GP Hyper-Heuristic Framework. In N. Monmarché, E.-G. Talbi, P. Collet, M. Schoenauer, and E. Lutton, editors, *Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, pages 37–49. Springer Berlin / Heidelberg, 2008.
- [22] R. Bai. *An Investigation of Novel Approaches for Optimising Retail Shelf Space Allocation*. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, United Kingdom, 2005.
- [23] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein, M. J. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg, 2007.
- [24] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.
- [25] R. Barr, B. Golden, J. Kelly, M. Resende, and J. Stewart, WilliamR. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.
- [26] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010.
- [27] T. Bartz-Beielstein, M. Parsopoulos, and M. Vrahatis. Analysis of particle swarm optimization using computational statistics. In Chalkis, editor, *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics, ICNAAM 2004*, pages 34–37. Wiley, 1994.
- [28] E. B. Baum. Towards practical ‘neural’ computation for combinatorial optimization problems. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 53–58, Woodbury, NY, USA, 1987. American Institute of Physics Inc.
- [29] J. Baxter. Local optima avoidance in depot location. *The Journal of the Operational Research Society*, 32(9):pp. 815–819, 1981.

BIBLIOGRAPHY

- [30] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [31] H. Bersini and F. Varela. Hints for adaptive problem solving gleaned from immune networks. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 343–354. Springer Berlin Heidelberg, 1991.
- [32] M. Biazizini, B. Banhelyi, A. Montresor, and M. Jelasity. Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1339–1346, New York, NY, USA, 2009. ACM.
- [33] M. Birattari. F-race for tuning metaheuristics. In *Tuning Metaheuristics*, volume 197 of *Studies in Computational Intelligence*, pages 85–115. Springer Berlin Heidelberg, 2009.
- [34] J. Blazewicz, E. K. Burke, G. Kendall, W. Mruczkiewicz, C. Oguz, and A. Swiercz. A hyper-heuristic approach to sequencing by hybridization of DNA sequences. *Annals of Operations Research*, 207(1):27–41, 2013.
- [35] J. Blazewicz and R. Walkowiak. A new parallel approach for multi-dimensional packing problem. In *4th International Conference on Parallel Processing and Applied Mathematics (PPAM)*, volume 2328 of *LNCS*, pages 194–201. Naleczow, Poland, Springer Berlin, September 2002.
- [36] S. Bleuler, J. Bader, and E. Zitzler. Reducing Bloat in GP with Multiple Objectives. In J. Knowles, D. Corne, K. Deb, and D. Chair, editors, *Multiobjective Problem Solving from Nature*, Natural Computing Series, pages 177–200. Springer Berlin Heidelberg, 2008.
- [37] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
- [38] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, 76:73–93, 1998.
- [39] G. Brassard and P. Bratley. *Fundamentals of Algorithms*. Prentice-Hall, New Jersey, 1996.
- [40] H. J. Bremermann. The evolution of intelligence. The nervous system as a model of its environment. Technical Report 1, Contract No. 477(17), Department of Mathematics, University of Washington, Seattle, July 1958.

- [41] H. J. Bremermann. Optimization through evolution and recombination. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems*, pages 93–106. Spartan Books, Washington, D.C., 1962.
- [42] F. H. Brito, A. N. Teixeira, O. N. Teixeira, and R. C. L. Oliveira. A fuzzy intelligent controller for genetic algorithms’ parameters. In L. Jiao, L. Wang, X.-b. Gao, J. Liu, and F. Wu, editors, *Advances in Natural Computation*, volume 4221 of *Lecture Notes in Computer Science*, pages 633–642. Springer Berlin Heidelberg, 2006.
- [43] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler. Do additional objectives make a problem harder? In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO ’07, pages 765–772, New York, NY, USA, 2007. ACM.
- [44] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler. On the Effects of Adding Objectives to Plateau Functions. *IEEE Trans. Evol. Comput.*, 13(3):591–603, 2009.
- [45] L. T. Bui, H. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2349–2356 Vol. 3, Sept 2005.
- [46] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *J Oper Res Soc*, 64(12):1695–1724, Dec 2013.
- [47] E. K. Burke, M. Hyde, and G. Kendall. Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation*, 16(3):406–417, June 2012.
- [48] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward. A classification of hyper-heuristic approaches. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 449–468. Springer US, 2010.
- [49] E. K. Burke and G. Kendall. *Search Methodologies*. Springer US, 2014.
- [50] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, vol-

BIBLIOGRAPHY

- ume 57 of *International Series in Operations Research & Management Science*, pages 457–474. Springer US, 2003.
- [51] E. K. Burke, G. Kendall, and E. Soubeiga. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*, 9(6):451–470, 2003.
 - [52] D. R. Butenhof. *Programming with POSIX Threads*. Addison-Wesley, 1997.
 - [53] M. Buzdalov and A. Buzdalova. Adaptive selection of helper-objectives for test case generation. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 2245–2250, 2013.
 - [54] A. Buzdalova, M. Buzdalov, and V. Parfenov. Generation of tests for programming challenge tasks using helper-objectives. In G. Ruhe and Y. Zhang, editors, *Search Based Software Engineering*, volume 8084 of *Lecture Notes in Computer Science*, pages 300–305. Springer Berlin Heidelberg, 2013.
 - [55] C. L. Li and Z.L. Chen. Binpacking Problem with Concave Costs of Bin Utilization. *Naval Research Logistics*, 53:298–308, 2006.
 - [56] P. Caamaño, A. Prieto, J. Becerra, F. Bellas, and R. Duro. Real-valued multimodal fitness landscape characterization for evolution. In K. Wong, B. Mendis, and A. Bouzerdoun, editors, *Neural Information Processing. Theory and Algorithms*, volume 6443 of *Lecture Notes in Computer Science*, pages 567–574. Springer Berlin / Heidelberg, 2010.
 - [57] J. Calderín, A. Masegosa, A. Suárez, and D. Pelta. Adaptation schemes and dynamic optimization problems: A basic study on the adaptive hill climbing memetic algorithm. *Studies in Computational Intelligence*, 512:85–97, 2014.
 - [58] W. B. Cannon. *The Wisdom of the Body*. The Norton Library, 1932.
 - [59] E. Cantú-Paz. A survey of parallel genetic algorithms. Technical report, University of Illinois at Urbana-Champaign, 1997.
 - [60] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7:311–334, 2001.
 - [61] M. Caserta and S. Voß. Metaheuristics: Intelligent problem solving. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics*, volume 10 of *Annals of Information Systems*, pages 1–38. Springer US, 2010.
 - [62] O. Castillo, A. Melendez, P. Melin, and L. Astudillo. Neuro-fuzzy fitness in a genetic algorithm for optimal fuzzy controller design. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5, 2013.

- [63] O. Castillo, H. Neyoy, J. Soria, M. García, and F. Valdez. Dynamic fuzzy logic parameter tuning for ACO and its application in the fuzzy logic control of an autonomous mobile robot. *International Journal of Advanced Robotic Systems*, 10, 2013.
- [64] K. Chakhlevitch and P. I. Cowling. Hyperheuristics: Recent developments. In C. Cotta, M. Sevaux, and K. Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 3–29. Springer, 2008.
- [65] B. Chapman, G. Jost, and R. van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press, 2007.
- [66] I. Charon. The noising methods: A generalization of some metaheuristics. *European Journal Of Operational Research*, 135(1):86–101, 2001.
- [67] A. Chatterjee. Differential evolution tuned fuzzy supervisor adapted extended kalman filtering for slam problems in mobile robots. *Robotica*, 27(3):411–423, May 2009.
- [68] C.-L. Chen and C.-P. Weng. A fuzzy multi-objective genetic algorithm approach to optimal parameter design for laser electrophotographic systems. In *Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications, 2009. ICIEA 2009*, pages 2786–2791, may 2009.
- [69] C. A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191:1245–1287, 2002.
- [70] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. Basic concepts. In *Evolutionary Algorithms for Solving Multi-Objective Problems*, Genetic and Evolutionary Computation Series, pages 1–60. Springer US, 2007.
- [71] J. L. Cohon and D. H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resources Research*, 11(2):208–220, 1975.
- [72] S. Cook. The P versus NP problem. In *Clay Mathematical Institute; The Millennium Prize Problem*, 2000.
- [73] O. Cordon, F. Herrera, and P. Villar. Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems*, 9(4):667–674, 2001.

BIBLIOGRAPHY

- [74] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2009.
- [75] D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price, editors. *New ideas in optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.
- [76] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin. Review and study of genotypic diversity measures for real-coded representations. *IEEE Transactions on Evolutionary Computation*, 16(5):695–710, 2012.
- [77] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin. Review of phenotypic diversity formulations for diagnostic tool. *Applied Soft Computing Journal*, 13(1):9–26, 2013.
- [78] P. Cowling, G. Kendall, and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1185–1190, Honolulu, Hawaii, 2002. IEEE Computer Society.
- [79] P. Cowling, G. Kendall, and E. Soubeiga. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of 4th Metahuristics International Conference (MIC 2001)*, pages 127–131, Porto Portugal, July 16-20 2001.
- [80] P. I. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, PATAT '00, pages 176–190, London, UK, UK, 2001. Springer-Verlag.
- [81] S. Coy, B. Golden, G. Runger, and E. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.
- [82] T. Crainic and M. Toulouse. Parallel Meta-Heuristics. Technical Report CIRRELT-2009-22, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), May 2009.
- [83] A. Czarn, C. MacNish, K. Vijayan, B. Turlach, and R. Gupta. Statistical exploratory analysis of genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4):405–421, 2004.
- [84] L. Davis. Adapting operator probabilities in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

- [85] R. Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, UK, 1990.
- [86] J. de Armas, G. Miranda, and C. León. Hyperheuristic encoding scheme for multi-objective guillotine cutting problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1683–1690, New York, NY, USA, 2011. ACM.
- [87] E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing Bloat and Promoting Diversity using Multi-Objective Methods. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. K. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'01, pages 11–18, San Francisco, California, USA, 2001. Morgan Kaufmann.
- [88] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.
- [89] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.
- [90] K. Deb and M. Goyal. A Combined Genetic Adaptive Search (GeneAS) for Engineering Design. *Computer Science and Informatics*, 26(4):30–45, 1996.
- [91] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.
- [92] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- [93] T. Dereli and G. S. Das. A hybrid simulated-annealing algorithm for two-dimensional strip packing problem. In *8th International Conference on Adaptive and Natural Computing Algorithms*, volume 4431 of *LNCIS*, pages 508–516. Warsaw, Poland, Springer Berlin, April 2007.
- [94] R. Ding, R. Zhao, and L. Fu. Code generation for accurate array redistribution on automatic distributed-memory parallelization. In *Proceedings of the 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 267–274, 2013.
- [95] S. Doncieux and J. B. Mouret. Behavioral diversity measures for Evolutionary Robotics. In *2010 IEEE Congress on Evolutionary Computation*, CEC'10, pages 1–8, 2010.

BIBLIOGRAPHY

- [96] S. Doncieux and J. B. Mouret. Behavioral diversity with multiple behavioral distances. In *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 1427–1434, 2013.
- [97] M. Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.
- [98] K. Dowsland, E. Soubeiga, and E. K. Burke. A Simulated Annealing Hyperheuristic for Determining Shipper Sizes. *European Journal of Operational Research*, 179(3):759–774, June 2007.
- [99] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization*. Springer Berlin Heidelberg, 2006.
- [100] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Springer Berlin Heidelberg, 1996.
- [101] F. Y. Edgeworth. *Mathematical Psychics*. History of Economic Thought Books. McMaster University Archive for the History of Economic Thought, 1881.
- [102] Edinburgh Parallel Computing Centre (EPCC). *HECToR: UK National Supercomputing Service*, 2014. <http://www.hector.ac.uk/>.
- [103] M. Ehrgott. Efficiency and nondominance. In *Multicriteria Optimization*, pages 23–64. Springer Berlin Heidelberg, 2005.
- [104] A. E. Eiben and T. Bäck. An empirical investigation of multi-parent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3):347–365, 1997.
- [105] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, jul 1999.
- [106] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith. Parameter control in evolutionary algorithms. In F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 19–46. Springer Berlin Heidelberg, 2007.
- [107] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

- [108] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, 2003.
- [109] A. Emmen. A survey of vector and parallel processors for numerical applications. *Advances in Water Resources*, 13(3):103–116, 1990.
- [110] M. Erickson, A. Mayer, and J. Horn. The Niche Pareto Genetic Algorithm 2 applied to the design of groundwater remediation systems. In E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 681–695. Springer Berlin Heidelberg, 2001.
- [111] L. Eshelman. The CHC adaptive search algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1990.
- [112] M. Fazzolari, R. Alcalá, Y. Nojima, H. Ishibuchi, and F. Herrera. A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy Systems*, 21(1):45–65, 2013.
- [113] D. Feng, X. Wang, M. Fei, and T. Chen. Tuning parameters of pid controller based on fuzzy logic controlled genetic algorithms. In *Proceedings of the 6th International Symposium on Instrumentation and Control Technology*, volume 6358, pages 635849–635849–6, 2006.
- [114] X. Feng, A. C. Sanderson, P. P. Bonissone, and R. J. Graves. Fuzzy logic controlled multi-objective differential evolution. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ '05*, pages 720 –725, may 2005.
- [115] T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.*, 8(2):67–71, Apr. 1989.
- [116] T. A. Feo, M. G. C. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):pp. 860–878, 1994.
- [117] A. Fialho. *Adaptive Operator Selection for Optimization*. PhD thesis, Université Paris-Sud XI, Orsay, France, December 2010.
- [118] M. Fischetti, C. Lepschy, G. Minerva, G. Romanin-Jacur, and E. Toto. Frequency assignment in mobile radio systems using branch-and-cut techniques. *European Journal of Operational Research*, 123(2):241 – 255, 2000.

BIBLIOGRAPHY

- [119] M. Flynn. Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909, 1966.
- [120] D. B. Fogel. *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1st edition, 1998.
- [121] L. J. Fogel. Toward inductive inference automata. In *Proceedings of International Federation for Information Processing Congress*, pages 395 – 399, 1962.
- [122] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, Chichester, UK, 1966.
- [123] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. Morgan Kaufmann Publishers.
- [124] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
- [125] R. M. Friedberg. A Learning Machine: Part I. *IBM Journal of Research and Development*, 2(1):2–13, 1958.
- [126] G. Friedman. Select feedback computers for engineering synthesis and nervous system analogy. Master’s thesis, University of California, Los Angeles, USA, 1956.
- [127] M. Frutos and F. Tohmé. A multi-objective memetic algorithm for the job-shop scheduling problem. *Operational Research*, 13(2):233–250, 2013.
- [128] M. J. Gacto, R. Alcalá, and F. Herrera. Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. *Soft Computing*, 13(5):419–436, Dec. 2008.
- [129] M. J. Gacto, R. Alcalá, and F. Herrera. A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems. *Applied Intelligence*, 36(2):330–347, 2012.
- [130] A. García-Nájera. Preserving population diversity for the multi-objective vehicle routing problem with time windows. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Confer-*

- ence: *Late Breaking Papers*, GECCO '09, pages 2689–2692, New York, NY, USA, 2009. ACM.
- [131] P. Garg. A comparison between memetic algorithm and genetic algorithm for the cryptanalysis of simplified data encryption standard algorithm. *International Journal of Network Security & Its Applications*, 1(1):34 – 42, April 2009.
- [132] M. Garza-Fabre, E. Rodriguez-Tello, and G. Toscano-Pulido. An improved multiobjectivization strategy for HP model-based protein structure prediction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7492 LNCS(PART 2):82–92, 2012.
- [133] M. Garza-Fabre, G. Toscano-Pulido, and E. Rodriguez-Tello. Locality-based multiobjectivization for the HP model of protein structure prediction. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO'12, pages 473–480, New York, NY, USA, 2012. ACM.
- [134] A. Geist. *Advanced Tutorial on PVM 3.4: New Features and Capabilities*, 1997. <http://www.csm.ornl.gov/pvm/EuroPVM97/>.
- [135] M. Gen and Y. Yun. Soft computing approach for reliability optimization: State-of-the-art survey. *Reliability Engineering & System Safety*, 91(9):1008 – 1026, 2006.
- [136] C. Glaßer, S. Reith, and H. Vollmer. The complexity of base station positioning in cellular networks. *Discrete Applied Mathematics*, 148(1):1–12, 2005.
- [137] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.
- [138] F. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, May 1986.
- [139] F. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Springer US, 2003.
- [140] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1998.

BIBLIOGRAPHY

- [141] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
- [142] D. Goldsman, B. Nelson, and B. Schmeiser. Methods for selecting the best system [for simulation]. In *Proceedings of the 23rd Conference on Winter Simulation*, pages 177–186, Dec 1991.
- [143] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, Jan 1986.
- [144] D. Greiner, J. M. Emperador, G. Winter, and B. Galván. Improving Computational Mechanics Optimum Design Using Helper Objectives: An Application in Frame Bar Structures. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 575–589. Springer Berlin Heidelberg, 2007.
- [145] J. L. Gustafson. Reevaluating Amdahl’s law. *Commun. ACM*, 31(5):532–533, May 1988.
- [146] E. Hadavandi, H. Shavandi, A. Ghanbari, and S. Abbasian-Naghneh. Developing a hybrid artificial intelligence model for outpatient visits forecasting in hospitals. *Applied Soft Computing Journal*, 12(2):700–711, 2012.
- [147] S. Hajri, N. Liouane, S. Hammadi, and P. Borne. A controlled genetic algorithm by fuzzy logic and belief functions for job-shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(5):812–818, oct 2000.
- [148] W. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497 – 1514, dec. 1980.
- [149] J. Handl, D. B. Kell, and J. Knowles. Multiobjective Optimization in Bioinformatics and Computational Biology. *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, 4(2):279–292, 2007.
- [150] J. Handl, S. C. Lovell, and J. Knowles. Investigations into the Effect of Multiobjectivization in Protein Structure Prediction. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature - PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 702–711. Springer Berlin Heidelberg, 2008.

- [151] J. Handl, S. C. Lovell, and J. Knowles. Multiobjectivization by decomposition of scalar cost functions. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, pages 31–40, Berlin, Heidelberg, 2008. Springer-Verlag.
- [152] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation, 1996*, pages 312–317, May 1996.
- [153] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [154] F. Herrera. Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46, 2008.
- [155] F. Herrera and M. Lozano. Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions. *Soft Computing*, 7(8):545–562, 2003.
- [156] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 23–32. Springer Berlin Heidelberg, 1991.
- [157] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Phys. D*, 42(1-3):228–234, June 1990.
- [158] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. International Thomson Publishing, 1996.
- [159] J. H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, July 1962.
- [160] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.
- [161] T.-P. Hong, M.-W. Tsai, and T.-K. Liu. Two-dimensional encoding schema and genetic operators. In *JCIS*. Atlantis Press, 2006.
- [162] H. Hoos and T. Stützle. *Stochastic local search: foundations and applications*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers, 2005.

- [163] J. Horn, N. Nafpliotis, and D. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, 1994.
- [164] L. Idoumghar and R. Schott. A new hybrid GA-MDP algorithm for the frequency assignment problem. In *Proc. of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI'06)*, pages 18 – 25, 2006.
- [165] H. Ishibuchi. Multiobjective genetic fuzzy systems: Review and future research directions. In *Proceedings of the 2007 IEEE International Conference on Fuzzy Systems*, 2007.
- [166] H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of Scalarizing Fitness Functions into Evolutionary Multiobjective Optimization Algorithms. In *Proceedings of the 9th international conference on Parallel Problem Solving from Nature, PPSN'06*, pages 493–502, Berlin, Heidelberg, 2006. Springer-Verlag.
- [167] H. Ishibuchi, Y. Hitotsuyanagi, Y. Nakashima, and Y. Nojima. Multiobjectivization from Two Objectives to Four Objectives in Evolutionary Multi-Objective Optimization Algorithms. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*, pages 502–507, 2010.
- [168] H. Ishibuchi, Y. Hitotsuyanagi, and Y. Nojima. An empirical study on the specification of the local search application probability in multiobjective memetic algorithms. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pages 2788 –2795, sept. 2007.
- [169] H. Ishibuchi, T. Nakashima, and M. Nii. *Classification and Modeling with Linguistic Information Granules*. Advanced Information Processing. Springer Berlin Heidelberg, 2005.
- [170] H. Ishibuchi and Y. Nojima. Optimization of Scalarizing Functions through Evolutionary Multiobjective Optimization. In *Proceedings of the 4th international conference on Evolutionary multi-criterion optimization, EMO'07*, pages 51–65, Berlin, Heidelberg, 2007. Springer-Verlag.
- [171] W. Jackson, E. Özcan, and J. Drake. Late acceptance-based selection hyperheuristics for cross-domain heuristic search. In *Proceedings of the 13th UK Workshop on Computational Intelligence, UKCI 2013*, pages 228–235, 2013.
- [172] M. Jähne, X. Li, and J. Branke. Evolutionary Algorithms and Multi-Objectivization for the Travelling Salesman Problem. In *Proceedings of the*

- 11th Annual conference on Genetic and evolutionary computation*, GECCO'09, pages 595–602, New York, NY, USA, 2009. ACM.
- [173] D. Jakobović, M. Golub, and M. Čupić. Asynchronous and implicitly parallel evolutionary computation models. *Soft Computing*, pages 1–12, 2013.
- [174] M. Jamshidi, R. A. Krohling, L. d. S. Coelho, and P. J. Fleming. *Robust Control Systems with Genetic Algorithms*. Control Series. CRC, Boca Raton, FL, 2003.
- [175] T. Jansen and I. Wegener. Real royal road functions – where crossover probably is essential. *Discrete Applied Mathematics*, 149(1-3):111 – 125, 2005.
- [176] M. T. Jensen. Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. *Journal of Mathematical Modelling and Algorithms*, 3:323–347, 2004.
- [177] Y. Jin, J.-K. Hao, and J.-P. Hamiez. A memetic algorithm for the minimum sum coloring problem. *Computers and Operations Research*, 43:318–327, 2014.
- [178] K. Jong. Parameter setting in EAs: a 30 year perspective. In F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 1–18. Springer Berlin Heidelberg, 2007.
- [179] K. A. D. Jong. *Evolutionary computation - a unified approach*. MIT Press, 2006.
- [180] M. Kampouridis. An initial investigation of choice function hyper-heuristics for the problem of financial forecasting. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 2406–2413, 2013.
- [181] D. Karaboga and B. Basturk. Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4529 LNAI:789–798, 2007.
- [182] G. Kendall, P. Cowling, and E. Soubeiga. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL 2002)*, pages 667–671, Singapore, Nov 2002.
- [183] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

BIBLIOGRAPHY

- [184] S.-S. Kim, A. E. Smith, and J.-H. Lee. A memetic algorithm for channel assignment in wireless FDMA systems. *Computers & Operations Research*, 34:1842 – 1856, 2007.
- [185] D. B. Kirk and W.-m. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2010.
- [186] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [187] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto Archived Evolution Strategy. *Evol. Comput.*, 8(2):149–172, June 2000.
- [188] J. D. Knowles, R. A. Watson, and D. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, EMO '01, pages 269–283, London, UK, 2001. Springer-Verlag.
- [189] D. Knysh and V. Kureichik. Parallel genetic algorithms: A survey and problem state of the art. *Journal of Computer and Systems Sciences International*, 49(4):579–589, 2010.
- [190] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [191] O. Kramer. Evolutionary self-adaptation: a survey of operators and strategy parameters. *Evolutionary Intelligence*, 3:51–65, 2010.
- [192] N. Krasnogor and J. E. Smith. A memetic algorithm with self-adaptive local search: TSP as a case study. In L. D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, I. C. Parmee, and H.-G. Beyer, editors, *GECCO*, pages 987–994. Morgan Kaufmann, 2000.
- [193] E. Krempser, A. Fialho, and H. J. C. Barbosa. Adaptive operator selection at the hyper-level. In C. A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, volume 7492 of *Lecture Notes in Computer Science*, pages 378–387. Springer Berlin Heidelberg, 2012.
- [194] V. Kumar, A. Grama, A. Gupta, and G. Karypis, editors. *Introduction to Parallel Computing*. Addison-Wesley, 2003.

- [195] A. M. J. Kuurne. On GSM mobile measurement based interference matrix generation. In *IEEE 55th Vehicular Technology Conference, VTC Spring 2002*, pages 1965 – 1969, 2002.
- [196] A. Land and A. Doig. An automatic method for solving discrete programming problems. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer Berlin Heidelberg, 2010.
- [197] O. Lara and M. Labrador. A multiobjective ant colony-based optimization algorithm for the bin packing problem with load balancing. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, pages 1 –8, july 2010.
- [198] M. Lassouaoui and D. Boughaci. A choice function hyper-heuristic for the winner determination problem. *Studies in Computational Intelligence*, 512:303–314, 2014.
- [199] H. C. W. Lau, C. X. H. Tang, G. T. S. Ho, and T. M. Chan. A fuzzy genetic algorithm for the discovery of process parameter settings using knowledge representation. *Expert Syst. Appl.*, 36(4):7964–7974, May 2009.
- [200] M. N. Le, Y.-S. Ong, Y. Jin, and B. Sendhoff. Lamarckian memetic algorithms: local optimum and connectivity structure analysis. *Memetic Computing*, 1(3):175–190, 2009.
- [201] M. A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 76–83. Morgan Kaufmann, 1993.
- [202] J. Lehman and K. O. Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Proc. of the Eleventh Intl. Conf. on Artificial Life*, Cambridge, MA, 2008. MIT Press.
- [203] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189–223, June 2011.
- [204] C. León, G. Miranda, C. Rodríguez, and C. Segura. 2D Cutting Stock Problem: A New Parallel Algorithm and Bounds. In *Proceedings of Euro-Par*, volume 4641 of *LNCS*, pages 795–804. Springer-Verlag, 2007.
- [205] C. León, G. Miranda, and C. Segura. A memetic algorithm and a parallel hyperheuristic island-based model for a 2D packing problem. In *Proceedings of*

BIBLIOGRAPHY

- the 11th Annual conference on Genetic and evolutionary computation, GECCO '09*, pages 1371–1378, New York, NY, USA, 2009. ACM.
- [206] C. León, G. Miranda, and C. Segura. METCO: A Parallel Plugin-Based Framework for Multi-Objective Optimization. *International Journal on Artificial Intelligence Tools*, 18(4):569–588, 2009.
 - [207] Q. Li and Y. Maeda. Distributed adaptive search method for genetic algorithm controlled by fuzzy reasoning. In *IEEE International Conference on Fuzzy Systems, 2008. FUZZ-IEEE 2008*, pages 2022 –2027, june 2008.
 - [208] Z.-f. Z. Li-xiao Ma, Kun-qi Liu and N. Li. Exploring the effects of Lamarckian evolution and Baldwin effect in differential evolution. In *Communications in Computer and Information Science*, volume 107 of *Computational Intelligence and Intelligent Systems*, pages 127–136. Springer, 2010.
 - [209] R. F. Linton and T. B. Carroll. *Computational Optimization: New Research Developments*. Nova Science Publishers Inc, 2010.
 - [210] D. Liu and X. Liu. The improved genetic algorithm based on fuzzy controller with adaptive parameter adjustment. In M. Zhu, editor, *Information and Management Engineering*, volume 235 of *Communications in Computer and Information Science*, pages 491–497. Springer Berlin Heidelberg, 2011.
 - [211] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9:448–462, 2005.
 - [212] F. G. Lobo, C. F. Lima, and Z. Michalewicz. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2007.
 - [213] D. F. Lochtefeld. *Multi-objectivization in Genetic Algorithms*. PhD thesis, Wright State University, 2011.
 - [214] D. F. Lochtefeld and F. W. Ciarallo. Helper-objective optimization strategies for the job-shop scheduling problem. *Applied Soft Computing*, 11(6):4161 – 4174, 2011.
 - [215] D. F. Lochtefeld and F. W. Ciarallo. Multiobjectivization via helper-objectives with the tunable objectives problem. *IEEE Trans. on Evol. Comput.*, 16(3):373–390, 2012.

- [216] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, September 2002.
- [217] S. J. Louis and G. Rawlins. Pareto Optimality, GA-easiness and Deception. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 118–123. Morgan Kaufmann, 1993.
- [218] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer US, 2003.
- [219] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evol. Comput.*, 12(3):273–302, Sept. 2004.
- [220] M. Lozano, D. Molina, and F. Herrera. Editorial Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-scale Continuous Optimization Problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pages 1–3, 2010.
- [221] F. Luiz Usberti, P. Morelato França, and A. L. M. França. GRASP with evolutionary path-relinking for the capacitated ARC routing problem. *Computers and Operations Research*, 40(12):3206–3217, 2013.
- [222] F. Luna, C. Blum, E. Alba, and A. Nebro. ACO vs EAs for solving a real-world frequency assignment problem in GSM networks. In *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, pages 94–101, 2007.
- [223] F. Luna, C. Estébanez, C. León, J. Chaves-González, A. Nebro, R. Aler, C. Segura, M. Vega-Rodríguez, E. Alba, J. Valls, G. Miranda, and J. Gómez-Pulido. Optimization algorithms for large-scale real-world instances of the frequency assignment problem. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15:975–990, 2011.
- [224] F. Luna, C. Estébanez, C. León, J. M. Chaves-González, E. Alba, R. Aler, C. Segura, M. A. Vega-Rodríguez, A. J. Nebro, J. M. Valls, G. Miranda, and J. A. Gómez-Pulido. Metaheuristics for solving a real-world frequency assignment problem in GSM networks. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1579–1586. ACM, 2008.

BIBLIOGRAPHY

- [225] T. Lust and J. Teghem. The multiobjective multidimensional knapsack problem: A survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.
- [226] Y. Maeda and Q. Li. Fuzzy adaptive search method for parallel genetic algorithm tuned by evolution degree based on diversity measure. In P. Melin, O. Castillo, L. Aguilar, J. Kacprzyk, and W. Pedrycz, editors, *Foundations of Fuzzy Logic and Soft Computing*, volume 4529 of *Lecture Notes in Computer Science*, pages 677–687. Springer Berlin Heidelberg, 2007.
- [227] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *PPSN*, pages 27–36. Elsevier, 1992.
- [228] E. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Electrical Engineers, Proceedings of the Institution of*, 121(12):1585–1588, 1974.
- [229] C. Mannino and A. Sassano. An enumerative algorithm for the frequency assignment problem. *Discrete Appl. Math.*, 129(1):155–169, June 2003.
- [230] E. Marchiori. Genetic, iterated and multistart local search for the maximum clique problem. In *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, pages 112–121, London, UK, UK, 2002. Springer-Verlag.
- [231] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artif. Intell. Rev.*, 11(1-5):193–225, Feb. 1997.
- [232] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319, 2003.
- [233] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons Ltd, 1990.
- [234] O. C. Martin and S. W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.
- [235] O. C. Martin, S. W. Otto, and E. W. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
- [236] M. Matayoshi. Two dimensional rectilinear polygon packing using genetic algorithm with a hierarchical chromosome. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 989–996, 2011.

- [237] K. E. Mathias and L. D. Whitley. Changing representations during search: A comparative study of delta coding. *Evol. Comput.*, 2(3):249–278, Sept. 1994.
- [238] S. Matsui, I. Watanabe, and K.-I. Tokoro. Application of the parameter-free genetic algorithm to the fixed channel assignment problem. *Systems and Computers in Japan*, 36(4):71 – 81, 2005.
- [239] J. Maturana, F. Lardeux, and F. Saubion. Controlling behavioral and structural parameters in evolutionary algorithms. In P. Collet, N. Monmarché, P. Legrand, M. Schoenauer, and E. Lutton, editors, *Artificial Evolution*, volume 5975 of *Lecture Notes in Computer Science*, pages 110–121, Strasbourg, France, October 26-28 2009. Springer.
- [240] S. McClintock, T. Lunney, and A. Hashim. A fuzzy logic controlled genetic algorithm environment. In *IEEE International Conference on Systems, Man, and Cybernetics, 1997*, volume 3, pages 2181 –2186 vol.3, oct 1997.
- [241] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez. Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Systems with Applications*, 40(8):3196 – 3206, 2013.
- [242] S. P. Mendes, G. Molina, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, Y. Sáez, G. Miranda, C. Segura, E. Alba, P. Isasi, C. León, and J. M. Sánchez-Pérez. Benchmarking a wide spectrum of metaheuristic techniques for the radio network design problem. *IEEE Transactions on Evolutionary Computation*, 13(5):1133–1150, Oct. 2009.
- [243] R. E. Mercer and J. R. Sampson. Adaptive search using a reproductive metaplan. *Kybernetes*, 7(3):215–228, 1978.
- [244] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard. Version 3.0*, 2012. <http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf>.
- [245] S. Meyer-Nieberg and H.-G. Beyer. Self-adaptation in evolutionary algorithms. In F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 47–75. Springer Berlin Heidelberg, 2007.
- [246] E. Mezura-Montes and C. A. Coello Coello. Constrained Optimization via Multiobjective Evolutionary Algorithms. In J. Knowles, D. Corne, K. Deb,

BIBLIOGRAPHY

- and D. Chair, editors, *Multiobjective Problem Solving from Nature*, Natural Computing Series, pages 53–75. Springer Berlin Heidelberg, 2008.
- [247] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Berlin Heidelberg, 1996.
- [248] Z. Michalewicz, D. Dasgupta, R. G. L. Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Evolutionary Computation*, 4:1–32, 1996.
- [249] K. Miettinen and M. M. Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24(2):193–213, 2002.
- [250] N. Mladenovic. A variable neighborhood algorithm – a new metaheuristic for combinatorial optimization. In *Abstracts of Papers Presented at Optimization Days*, 1995.
- [251] E. Montero, M.-C. Riff, and B. Neveu. An evaluation of off-line calibration techniques for evolutionary algorithms. In *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, pages 299–300, 2010.
- [252] G. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.
- [253] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts. Towards Memetic Algorithms. Technical Report 158-79, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, California, September 1989.
- [254] M. Mouly and M. B. Paulet. *The GSM System for Mobile Communications*. Mouly et Paulet, Palaiseau, 1992.
- [255] J. B. Mouret. Novelty-based multiobjectivization. In S. Doncieux, N. Bredèche, and J. B. Mouret, editors, *New Horizons in Evolutionary Robotics*, volume 341 of *Studies in Computational Intelligence*, pages 139–154. Springer Berlin / Heidelberg, 2011.
- [256] J. B. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *2009 IEEE Congress on Evolutionary Computation, CEC'09*, pages 1161–1168, 2009.
- [257] J. B. Mouret and S. Doncieux. Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. In *Proceedings of the 11th An-*

- nual conference on Genetic and evolutionary computation*, GECCO'09, pages 627–634, New York, NY, USA, 2009. ACM.
- [258] J. B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput.*, 20(1):91–133, 2012.
- [259] M. Mucientes, D. Moreno, A. Bugarín, and S. Barro. Design of a fuzzy controller in mobile robotics using genetic algorithms. *Applied Soft Computing Journal*, 7(2):540–546, 2007.
- [260] R. Myers and E. R. Hancock. Empirical modelling of genetic algorithms. *Evol. Comput.*, 9(4):461–493, Dec. 2001.
- [261] V. Nannen and A. E. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 183–190, New York, NY, USA, 2006. ACM.
- [262] C. Navarro, N. Hitschfeld-Kahler, and L. Mateu. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. *Communications in Computational Physics*, 15(2):285–329, 2014.
- [263] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2007.
- [264] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Nat. Comput.*, 5(3):305–319, 2006.
- [265] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. Non-genetic transmission of memes by diffusion. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1017–1024, New York, NY, USA, 2008. ACM.
- [266] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. A Probabilistic Memetic Framework. *IEEE Transactions on Evolutionary Computation*, 13(3):604–623, 2009.
- [267] Q. U. Nguyen, X. H. Nguyen, M. O'Neill, and A. Agapitos. An investigation of fitness sharing with semantic and syntactic distance metrics. In A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, editors, *EuroGP*, volume 7244 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2012.

BIBLIOGRAPHY

- [268] S. Nguyen, M. Zhang, M. Johnston, and T. K. Chen. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. In X. Li, editor, *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 3332–3339, Brisbane, Australia, 10-15 June 2012.
- [269] J. Ni, L. Li, F. Qiao, and Q. Wu. A novel memetic algorithm and its application to data clustering. *Memetic Computing*, 5(1):65–78, 2013.
- [270] J. Nievergelt. Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*, SOFSEM '00, pages 18–35, London, UK, 2000. Springer-Verlag.
- [271] OpenMP Architecture Review Board. *OpenMP Application Program Interface. Version 4.0*, 2013. <http://www.openmp.org>.
- [272] A. Osyczka. Multicriteria optimization for engineering design. In J. S. Gero, editor, *Design Optimization*, pages 193 – 227. Academic Press, 1985.
- [273] D. Ouelhadj and S. Petrovic. A cooperative hyper-heuristic search framework. *Journal of Heuristics*, 16(6):835–857, 2010.
- [274] E. Özcan and C. Basaran. A case study of memetic algorithms for constraint optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13(8):871–882, 2009.
- [275] P. S. Pacheco. *An Introduction to Parallel Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2011.
- [276] V. Pareto. *Cours d'Économie Politique*, volume I and II. Lausanne, 1896.
- [277] D. A. Patterson and J. L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [278] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, New York, April 1984.
- [279] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1988.
- [280] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Trans. Evol. Comp*, 13(2):398–417, Apr. 2009.

- [281] J. Rada-Vilela. Fuzzylite: a fuzzy logic control library in C++, 2013. <http://www.fuzzylite.com>.
- [282] P. Rattadilok, A. Gaw, and R. Kwan. Distributed choice function hyper-heuristics for timetabling and scheduling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin / Heidelberg, 2005.
- [283] I. Rechenberg. Cybernetic solution path of an experimental problem. In *Library Translation 1122*, Farnborough: Royal Aircraft Establishment, 1965.
- [284] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog-Verlag, Stuttgart, 1973.
- [285] Z. Ren, H. Jiang, J. Xuan, and Z. Luo. Hyper-heuristics with low level parameter adaptation. *Evol. Comput.*, 20(2):189–227, June 2012.
- [286] R. G. Reynolds. An introduction to cultural algorithms. In A. V. Sebald and F. L. J., editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131 – 139. World Scientific, 1994.
- [287] C. Ribeiro, I. Rosseti, and R. Souza. Probabilistic stopping rules for GRASP heuristics and extensions. *International Transactions in Operational Research*, 20(3):301–323, 2013.
- [288] Y. Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics - Theory and Methods*, 7(8):799–811, 1978.
- [289] F. Rossi, P. v. Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.
- [290] O. Rui, A. Hajizadeh, and T. M. Undeland. Parameter optimization of a fuzzy logic controller for a power electronics boost converter using genetic algorithms. In *Proceedings of the 9th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering, and Data Bases, AIKED’10*, pages 120–124, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).
- [291] A. Saini and A. Saraswat. Multi-objective reactive power market clearing in competitive electricity market using HFMOEA. *Applied Soft Computing*, 13(4):2087 – 2103, 2013.

BIBLIOGRAPHY

- [292] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, Tennessee, 1984.
- [293] J. Scharnow, K. Tinnefeld, and I. Wegener. The Analysis of Evolutionary Algorithms on Sorting and Shortest Paths Problems. *J. of Math. Model. and Algorithms*, 3(4):349–366, 2005.
- [294] B. Schmeiser. Simulation experiments. In D. P. Heyman and M. J. Sobel, editors, *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, chapter 7, pages 295–330. Elsevier, July 1990.
- [295] H.-P. Schwefel. *Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 1965.
- [296] C. Segura. *Parallel Optimisation Schemes. A Hybrid Scheme based on Hyperheuristics and Evolutionary Computation*. PhD thesis, Universidad de La Laguna, La Laguna, Spain, November 2012.
- [297] C. Segura, C. A. Coello Coello, G. Miranda, and C. León. Using multi-objective evolutionary algorithms for single-objective optimization. *4OR*, 11(3):201–228, 2013.
- [298] C. Segura, G. Miranda, and C. León. Parallel Hyperheuristics for the Frequency Assignment Problem. *Memetic Computing*, 3(1):33–49, 2010.
- [299] J. Seybold. *Introduction to RF Propagation*. Wiley-Interscience, 2005.
- [300] D. Sharma, K. Deb, and N. Kishore. Customized evolutionary optimization procedure for generating minimum weight compliant mechanisms. *Engineering Optimization*, 46(1):39–60, 2014.
- [301] M. Sheikhan and S. Ghoreishi. Antiviral therapy using a fuzzy controller optimized by modified evolutionary algorithms: A comparative study. *Neural Computing and Applications*, 23(6):1801–1813, 2013.
- [302] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 5 edition, 2007.
- [303] Y. Shi. Combinations of evolutionary algorithms and fuzzy systems: A survey. In *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, pages 610–614, 1999.
- [304] Y. Shi and D. Ye. On-line bin packing with arbitrary release times. In *First International Symposium on Combinatorics, Algorithms, Probabilistic and Ex-*

- perimental Methodologies*, volume 4614 of *LNCS*, pages 340–349. Hangzhou, China, Springer Berlin, April 2007.
- [305] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.
- [306] A. Simões and E. Costa. Memory-based CHC algorithms for the dynamic traveling salesman problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1037–1044, New York, NY, USA, 2011. ACM.
- [307] M. K. Simon and M.-S. Alouini. *Digital Communication Over Fading Channels: A Unified Approach to Performance Analysis*. John Wiley & Sons, Inc., New York, USA, 2002.
- [308] K. Sindhya, A. Sinha, K. Deb, and K. Miettinen. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In *Proceedings of the Eleventh Congress on Evolutionary Computation*, CEC'09, pages 2919–2926, Piscataway, NJ, USA, 2009. IEEE Press.
- [309] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the Eleventh Congress on Evolutionary Computation*, CEC'09, pages 399–406, Piscataway, NJ, USA, 2009. IEEE Press.
- [310] S. K. Smit, A. E. Eiben, and Z. Szilávik. An MOEA-based method to tune EA parameters on multiple objective functions. In J. Filipe and J. Kacprzyk, editors, *IJCCI (ICEC)*, pages 261–268. SciTePress, 2010.
- [311] J. E. Smith. *Self Adaptation in Evolutionary Algorithms*. PhD thesis, University of the West of England, Bristol, UK, 1998.
- [312] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1:81–87, 1997.
- [313] Y. H. Song, G. Wang, P. Wang, and A. Johns. Environmental/economic dispatch using fuzzy logic controlled genetic algorithms. *IEE Proceedings – Generation, Transmission and Distribution*, 144(4):377–382, Jul 1997.
- [314] E. Soubeiga. *Development and application of hyperheuristics to personnel scheduling*. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, United Kingdom, 2003.

BIBLIOGRAPHY

- [315] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 367 – 384. MIT Press, 1995.
- [316] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):656 –667, apr 1994.
- [317] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
- [318] W. Stadler. Fundamentals of multicriteria optimization. In W. Stadler, editor, *Multicriteria Optimization in Engineering and the Sciences*, pages 1–25. Plenum Press, New York, 1988.
- [319] R. Storn and K. Price. Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization*, 11(4):341–359, Dec. 1997.
- [320] R. J. Streifel. Dynamic fuzzy control of genetic algorithm parameter coding, 1999.
- [321] R. Subbu, A. Sanderson, and P. Bonissone. Fuzzy logic controlled genetic algorithms versus tuned genetic algorithms: an agile manufacturing application. In *Intelligent Control (ISIC)*, 1998, pages 434 –440, sep 1998.
- [322] T. A. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, 2006.
- [323] G. Sywerda. Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [324] G. Taguchi and T. Yokoyama. *Taguchi Methods: Design of Experiments*. ASI Press, 1993.
- [325] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(1):116–132, 1985.
- [326] E.-G. Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009.
- [327] E.-G. Talbi and H. Meunier. Hierarchical parallel approach for GSM mobile network design. *J. Parallel Distrib. Comput.*, 66(2):274–290, 2006.

- [328] The GCC team. *The GNU Compiler Collection*, 2013. <http://gcc.gnu.org/>.
- [329] The Open MPI Project. *Open MPI v1.6.4 documentation*, 2013. <http://www.open-mpi.org/doc/v1.6/>.
- [330] A. Toffolo and E. Benini. Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation*, 11:151–167, May 2003.
- [331] G. Toscano Pulido and C. A. Coello Coello. The Micro Genetic Algorithm 2: Towards online adaptation in evolutionary multiobjective optimization. In C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 252–266. Springer Berlin Heidelberg, 2003.
- [332] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.
- [333] V. Vakula and K. Sudha. Design of differential evolution algorithm-based robust fuzzy logic power system stabiliser using minimum rule base. *Generation, Transmission Distribution, IET*, 6(2):121–132, 2012.
- [334] Van Der Steen, A. and Dongarra, J. Overview of recent supercomputers, 2007.
- [335] V. Vazirani. *Approximation Algorithms*. Springer Berlin Heidelberg, 2003.
- [336] J. A. Vázquez-Rodríguez and S. Petrovic. A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics*, 16(6):771–793, Dec. 2010.
- [337] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, July 2013.
- [338] D. A. V. Veldhuizen, J. B. Zydallis, and G. B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 7(2):144–173, 2003.
- [339] T. Vinkó and D. Izzo. Learning the best combination of solvers in a distributed global optimization environment. In *Proceedings of Advances in Global Optimization: Methods and Applications (AGO)*, pages 13–17, Mykonos, Greece, June 2007.

BIBLIOGRAPHY

- [340] K. V. Viswanathan and A. Bagchi. Best-First Search Methods for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research*, 41(4):768–776, 1993.
- [341] C. Voudouris. Guided local search – an illustrative example in function optimisation. *BT Technology Journal*, 16(3):46–50, 1998.
- [342] B. H. Walke. *Mobile Radio Networks: Networking, protocols and traffic performance*. John Wiley & Sons, Ltd., West Sussex, England, 2002.
- [343] P. Wang, K. Tang, T. Weise, E. Tsang, and X. Yao. Multiobjective genetic programming for maximizing ROC performance. *Neurocomputing*, 125:102–118, 2014.
- [344] P. Wang, G. Wang, and Z. Hu. Speeding up the search process of genetic algorithm by fuzzy logic. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, pages 665–671, 1997.
- [345] Y. Wang, J.-K. Hao, F. Glover, and Z. Lü. A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence*, 27:103–114, 2014.
- [346] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, December 2007.
- [347] S. Watanabe and K. Sakakibara. A multiobjectivization approach for vehicle routing problems. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 660–672. Springer Berlin / Heidelberg, 2007.
- [348] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer. Evolutionary multi-objective optimization for base station transmitter placement with frequency assignment. *IEEE Transactions on Evolutionary Computation*, 7(2):189–203, April 2003.
- [349] L. D. Whitley, V. S. Gordon, and K. E. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 6–15, London, UK, 1994. Springer-Verlag.

- [350] G. V. Wilson. *Practical Parallel Programming*. MIT Press, Cambridge, MA, USA, 1996.
- [351] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [352] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the VI International Congress of Genetics*, pages 356–366, 1932.
- [353] F. Xhafa, B. Duran, L. Barolli, V. Kolici, R. Miho, and M. Takizawa. Tuning of operators in memetic algorithms for independent batch scheduling in computational grids. In *Proceedings of the 6th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2012*, pages 335–342, 2012.
- [354] B. Yuan and M. Gallagher. Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks. In F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 121–142. Springer Berlin Heidelberg, 2007.
- [355] Z.-H. Zhan and J. Zhang. Adaptive particle swarm optimization. In *Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, ANTS '08, pages 227–234, Berlin, Heidelberg, 2008. Springer-Verlag.
- [356] C. Zhang, J. Chen, and B. Xin. Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning. *Applied Soft Computing Journal*, 13(5):2947–2959, 2013.
- [357] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evol. Comp.*, 1(1):32 – 49, 2011.
- [358] Y. Zhou, Y. Rao, G. Zhang, and C. Zhang. An adaptive memetic algorithm for packing problems of irregular shapes. *Advanced Materials Research*, 314-316:1029–1033, 2011.
- [359] K. Zielinski and R. Laur. Adaptive parameter setting for a multi-objective particle swarm optimization algorithm. In *The 2007 IEEE Congress on Evolutionary Computation*, pages 3019 –3026, sept. 2007.
- [360] K. Zielinski and R. Laur. Stopping criteria for differential evolution in constrained single-objective optimization. In U. Chakraborty, editor, *Advances in*

BIBLIOGRAPHY

- Differential Evolution*, volume 143 of *Studies in Computational Intelligence*, pages 111–138. Springer Berlin / Heidelberg, 2008.
- [361] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *VIII Conference on Parallel Problem Solving from Nature*, volume 3242 of *LNCS*, pages 832–842. Springer, 2004.
- [362] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design, Optimization and Control*, pages 19–26, Barcelona, Spain, 2002. CIMNE.
- [363] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.