HAL
open science

Hybrid Evolutionary Metaheuristics for Multiobjective Decision Support

Ahmed Kafafy

To cite this version:

Ahmed Kafafy. Hybrid Evolutionary Metaheuristics for Multiobjective Decision Support. Data Structures and Algorithms [cs.DS]. Université Claude Bernard - Lyon I, 2013. English. NNT : 2013LYO10184 . tel-01174720

HAL Id: tel-01174720

https://theses.hal.science/tel-01174720v1

Submitted on 9 Jul 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N d'ordre: 184-2013 Année: 2013

THÈSE

DE L'UNIVERSITÉ DE LYON

Délivrée par:

L'UNIVERSITÉ CLAUDE BERNARD LYON I

ÉCOLE DOCTORALE INFORMATIQUE ET MATHÉMATIQUES

DIPLOME DE DOCTORAT
(arrêté du 7 août 2006)

Soutenue publiquement le JEUDI 24 Octobre 2013

présentée par:

Ahmed M. KAFAFY

Hybrid Evolutionary Metaheuristics for Multiobjective Decision Support

"Métaheuristiques hybrides évolutionnaires pour l'aide à la décision multi-objectifs"

préparée au sein du laboratoire

sous la direction de:

M. Stéphane BONNEVAY et **M. Ahmed BOUNEKKAR**

JURY: **M. Jin-Kao HAO**, Pr., Université d'Angers, (*Rapporteur*)
Mme. Laetitia JOURDAN, Pr., Université de Lille 1, (*Rapporteur*)
M. Motaz KHORSHID, Pr., Université du Caire, Egypte, (*Examinateur*)
M. Ridha MAHJOUB, Pr., Université Paris Dauphine, (*Examinateur*)
M. Stéphane BONNEVAY, MCF HDR, Université Lyon 1, (*Directeur de thèse*)
M. Ahmed BOUNEKKAR, MCF, Université Lyon 1, (*Co-Directeur*)

Acknowledgments

First of all, I thank my God for helping me to achieve this work and giving me the ability to finish this thesis in that satisfactory form.

Second, I would like to express my sincere appreciation to my supervisors Stéphane Bonnevay and Ahmed Bounekkar for their continuous support, encouragement and comprehension during the research study in this thesis. They really influenced my way of thinking and developing the research ideas adopted in this thesis. I am very grateful for their strong effort with me and their highly useful advice through out the development of this work. Really, I can't find the appropriate words to thank them.

It is a great pleasure for me to express my sincere appreciation and my thanks to Prof. Jin-Kao Hao and Prof. Laetitia Jourdan for their accepting to be the reviewers of this work and for their useful and constructive comments. besides, I would like to express my thanks to Prof. Motaz Khorshid and Prof. Ridha Mahjoub for their accepting to be the jury examiners and for their valuable remarks.

I would like also to express my deepest thanks to my colleges and all the members of the ERIC laboratory for their cooperation during the period I spend with them to prepare this work.

I can't forget my dearest prof. Waiel F. AbdelWahed, the man who learn me not only how to make scientific research but also more diverse things in my practical life. I can't also forget my dearest father Abdalla Al-Hosainy who has a greatest place in my heart. Really, I lost both of them during preparation of this thesis. I ask my God almighty to bless their souls.

I am extremely grateful to all my friends especially Emad Elabd, Abdel-Alim Kamal, Marouane Hachicha, Rashed Salem, Waleed Ali, Mohamed AbuElAtta and Hazem Hassan and also all my professors and colleges in the faculty of computers and information in Menoufia university.

I am extremely grateful to my family especially my father, mother, my brothers, my sisters, my wife's family and my sincere wife Hayam who help and encourage me during the whole period of this thesis. Special acknowledgment goes to her. Really, I am so lucky with her love. Without her support, encouragement and patience I would not have been able to finish this work.

Finally, I dedicate this work to my children Mohammed, Alaa and Jana.

Ahmed Kafafy

Abstract

Many real-world decision making problems consist of - in most cases - several conflicting objectives, the solutions of which is a set of trade-offs called the Pareto-optimal set. Recently, hybrid evolutionary metaheuristics based algorithms proved their efficiency and effectiveness in handling this class of problems. They have been utilized to find a good approximation to the Pareto-optimal set. However, the approximation set must possess solutions with high convergence towards the Pareto-optimal set and hold a good diversity in order to demonstrate a good approximation. Hybrid evolutionary metaheuristics tend to enhance search capabilities, by improving intensification and diversification, through incorporating different cooperative metaheuristics. So that, we are concerned with developing a new search strategy through incorporating different search strategies such that the new strategy exploits the advantages as well as avoid the drawbacks of the original strategies. In this thesis, new hybrid evolutionary metaheuristics approaches are developed based on the framework of the MOEA/D [Zhang & Li 2007].

First, the Hybrid Evolutionary Metaheuristics (HEMH) is proposed. In HEMH, the search process is divided into two phases. In the first phase, the DM-GRASP is applied to obtain an initial set of high quality solutions dispersed along the Pareto front. Then, the search efforts are intensified on the promising regions around these solutions through the second phase. The greedy randomized path-relinking with local search or reproduction operators are applied to improve the quality and to guide the search process to explore the non-discovered regions in the search space. The two phases are combined with a suitable evolutionary framework for supporting the integration and cooperation. Moreover, the efficient solutions explored over the search are collected in an external archive.

Second, a comparative study is developed to study the effect of the hybridization of different metaheuristics with MOEA/D framework. four proposals of hybridization are considered in this study, the first proposal is to combine the adaptive discrete differential evolution operator with MOEA/D. The second one is to combine the greedy path-relinking operator with MOEA/D. the third and the fourth proposals combine both of them with MOEA/D using two different ways.

Third, based on the results of the developed comparative study, an improved hybrid evolutionary metaheuristics (HEMH2) is presented. Unlike HEMH, HEMH2 uses simple inverse greedy algorithm to construct its initial population. Then, the search efforts are directed to improve these solutions by exploring the search

space using the adaptive binary differential evolution. After a certain number of evaluations, greedy path relinking is applied on high quality solutions to investigate the non-visited regions in the search space. During evaluations, the dynamic-sized neighborhood structure is adopted to shrink/extend the mating/updating range. Furthermore, the Pareto adaptive epsilon concept is used to control the archiving process with preserving the extreme solutions.

Motivated by the results achieved by the previous proposed algorithms for the discrete search domains, a new hybrid evolutionary approach with search strategy adaptation (HESSA) is proposed to deal with the continuous search domains. In HESSA, the search process is carried out through adopting a pool of different search strategies, each of which has a specified success ratio. A new offspring is generated using a randomly selected strategy. Then, according to the success of the generated offspring to update the population or the archive, the success ratio of the selected strategy is adapted. This provides the ability for HESSA to adopt the appropriate search strategy according to the problem on hand. Furthermore, the cooperation among different strategies leads to improve the exploration and the exploitation of the search space. The proposed pool is combined to a suitable evolutionary framework for supporting the integration and cooperation. Moreover, the efficient solutions explored over the search are collected in an external repository to be used as global guides to the search process.

All the proposed algorithms are verified and tested against some of the state of the art MOEAs using a set of test instances commonly used in the literature. The experimental results indicate that all proposals are highly competitive and can be considered as viable alternatives.

Keywords:
Multiobjective Optimization; Hybrid Metaheuristics; Evolutionary Algorithms; DM-GRASP; Path-relinking; Adaptive Binary Differential Evolution; 0/1 Multiobjective Knapsack Problems; Search Strategy adaptation.

Résumé[1]

La prise de décision est une partie intégrante de notre vie quotidienne. Lors de ces prises de décision, le décideur est confronté le plus souvent à des problèmes composés de plusieurs objectifs habituellement contradictoires. Dans ce travail de thèse, nous traitons des problèmes d'optimisation multiobjectif dans des espaces de recherche continus ou discrets. Dans l'optimisation multiobjectif, il n'y a généralement pas de solution optimale qui satisfasse tous les objectifs à la fois. Il faut donc trouver des solutions de compromis, non "dominées" par d'autres solutions, appelées solutions Pareto optimales. Les méthodes classiques pour générer les solutions Pareto optimales agrègent généralement les fonctions objectifs en une seule fonction et souffrent de plusieurs limitations. Récemment, des algorithmes basés sur les métaheuristiques hybrides évolutionnaires ont prouvé leur efficacité dans le traitement de ce type de problèmes et ont permis de lever certaines de ces limitations. Ces algorithmes permettent de manipuler et de générer des ensembles de solutions (populations de solutions) qui convergent tout le long de l'ensemble des solutions Pareto optimales. Dans cette thèse, nous avons développé plusieurs nouveaux algorithmes basés sur les métaheuristiques hybrides évolutionnaires et en particulier sur l'algorithme de décomposition MOEA/D [Zhang & Li 2007].

Tout d'abord, nous avons proposé l'algorithme HEMH (Hybrid Evolutionary MetaHeuristic). HEMH utilise l'algorithme DM-GRASP pour construire une population initiale de solutions de bonne qualité dispersées le long de l'ensemble des solutions Pareto optimales. La phase de génération de populations successives à partir de cette population initiale est assurée par, soit des opérateurs de reproduction (croisement, mutation), soit par l'algorithme du Path Relinking glouton randomisé. L'ensemble des "bonnes" solutions trouvées au cours de cette phase est collecté dans une archive externe. Les résultats montrent que HEMH génère un ensemble de solutions de grande qualité. Une étude comparative a été réalisée pour étudier l'effet de l'hybridation de différentes méthodes de génération de solutions. Quatre variantes d'hybridation ont été étudiées: MOEAD_{de}, MOEAD_{pr}, MOEAD_{dp1} et MOEAD_{dp2}. Les résultats expérimentaux montrent la supériorité de toutes les variantes hybrides proposées sur les algorithmes originaux: MOEA/D et SPEA2. Malgré ces bons résultats, notre approche possède quelques limitations qui sont levées dans une version améliorée de HEMH: HEMH2 et deux autres variantes, appelées HEMH_{de} et HEMH_{pr}.

Contrairement à HEMH, HEMH2 utilise un algorithme glouton inverse simple

[1] Un résumé détaillé de la thèse en français est disponible à la Page 187 (Appendix A).

pour constituer la population initiale. La phase de génération de populations est assurée cette fois-ci par la combinaison de l'algorithme du Path Relinking et du Adaptive Binary DE. Pendant cette phase, une structure de voisinage dynamique est adoptée pour réduire/augmenter le croisement/mise à jour des solutions. En outre, le concept "pareto adaptive epsilon" est utilisé pour contrôler le processus d'archivage avec la préservation des solutions extrêmes. Les résultats expérimentaux montrent la supériorité de toutes les propositions HEMH2 et HEMH_{de} sur la MOEA/D et HEMH. Il est clair que, le Adaptive Binary DE inclus dans les HEMH2 et HEMH_{de} a de meilleures capacités d'exploration qui pallient aux capacités de recherche locales contenues dans la HEMH original. Ainsi, HEMH2 et HEMH_{de} surpassent HEMH.

Motivés par ces résultats dans un espace discret, nous avons proposé un nouvel algorithme baptisé HESSA (Hybrid Evolutionary approach with Search Strategy Adaptation) pour explorer un espace continu de recherche. Dans HESSA, le processus de recherche, pendant le processus d'évolution, peut être réalisé par différentes stratégies de recherche. Cela donne la possibilité à HESSA d'adopter la stratégie de recherche appropriée en fonction du problème étudié. En outre, la coopération entre les différentes stratégies conduit à améliorer l'exploration et l'exploitation de l'espace de recherche. L'ensemble proposé est combiné à un cadre évolutif adapté pour favoriser l'intégration et la coopération. L'ensemble des "bonnes" solutions trouvées au cours de la recherche est stocké dans un référentiel externe qui est utilisé comme guide global. Les résultats expérimentaux montrent la supériorité de HESSA à la fois sur MOEA/D et dMOPSO dans la plupart des tests.

Tous les algorithmes proposés ont été vérifiés, testés et comparés à certaines méthodes de l'état de l'art des MOEAs, en utilisant un ensemble d'exemples couramment utilisés dans la littérature. Les résultats expérimentaux indiquent que toutes les propositions sont très compétitives et peuvent être onsidérés comme une alternative fiable.

Mots-clés:

Optimisation multi-objectifs; Métaheuristiques hybrides; Algorithmes évolutionnaires; Problèmes du sac à dos multi-objectifs 0/1; Adaptation de la stratégie de recherche.

Contents

Acknowledgments **i**

Abstract **iii**

Résumé **v**

Contents **vii**

List of Figures **xiii**

List of Tables **xvii**

List of Algorithms **xix**

List of Acronyms **xxi**

Nomenclatures **xxiii**

1 Introduction **1**

1.1 Overview 1

1.2 Problem Statement 2

1.3 Research Objectives 3

1.4 Thesis Methodology 4

1.5 Thesis Contribution 4

1.6 Thesis Organization 6

2 Multiobjective Decision making **7**

2.1 Introduction 7

2.2 Single objective vs. multiple objectives 8

2.3 Multiobjective Optimization Problem 9

2.3.1 Pareto optimality 10

2.3.2 Weakly and Strictly Efficient Solutions 12

2.3.3 Bounds on the Pareto-optimal set 14
2.3.4 Pareto Epsilon optimality 14
2.4 Multiobjective Knapsack Problem 15
2.4.1 Problem Statement 15
2.5 Classification of multiobjective methods 16
2.6 Quality Assessment 19
2.6.1 Concepts and definitions 20
2.6.2 Assessment indicators 20
2.7 Classical Methods for MODM 23
2.7.1 Scalarization Techniques 23
2.7.2 Interactive methods 30
2.7.3 Goal programming methods 31
2.7.4 Fuzzy programming methods 32
2.8 Drawbacks of classical Methods 33

3 Optimization Metaheuristics: A Survey 35
3.1 Introduction 36
3.2 General Overview 36
3.3 Metaheuristics 39
3.4 Classification of Metaheuristics 41
3.5 Single-solution based metaheuristics 43
3.5.1 Greedy Heuristics 43
3.5.2 Simple Local Search 43
3.5.3 GRASP 46
3.5.4 Iterated Local Search 49
3.5.5 Variable Neighborhood Search 50
3.5.6 Guided Local Search 51
3.5.7 Simulated Annealing 54
3.5.8 Tabu Search 55
3.6 Population-based metaheuristics 57
3.6.1 Evolutionary computations 57
3.6.2 Other Evolutionary Algorithms 63

3.6.3 Swarm Intelligence 70
3.7 Multiobjective Metaheuristics 75
3.7.1 Fitness assignment strategies 75
3.7.2 Diversity preservation 77
3.7.3 Elitism 77
3.7.4 Multiobjective Evolutionary Algorithms 78
3.8 Hybrid Metaheuristics 83
3.8.1 Hybrid Metaheuristics classifications 85
3.8.2 Multiobjective Hybrid Metaheuristics 87
3.9 Summary 90

4 HEMH: A Hybrid Evolutionary Metaheuristics 91
4.1 Introduction 91
4.2 A review of GRASP and data mining 93
4.2.1 GRASP Algorithm 94
4.2.2 GRASP with Data Mining (DM-GRASP) 94
4.3 Path Relinking 95
4.4 The MOEA/D 96
4.4.1 MOEA/D Framework 96
4.4.2 MOEA/D Features 97
4.5 HEMH: Hybrid Evolutionary Metaheuristics 98
4.5.1 Motivations 98
4.5.2 The Proposed HEMH 98
4.6 Experimental design 107
4.6.1 Tested Algorithms and instances 107
4.6.2 Parameters settings 109
4.6.3 Performance Assessment metrics 111
4.7 Experimental Results 112
4.8 Summary 116

5 Hybrid Metaheuristics based on MOEA/D 119
5.1 Introduction 119

5.2 MOEA/D Framework 121
5.3 Adaptive Discrete Differential Evolution 121
5.4 Greedy Path-Relinking 122
5.5 Proposed Hybridization Variants 125
5.6 Experimental Design 126
5.6.1 Parameter settings 126
5.6.2 Assessment Metrics 129
5.7 Experimental Results 129
5.8 Concluding Remarks 134
5.9 Summary 135

6 An Improved Hybrid Evolutionary Metaheuristics 137
6.1 Introduction 137
6.2 HEMH, an overview 138
6.3 Adaptive Binary Differential Evolution 139
6.4 Path Relinking 140
6.5 Pareto Adaptive ε-Dominance Archiving 142
6.6 The Proposed HEMH2 143
6.7 Experimental Design 146
6.7.1 Parameter settings 146
6.7.2 Assessment Metrics 148
6.8 Experimental Results 149
6.9 Summary 153

7 A Search Strategy Adaptation based Approach 155
7.1 Introduction 155
7.2 A Review of Search Operators 156
7.2.1 Genetic operators 156
7.2.2 Differential Evolution operator 158
7.2.3 Particle swarm optimization 158
7.2.4 Guided Mutation operator 159
7.3 The proposed HESSA 159

7.3.1 Multiple Search Strategies Adaptation 160
7.3.2 HESSA framework 161
7.3.3 HESSA procedure 162
7.4 Experimental Design 164
7.4.1 Parameter settings 164
7.4.2 Assessment Metrics 167
7.5 Experimental Results 169
7.6 Summary 175

8 Conclusions and Perspectives 183
8.1 Conclusions 183
8.2 Perspectives 185

A Résumé Général de la Thèse 187
A.1 Introduction 187
A.2 Exposé du problème 189
A.3 Historique des Métaheuristiques 190
A.4 Contribution de la thèse 194
A.5 HEMH: Métaheuristiques évolutionnaire hybrides 196
A.5.1 Contexte du HEMH 196
A.5.2 L'algorithme HEMH 197
A.5.3 Résultats du HEMH 198
A.6 Métaheuristiques hybride basées sur le MOEA/D 198
A.6.1 Algorithmes pour les variantes hybrides 199
A.6.2 Les résultats des variantes hybrides 200
A.7 HEMH2: Un HEMH amélioré 201
A.7.1 L'algorithme HEMH2 202
A.7.2 Résultats du HEMH2 203
A.8 HESSA: Une stratégie de recherche basée sur l'adaptation 203
A.8.1 Adaptation de la stratégie de recherche 203
A.8.2 La méthode HESSA 204
A.8.3 L'algorithme HESSA 206

A.8.4 Résultats de la méthode HESSA 207
A.9 Conclusions et perspectives 207

B Detailed Statistical Results 209

Bibliography 231

List of Figures

2.1 Search spaces in multiobjective optimization problems 10
2.2 Convex and Non-convex sets 11
2.3 Illustration of Pareto optimality concept 13
2.4 Strict and weak Pareto optimality 13
2.5 Bounds of Pareto optimal set 15
2.6 Pareto Epsilon optimality 16
2.7 Classification of MOP solution methods 17
2.8 Mechanism of Priori methods 18
2.9 Mechanism of Posteriori methods 19
2.10 Illustration of hypervolume and referenced hypervolume 21
2.11 Global criterion method 24
2.12 Weighted sum approach 26
2.13 Epsilon constrained method 27
2.14 Illustration of CHIM on two-objective problem 29
2.15 Illustration of BI approach (left) and PBI approach (right) 30
2.16 Illustration of Interactive approach for MOP 31
2.17 Illustration of a membership function 33

3.1 The taxonomy of global optimization algorithms [Weise 2009] . . 38
3.2 The Different classifications of metaheuristics algorithms 42
3.3 Local optimum and global optimum in a search space. 45
3.4 Local Search improvement Strategies 47
3.5 Local Search and its alternatives 53
3.6 Roulette Wheel Selection vs. stochastic universal sampling 60
3.7 One and two points crossover for binary and permutation strings . 61
3.8 Mutation on binary and real value encoding 61
3.9 Path Relinking strategies 69
3.10 Changing particles' positions in PSO 72
3.11 Some dominance-based ranking methods 76

3.12 Diversity maintaining strategies 77
3.13 Updating archive by ε-dominance vs. $pa\varepsilon$-dominance 78
3.14 NSGAII procedure [Deb *et al.* 2003] 83
3.15 Hybrid metaheuristics classification (design issue) [Talbi 2002] . . 86
3.16 A summarized classification of hybrid metaheuristics [Raidl 2006] 88

4.1 The MOEAD structure 97
4.2 the proposed HEMH flow diagram 100
4.3 the proposed HEMH flowchart 109
4.4 Results of the coverage indicator I_C 111
4.5 Results of the Hypervolume indicator I_{Hyp} 112
4.6 Results of the Generational Distance indicator I_{GD} 113
4.7 Average results of the Inverted Generational Distance I_{IGD} 114
4.8 Results of the Maximum Spread indicator I_{MS} 115
4.9 The obtained Pareto approximation set for KSP252 Instance . . . 116
4.10 The obtained Pareto approximation set for KSP502 Instance . . . 116
4.11 The obtained Pareto approximation set for KSP752 Instance . . . 117

5.1 The proposed hybridization variants 125
5.2 Results of I_C indicator 130
5.3 Average results of the referenced hypervolume indicator I_{Rhyp} . . 131
5.4 Average results of the generational distance indicator I_{GD} 132
5.5 Average results of the inverted generational distance I_{IGD} 133
5.6 Average results of the R_3 indicator I_{R_3} 134

6.1 The adaptive binary differential evolution 141
6.2 The Pareto ε-dominance Archiving 143
6.3 The framework of the proposed HEMH2 145
6.4 Results of the coverage I_C indicator 149
6.5 Average results of the referenced hypervolume I_{Rhyp} 150
6.6 Average results of the generational distance (I_{GD}) 151
6.7 Average results of the Inverted. Generational Distances (I_{IGD}) . . 152
6.8 Average results of the R_3 indicator (I_{R_3}) 153

7.1 Illustration of Guided Mutation 160
7.2 The structure of the candidate pool. 161
7.3 HESSA Algorithm structure 163
7.4 Results of the coverage I_C indicator 169
7.5 Search Strategy Adaptation for bi-objectives test problems 172
7.6 Search Strategy Adaptation for three-objectives test problems . . 173
7.7 The Pareto fronts achieved for Fonseca test problems 175
7.8 The Pareto fronts achieved for Kursawe test problems 176
7.9 The Pareto fronts achieved for ZDT1 test problems 176
7.10 The Pareto fronts achieved for ZDT2 test problems 177
7.11 The Pareto fronts achieved for ZDT3 test problems 177
7.12 The Pareto fronts achieved for ZDT4 test problems 178
7.13 The Pareto fronts achieved for ZDT6 test problems 178
7.14 The Pareto fronts achieved for DTLZ2 test problems 179
7.15 The Pareto fronts achieved for DTLZ4 test problems 179
7.16 The Pareto fronts achieved for DTLZ6 test problems 180
7.17 The Pareto fronts achieved for DTLZ7 test problems 180

List of Tables

4.1 The set of used knapsack test instances 110
4.2 Set of common parameters used 110
4.3 Results of the Hypervolume indicator (I_{Hyp}) 112
4.4 Results of the Generational distance indicator (I_{GD}) 113
4.5 Average results of the Inverted generational distance (I_{IGD}) . . . 114
4.6 Results of the Maximum Spread indicator (I_{MS}) 115

5.1 Set of knapsack test instances 127
5.2 Set of common parameter used 129
5.3 Average results of the referenced Hypervolume (I_{Rhyp}) 131
5.4 Average results of the generational distance indicator(I_{GD}) 132
5.5 average results of the inverted generational distance(I_{IGD}) 133
5.6 Average results of R_3 indicator (I_{R_3}) 134

6.1 Set of knapsack instances 148
6.2 Set of common parameters 148
6.3 Average results of the referenced hypervolume (I_{Rhyp}) 150
6.4 Average results of the generational distance (I_{GD}) 151
6.5 Average results of the Inverted. Generational Distances (I_{IGD}) . . 152
6.6 Average results of the R_3 indicator (I_{R_3}) 153

7.1 Set of reproduction strategies used 161
7.2 List of the bi-objective test problems 167
7.3 List of three-Objective Test Problems 168
7.4 Set of common parameters 168
7.5 Results of I_{Rhyp} indicator (Average,σ) 171
7.6 Results of I_{GD} indicator (Average,σ) 171
7.7 Results of I_{IGD} indicator (Average,σ) 171
7.8 Results of I_{ε_+} indicator (Average,σ) 171

A.1 Ensemble des stratégies de reproduction utilisées 205

B.1 Detailed results of the Coverage I_c indicator (Median)(Ch. 4) . . . 210
B.2 Detailed results of the hypervolume indicator I_{Hyp} (Ch. 4) 211
B.3 Detailed results of the Generational distance indicator I_{GD} (Ch. 4) 212
B.4 Detailed results of the inverted Generational distance I_{IGD} (Ch. 4) 213
B.5 Detailed results of the Maximum Spread indicator I_{MS} (Ch. 4) . . 214
B.6 Detailed results of the Coverage I_c indicator (Median) (Ch. 5) . . 215
B.7 Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 5) 216
B.8 Detailed results of the Generational distance indicator I_{GD} (Ch. 5) 217
B.9 Detailed results of the inverted generational distance I_{IGD} (Ch. 5) 218
B.10 Detailed results of the $R3$ indicator I_{R3} (Ch. 5) 219
B.11 Detailed results of the Coverage I_c indicator (Median) (Ch. 6) . . 220
B.12 Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 6) 221
B.13 Detailed results of the Generational distance indicator I_{GD} (Ch. 6) 222
B.14 Detailed results of Inverted Generational distance I_{IGD} (Ch. 6) . . 223
B.15 Detailed results of R3 indicator I_{R3} (Ch. 6) 224
B.16 Detailed results of the Coverage I_C indicator (Median) (Ch. 7) . . 225
B.17 Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 7) 226
B.18 Detailed results of the generational distance indicator I_{GD} (Ch. 7) 227
B.19 Detailed results of inverted generational distance I_{IGD} (Ch. 7) . . 228
B.20 Detailed results of unary additive Epsilon indicator I_{ε_+} (Ch. 7) . . 229

List of Algorithms

3.1 Greedy Heuristics 44
3.2 Local Search (LS) 44
3.3 GRASP(α) 47
3.4 Construction(s, α) 48
3.5 LocalSearchPhase(s) 49
3.6 Iterated Local Search (ILS) 50
3.7 Variable Neighborhood Search (VNS) 51
3.8 Guided Local Search (GLS) 53
3.9 Simulated Annealing (SA) 55
3.10 Tabu Search (TS) 56
3.11 Evolutionary computation (EC) 58
3.12 Genetic Algorithm (GA) 62
3.13 Evolutionary Strategies (ES) 63
3.14 Differential Evolution (DE) 66
3.15 Scatter Search (SS) 67
3.16 Path Relinking (PR) 69
3.17 Particle Swarm Optimization(PSO) 72
3.18 Ant colony optimization(sol) 75
3.19 SPEA2($N, ArchSize$) 82
3.20 NSGA-II(N) 84
4.1 DM-GRASP(n) 95
4.2 Construction($p, \alpha, \Lambda, Arch$) 101
4.3 LocalSearch($x, \beta, \Lambda, Arch$) 102
4.4 PatternMining($\sigma, \mathcal{E}$) 103
4.5 GreedyRandomPathRelink($x^s, x^t, \alpha, \Lambda, Arch$) 104
4.6 GreedyRepair(x, Λ) 106
4.7 UpdateSolutions(y, t, M) 106
4.8 HEMH($N, T, t, \delta, \alpha, \beta, \sigma, \varepsilon$) 108
5.1 MOEAD(N, T, t, δ) 121
5.2 ABDifferentialEvolution($x, y_1, y_2, y_3, F_0, CR_0, a_1, a_2$) 123
5.3 GreedyPathRelinking(x^s, x^t, Λ) 124
5.4 MOEAD$_{pr}$($N, T, t, \delta, \varepsilon, \gamma$) 127
5.5 MOEAD$_{de}$($N, T, t, \delta, F_0, CR_0, a_1, a_2$) 128
5.6 MOEAD$_{dp1}$($N, T, t, \delta, \varepsilon, \gamma, F_0, CR_0, a_1, a_2$) 128
5.7 MOEAD$_{dp2}$($N, T, t, \delta, \varepsilon, \gamma, F_0, CR_0, a_1, a_2$) 128
6.1 AdaptiveBinaryDiffEvol($x, x^a, x^b, x^c, CR_0, a$) 141
6.2 UpdateArchive$^{pa\varepsilon}$($y, Archive$) 143

6.3 InverseGreedy(x, Λ) 144
6.4 HEMH2$(N, T, t, \varepsilon, \gamma, CR_0, a)$ 147
7.1 HESSA$(N, T, t, LP, \delta, p_c, p_m, \eta_c, \eta_m, CR, F, T_a)$ 165
7.2 Reproduction$(S_k, x^i, x^a, x^b, x^c, v^i, x^i_{pb}, x^i_{gb}, a_i)$ 166
7.3 UpdateSolutions(y, t, M_i, P, S_k, r^*) 166

List of Acronyms

ACO	Ant Colony Optimization
DE	Differential Evolution
DM	Decision Maker
EC	Evolutionary Computation
ES	Evolutionary Strategies
GA	Genetic Algorithms
GLS	Guided Local Search
GRASP	Greedy Randomized Adaptive Search Procedure
HEMH	Hybrid Evolutionary Metaheuristics
HEMH2	Improved Hybrid Evolutionary Metaheuristics
HESSA	Hybrid Evolutionary Approach with Search Strategy Adaptation
LS, ILS	Local Search, Iterated Local search
MA	Memetic Algorithms
MADM	Multi-Attribute Decision Making
MCDM	Multiple Criteria Decision Making
MOCO	Multiobjective Combinatorial Optimization
MODM	Multiobjective Decision Making
MOEA/D	Multiobjective Evolutionary Algorithm based on Decomposition
MOGA	Multiobjective Genetic Algorithm
MOIP	Multiobjective Combinatorial Optimization
MOKP	Multiobjective Knapsack Problems
MOP	Multiobjective Optimization Problem
NPGA	Niched Pareto Genetic Algorithm
NSGA	Nondominated Sorting Genetic Algorithm
NSGA-II	Elitist Nondominated Sorting Genetic Algorithm II
PAES	Pareto Archived Evolution Strategy
PESA	Pareto Envelope-based Selection Algorithm
PR	Path Relinking
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SI	Swarm Intelligence
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2
SS	Scatter Search
TS	Tabu Search
VEGA	Vector Evaluated Genetic Algorithm
VNS	Variable Neighborhood Search

Nomenclatures

$\Delta(x, y)$ Hamming distance between x and y 110
η_c, η_m distribution indexes for SBX-Crossover and Polynomial mutation . . . 163
$\mathbb{N}$ The set of natural numbers 12
$\mathbb{R}^n$ n-dimensional Euclidean space 11
$\mathcal{N}(s)$ Neighborhood of solution s 46
$\mathcal{PF}^*$ Pareto front 14
$\mathcal{PF}^*_\varepsilon$ Pareto epsilon front 17
$\mathcal{P}^*$ Pareto-optimal set, Efficient set 14
$\mathcal{P}^*_\varepsilon$ Pareto epsilon optimal set 17
$\mathcal{S}$ Feasible decision variable space 11
$\mathcal{Z}$ Feasible (*attainable*) objective space 11
CR, F Crossover rate, Scaling factor for diffrential evolution 128
$E \prec\succ F$ E and F are incomparable 14
F objective vector 191
$F(x)$ Objective vector 11
$f_i(x)$ The ith Objective function 11
$g_j(x)$ The jth constraint 11
I_C The set coverage indicator 22
$I_{\varepsilon+}$ The unary additive epsilon indicator 25
I_{GD} The generational distance indicator 24
I_{Hyp} Hypervolume indicator 23
I_{IGD} The Inverted generational distance indicator 24
I_{MS} The Maximum Spread indicator 25
I_{R_3} The R3-indicator 24
I_{Rhyp} Referenced hypervolume indicator 23
k Number of constrains 11
m Number of objective functions 11
n Number of decision variables 11
p_c, p_m Crossover probability, Mutation probability 163
$u \preccurlyeq v$ u dominates v 13
$u \preccurlyeq_\varepsilon v$ u Epsilon dominates v 16
x decision vector 11
z^* ideal objective vector 16
z^{**} Utopian objective vector 16
z^{nad} nadir objective vector 16

Chapter 1

Introduction

Contents

1.1 Overview 1
1.2 Problem Statement 2
1.3 Research Objectives 3
1.4 Thesis Methodology 4
1.5 Thesis Contribution 4
1.6 Thesis Organization 6

In this introductory chapter, an overview of the thesis context is provided in section 1.1. Section 1.2 describes the problem statement. The research objectives and the methodology are presented in sections 1.3 and 1.4 respectively. Moreover, the principle contributions are highlighted in section 1.5. Finally, the organization of the dissertation is presented in section 1.6.

1.1 Overview

Industries, companies, decision makers, people and others face many problems everyday. Most of those problems have many trade-offs as well as many conflicting objectives with often higher complex search space. It is natural to want everything to be as good as possible, in other words, optimal. It is too hard for a decision maker to take a decision in such problems. Regarding that many problems in our life can be formulated as optimization problems, optimization became one of the most important research fields. Optimization techniques especially metaheuristics represent a promising area of research as they achieved very attractive results with real life problems. Single and multiple objective optimization problems can be used to formulate any problem to help the decision maker to take an optimal or a near optimal decision. Much advancement has been achieved in this domain since it was initiated. Many optimization methods have been proposed such as classical (Exact Methods or traditional methods such as Branch-and-Bound, dynamic and mathematical programing approaches etc.),

Intelligent based methods (Evolutionary Computations), heuristic, metaheuristics etc. All of those optimization techniques differ from each in terms of the quality of the obtained solution and the time required to get these solutions. In the last decades, hybridization among different optimization techniques has more increased interest, especially in handling multiobjective optimization. Hybridization aims to incorporate and combine different search techniques with each other to enhance the search capabilities. It can improve both of intensification and diversification toward the preferred solutions and concentrates the search efforts to investigate the promising regions in the search space. In this thesis, we are concerned with using hybrid metaheuristics based techniques to solve multiobjective optimization problems in both discrete and continuous search domains.

1.2 Problem Statement

Many of real world optimization problems can be modeled as Multiobjective Optimization Problems (MOP), which are often characterized by their large size and the presence of multiple, conflicting objectives. The difficulty of a problem arises when there are conflict between different goals and objectives. On the one hand, instead of a single optimal solution, competing goals give rise to a set of Pareto-optimal solutions. On the other hand, the search space can be too large and too complex to be solved by exact methods. In general, the basic task in multiobjective optimization is the identification of the set of Pareto optimal solutions or even a good approximation set to the true Pareto front (PF). Any solution of these set is optimal in the sense that no improvement can be made on a component of the objective vector without degradation of at least another of its components. Based on the nature of the search space, the general multiobjective optimization problems involves two basic classes. The first class is called continuous multiobjective problems in which the solutions are encoded in real-valued variables. The second class is the discrete multiobjective problems that uses discrete decision variables. In the second class, we find the multiobjective combinatorial optimization (MOCO) problems.

Many real world optimization problems with multiple conflicting objectives exist in divers fields as science, commerce, economy, finance and industry which are of great complexity. Thus, efficient optimization strategies are required that are able to deal with these difficulties. Over the past decades, lots of new ideas have been investigated and studied to solve multiobjective optimization problems. Classical methods for generating the Pareto optimal solutions aggregate the objective functions into a single, parametrized objective function by analogy to decision making before search. These methods suffer from some limitations such as: generating one efficient solution at a time, the requirement to prior knowledge and the requirement to time exponentially proportion with problem

size. The researchers found the metaheuristic based techniques are a promising tool that can treat the limitations of the conventional methods. These techniques have the ability to find a set of optimal solutions in each simulation run by using population-based stochastic search and optimization methods like Evolutionary Computations (EC).

Over the last years, a large number of search algorithms were reported that do not purely follow the concepts of one single classical metaheuristic, but they attempt to obtain the best from a set of different metaheuristics (and even other kinds of optimization methods) that perform together, complement each other and augment their exploration capabilities to produce a profitable synergy from their combination. These approaches are commonly referred to as hybrid metaheuristics [Raidl 2006]. Intensification and diversification are the two major issues when designing a global search method [Blum & Roli 2003]. Diversification generally refers to the ability to visit many and different regions in the search space whereas, intensification refers to the ability to obtain high quality solutions within those regions. A search algorithm must balance between sometimes-conflicting two goals. It is a complicated task for metaheuristics to provide adequate balance between intensification and diversification, but the hybrid metaheuristics can give the ability to control the balance between intensification and diversification through involving the design of hybrid metaheuristics with search algorithms specializing in intensification and diversification, which combines these types of algorithms with the objective of compensating each other and put together their complementary behaviors [Lozano & García-Martínez 2010].

1.3 Research Objectives

Decision Making plays an important role to solve our daily life problems, the problem becomes more difficult to handle in case of multiple objectives are involved with highly complex search space. In this case, the decision makers need powerful techniques and effective tools to help them to handle these decision tasks. Recently, hybrid metaheuristics achieve successful results in handling a wide range of optimization problems with single or multiple objectives. They have received increased interest due to their simplicity and their ability to provide the efficient solutions for hard optimization tasks in a most reasonable execution time. In this work, the main target is to study and develop hybrid metaheuristic based approaches for handling real life multiobjective optimization problems. The developed approaches should have the ability to maintain good balance between intensification and diversification to obtain a high quality approximation to the PF. This can be achieved by enhancing the current approaches through developing new hybrid schemes that have the ability to concentrate the search efforts for discovering the most promising regions in the search space. Also, the

developed hybrid schemes should have the ability to combine different cooperative search techniques in an integrated framework in order to augment their exploration capabilities and to maximize their cooperation with each other. The methodology described in the next section is used to achieve these objectives.

1.4 Thesis Methodology

In this thesis, we study how to improve the efficiency and to enhance the performance of metaheuristics through developing hybridization among different types of metaheuristics or other techniques. In order to accomplish this task with obtaining consistence results, a library which contains different metaheuristics methods and tools has been built. The library is constructed from the algorithms proposed in this thesis. Besides, collecting and revising different metaheuristics techniques that proposed in the literature especially those techniques that used in comparative studies and experimental designs provided in the thesis. The proposed library was implemented using c++ according to the principles of object-oriented programming. The set of algorithms collected from literature include NSGA-II [Deb *et al.* 2000], SPEA2 [Zitzler *et al.* 2001], MOEA/D [Zhang & Li 2007], GRASPM [Vianna & Arroyo 2004], dMOPSO [Martínez & Coello 2011] etc. The library also includes some of the search operators and tools used in the algorithms that proposed in this study such as local search, greedy randomized path-relinking , adaptive binary differential evolution, pattern mining and DM-GRASP. The developed library also contains the set of test problems that used in the studies provided in the thesis. In order to assess the proposed algorithms against the state of the art algorithms, a set of assessment indicators are also involved in the proposed library including the set coverage indicator [Zitzler & Thiele 1999], hypervolume [Zitzler & Thiele 1999], generational and inverted generational [Van Veldhuizen & Lamont 2000], maximum spread [Zitzler *et al.* 2000] and the unary additive epsilon [Zitzler *et al.* 2003]. Concerning the preservation of the efficient solutions, the proposed library involves some of different archiving strategies such as crowding distances [Deb *et al.* 2000], epsilon dominance, adaptive grid [Knowles & Corne 2000], and Pareto-adaptive epsilon dominance [Hernández-Díaz *et al.* 2007]. Finally, the proposed Library is used to carry out the comparative studies and also to implement our proposals such as HEMH, HEMH2, HESSA and their variants which are provided in this thesis.

1.5 Thesis Contribution

The main contributions of this thesis are summarized in the following points:

- A hybrid evolutionary metaheuristics HEMH based on DM-GRASP and greedy randomized path-relinking is proposed. The experimental results indicate the superiority of the decomposition based MOEAs over the Pareto dominance based MOEAs. the proposed HEMH is able to intensify the search process for discovering the most promising regions in the search space through the combination among different metaheuristics techniques that integrate each others. HEMH algorithm has the ability to find a good approximation set of high quality solutions using a small set of uniformly distributed search directions due to the use of path-relinking and local search strategies.
- Other four different hybridization variants within MOEA/D framework are presented, MOEAD_{de}, MOEAD_{pr}, MOEAD_{dp1} and MOEAD_{dp2}. The experimental results indicate the superiority of all proposed hybrid variants over the original MOEA/D and SPEA2 for most test instances. For bi-objective MOKP test instances, path-relinking operator has the best performance followed by differential evolution and standard crossover and mutation. Where in MOKP test instances with three or four objectives, differential evolution has the superiority, followed by path-relinking and standard crossover and mutation.
- An improved hybrid evolutionary metaheuristics HEMH2 and two other variants called HEMH_{de} and HEMH_{pr} are proposed to enhance HEMH performance. The experimental results indicate the superiority of all proposals HEMH2 and HEMH_{de} over the original MOEA/D and HEMH. It is clear that, the adaptive binary differential evolution included in both HEMH2 and HEMH_{de} has better exploration capabilities which overcome the local search capabilities contained in the original HEMH. Therefore both of HEMH2 and HEMH_{de} outperform HEMH. The adaptive binary differential evolution can achieve better performance than path-relinking. So, in some cases, HEMH_{de} can achieve highly competitive results compared with HEMH2.
- A hybrid evolutionary approach with search strategy adaptation HESSA for handling multiobjective continuous problems is proposed. In HESSA, the search process is controlled by adapting the search strategies used during the evolution process. The experimental results indicate the superiority of HESSA over both MOEA/D and dMOPSO on the most of test problems used. HESSA combines among different cooperative search operators that intensify the search process to discover the promising regions in the search space and enhance the ability to explore high quality solutions. HESSA is able to adapt the search process by adopting the suitable search operator to the problem on hand.

1.6 Thesis Organization

This thesis is organized as follows. Chapter 2 reviews the basic concepts of multiobjective decision making problems, with concentration on the traditional methods used in handling these problems, especially their classification and their drawbacks that force researchers to direct their attention to metaheuristics based techniques for handling these types of problems. The nature of metaheuristics based multiobjective is provided in Chapter 3 in more details, with focusing the lights on some of the commonly used metaheuristic approaches such as, Path-Relinking, GRASP, Local Search, Differential Evolution, Tabu Search, Simulated annealing, ACO, Particle Swarm, etc. As, we are concerned with enhancing the multiobjective search techniques through developing hybrid metaheuristic approaches. Chapter 4 provides a hybrid evolutionary metaheuristic approach (HEMH) based on combination between DM-GRASP and greedy randomized path-relinking with local search or reproduction operators to handle MOKP, with testing its efficiency using different test instances and quality assessment metrics. A comparative study among different four proposals that combine both adaptive discrete differential evolution operator and/or greedy path-Relinking within MOEA/D is presented in chapter 5. The four proposals tend to determine the best combination of search operators that can be used to improve the results of HEMH. Based on the results achieved in chapter 5 , chapter 6 presents a modified version of the HEME called HEMH2. Motivated by the results achieved in both HEMH and HEMH2 for discrete search space. Chapter 7 introduces a Hybrid Evolutionary Approach with Search Strategy Adaptation (HESSA) for multiobjective continuous optimization. Finally, the conclusions and perspectives for the whole thesis are listed in chapter 8.

Chapter 2

Multiobjective Decision making

Contents

2.1 Introduction **7**

2.2 Single objective vs. multiple objectives **8**

2.3 Multiobjective Optimization Problem **9**

2.3.1 Pareto optimality 10

2.3.2 Weakly and Strictly Efficient Solutions 12

2.3.3 Bounds on the Pareto-optimal set 14

2.3.4 Pareto Epsilon optimality 14

2.4 Multiobjective Knapsack Problem **15**

2.4.1 Problem Statement 15

2.5 Classification of multiobjective methods **16**

2.6 Quality Assessment **19**

2.6.1 Concepts and definitions 20

2.6.2 Assessment indicators 20

2.7 Classical Methods for MODM **23**

2.7.1 Scalarization Techniques 23

2.7.2 Interactive methods 30

2.7.3 Goal programming methods 31

2.7.4 Fuzzy programming methods 32

2.8 Drawbacks of classical Methods **33**

2.1 Introduction

Decision making is an integral part of our daily life. It ranges in scope from the individual to the largest groups and societies, including nations and, ultimately, organization at the global level. It considers situations ranging in complexity from the simple to the most complex involving multiobjective, so with multiple

objectives, confliction will appear. Multiple criteria decision making (MCDM) [Zeleny 1982, Gál *et al.* 1999, Triantaphyllou 2000] deals with decision situations where the decision maker has several usually conflicting objectives. In typical real-life problems, there is no ideal alternative can be considered as the optimal for each objective. Hence, the most important task in multiple criteria decision making is to find a good compromise solution which performs the best alternative from the decision maker's point of view, taking into account all objectives simultaneously. Consequently, the quality of the compromise will depend on the decision maker's preferences. According to many authors [Zimmermann 1991] MCDM is divided into Multiobjective Decision Making (MODM) and Multi-Attribute Decision Making (MADM). Practically, MODM studies decision problems in which the decision space is continuous as in mathematical programming problems with multiple objective functions. On the other hand, MADM concentrates on problems with discrete decision spaces. In these problems the set of decision alternatives has been predetermined. In general, the Decision Maker (DM) is primarily concerned to find "the most" promising alternative with respect to limited resources. In this dissertation, we deal with multiobjective optimization problems weather with continuous or discrete search space.

2.2 Single objective vs. multiple objectives

Many real world decision making problems involve simultaneous optimization of multiple objectives. In principle, multiobjective optimization is very different than the single-objective optimization. In single-objective optimization one attempts to obtain the best decision which is usually the *global minimum* or the *global maximum* depending on the nature of the optimization problem. On the contrary, in a multiobjective optimization with usually conflicting objectives, there is no single optimal solution with respect to all objectives. The interaction among different objectives gives rise to a set of compromised solutions, largely known as the *efficient*, *trade-off*, *nondominated*, *noninferior* or *Pareto-optimal* solutions[Chankong & Haimes 1983]. This set contains the solutions which are superior to the rest of solutions in the search space when all objectives are considered. The choice of one solution over the others requires knowledge and preference information from the decision maker (DM). Thus, both of the search and decision making are required. Multiple objectives encountered in real-life problems can often be expressed as mathematical functions of a variety of forms, i.e., not only do we deal with conflicting objectives, but with objectives of different structures. Multiobjective optimization has its roots in the 19th century in a work in economy of Edgeworth and Pareto [Edgeworth 1961, Pareto 1964]. It was initially applied to economic sciences and management, and gradually to engineering sciences.

In the following, the formulation of the multiobjective optimization problem is presented as well as some of the basic related concepts and definitions.

2.3 Multiobjective Optimization Problem

Without loss of generality, the general multiobjective optimization problem (MOP) is formulated as follows[1]:

$$\begin{array}{ll} Min: & F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \\ s.t.: & g_j(x) \leq 0, \forall j = 1, 2, ..., k \\ & x = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n \end{array} \tag{2.1}$$

where $F(x)$ is the m-dimensional objective vector, $f_i(x)$ is the ith objective function to be minimized, $\forall i = 1, ..., m$, $x = (x_1, x_2, ..., x_n)^T$ is the n-dimensional decision vector, $g_j(x)$ is the jth constraints, $\forall j = 1, 2, ..., k$. A feasible solution $x \in \mathbb{R}^n$ must respect the k constraints. Therefore, the feasible region in the decision variables space is given by the set $\mathcal{S} = \{x \in \mathbb{R}^n : g_j(x) \leq 0, \forall j = 1, 2, ..., k\}$. Consequently, the feasible region in the objective (*criterion*) space, that is the evaluation of the feasible set $\mathcal{S}$ (also called *the attainable objective set*), is given by $\mathcal{Z} = F(S) = \{F(x) : x \in \mathcal{S}\}$.

According to the formulation above, the MOP consists of $m \geq 2$ objective functions, n decision variables and $k \geq 0$ constraints. The objective functions and constrains can be linear [Steuer 1986] or non-linear [Miettinen 1999] and the variables are contentious. Therefore, an infinite number of solutions are feasible. The MOP's evaluation function, $F : \mathcal{S} \rightarrow \mathcal{Z}$ maps the decision vector $x = (x_1, x_2, ..., x_n)^T$ to its corresponding vector $y = (y_1, y_2, ..., y_m)^T$ in the objective space as represented in Fig. 2.1 for the case $n = 2$ and $m = 3$.

As well-known, the modeling of many of real-life applications involves discrete decision variables. Considering the MOP formulation in (2.1), if the decision variables are discrete, i.e. $x \in \mathbb{N}^n$, where $\mathbb{N}$ is the set of natural numbers, the MOP is called "*Multiobjective Integer Problem*" (MOIP). the decision space is defined as $S = \{x \in \mathbb{N}^n : g_j(x) \leq 0, \forall j = 1, 2, ..., k\}$. The Multiobjective combinatorial optimization (MOCO) problem is a MOIP with binary decision variables, i.e. $x \in 0, 1$. The decision space in this case is defined as $S = \{x \in 0, 1 : g_j(x) \leq b, \forall j = 1, 2, ..., K\}$. The objective vector $F(x)$ and the set of constrains $g(x)$ defines the structure of the MOCO problem. In this dissertation, we will deal with the general MOP as well as MOCO represented by the multiobjective knapsack problems (MOKP).The general concepts and definitions related to MOP and MOCO will be discussed in the next sections. According

[1]The minimization case is considered in our discussion

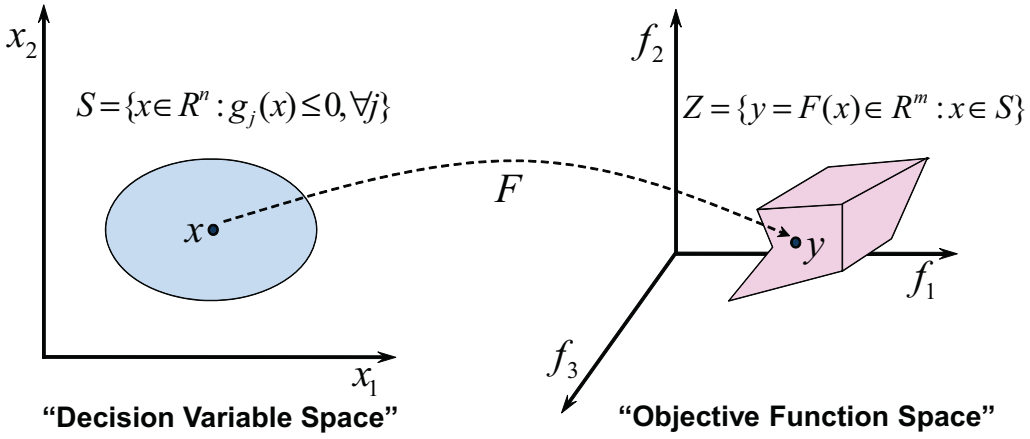


Figure 2.1: Search spaces in multiobjective optimization problems

to the nature of the objective functions and the constraints forming the feasible region, different types of MOP can be defined as follows:

Definition 2.1 (Linear MOP) [Miettinen 1999]: *The MOP is called linear (MOLP) if all objective functions and all constraints are linear. if at least one objective function or constraint is nonlinear, then MOP will be nonlinear (MONLP)*

Definition 2.2 (Convex function) [Miettinen 1999]: *A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is convex if for all* x_1, $x_2 \in \mathbb{R}^n$, *is true that* $f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$ *for all* $0 \leq \alpha \leq 1$.

Definition 2.3 (Convex set) [Miettinen 1999]: *A set* $\mathcal{S} \subset \mathbb{R}^n$ *is convex if* x_1, $x_2 \in \mathcal{S}$ *implies that* $\alpha x_1 + (1-\alpha)x_2) \in \mathcal{S}$ *for all* $0 \leq \alpha \leq 1$. *see Fig. 2.2*

Definition 2.4 (Convex MOP) [Miettinen 1999]: *A multiobjective optimization problem (MOP) is called convex if all objective functions and the feasible region are convex.*

As mentioned before, the basic task in MOP is to identify the set of efficient "trade-offs" solutions. The concept of Pareto-optimality must be used in determining a set of MOP solutions. This concept will be defined precisely in details in the following section.

2.3.1 Pareto optimality

It is rarely the case that there is a single point that simultaneously optimizes all the objective functions of a multiobjective optimization problem. Therefore,

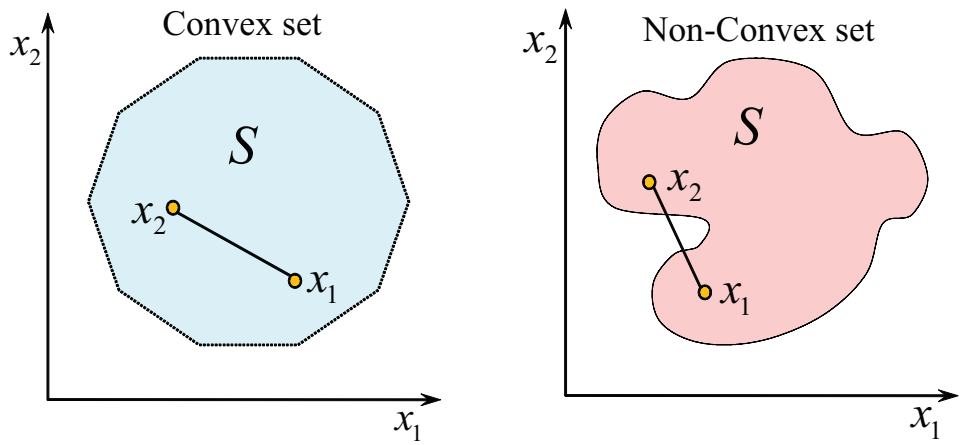


Figure 2.2: Convex and Non-convex sets

we normally look for "*trade-offs*", rather than single solutions when dealing with multiobjective optimization problems. In this case, the notions of optimality are different. These notions are presented through the following definitions.

Definition 2.5 (Complete optimal solution) [Sakawa 1993]: *A solution x^* is said to be a complete optimal solution of MOP, if and only if there exists $x^* \in \mathcal{S}$ such that $f_j(x^*) \leq f_j(x), \forall j = 1, 2, ..., m, \forall x \in \mathcal{S}$.*

In general, such a complete optimal solution that simultaneously minimizes all of the multiple objective functions does not always exist when the objective functions conflict with each other. Thus, the Pareto optimal concept is defined as follows:

Definition 2.6 (Pareto-optimal solution) [Miettinen 1999]: *A decision vector $x^* \in \mathcal{S}$ corresponding the objective vector $F(x^*)$ is said to be a Pareto optimal ("efficient") solution of MOP, if there does not exist another decision vector $x \in \mathcal{S}$ which is at least as good as x^* with respect to all objectives and strictly better for at least one objective i.e. if there is no $x \in \mathcal{S}$ such that $f_i(x) \leq f_i(x^*)$, $\forall i = 1, 2, ..., m$ and $f_j(x) < f_j(x*)$ for at least one index j.*

Definition 2.7 (Pareto dominance) [Engelbrecht 2007]: *An objective vector $u = (u_1, u_2, ..., u_m)$ is said to dominate the objective vector $v = (v_1, v_2, ..., v_m)$ (denoted by "$u \preccurlyeq v$") if and only if u is partially less than v, i.e. $\forall i \in \{1, 2, ..., m\}$: $u_i \leq v_i$ and $\exists i \in 1, 2, ..., m : u_i < v_i$.*

Practically, **definitions** 2.6 and 2.7 are the same. However, Pareto-optimality ("Efficiency") typically refers to a vector of decision variables in the decision space

$\mathcal{S}$, whereas dominance refers to vectors in the objective space $\mathcal{Z}$. By applying the above concepts to the solutions in the feasible search space $\mathcal{S}$, we can obtain the Pareto optimal (*efficient*) set for MOP in (2.1) according to the definitions below.

Definition 2.8 (Pareto-optimal set) *For a given MOP $F(x)$, the set of efficient solutions (Pareto optimal set "$\mathcal{P}^*$") is defined as:*

$$\mathcal{P}^* = \{x \in \mathcal{S} | \nexists y \in \mathcal{S} : F(y) \preccurlyeq F(x)\}. \tag{2.2}$$

The vectors x^* corresponding to the solutions included in the Pareto optimal set are sometimes called a *nondominated* or a *noninferior* solution since it is not inferior to other feasible solutions. The image of the Pareto optimal set in the objective function space is known as *"Pareto front"* ($\mathcal{PF}^*$) according to the following definition:

Definition 2.9 (Pareto front) *For a given MOP, $F(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front ($\mathcal{PF}^*$) is defined as:*

$$\mathcal{PF}^* = \{F(x) = (f_1(x), ..., f_m(x)) | x \in \mathcal{P}^*\}. \tag{2.3}$$

Fig. 2.3 depicts the above concepts, which is to minimize two conflicting objectives f_1 and f_2. All Pareto optimal solutions lies on the boundary of the attainable region, the Pareto front is represented by the bold line, thus solutions A, B, C, and D are members in the Pareto optimal set, and it can be observed that solution F dominates G ($F \preccurlyeq G$) but at the same time solution B dominates both F and G ($B \preccurlyeq F \& B \preccurlyeq G$). It can be observed also that solutions E and F are incomparable ($E \prec\succ F$).

2.3.2 Weakly and Strictly Efficient Solutions

If we use the weak and strict component wise order, we will obtain the definitions of weakly and strictly Pareto-optimal solution as follows

Definition 2.10 (Weak Pareto optimality) [Coello *et al.* 2007]: *A feasible solution x of MOP is called weakly Pareto optimal, if there does not exist another feasible solution $\acute{x}$ which is strictly better with respect to all objectives i.e. $\nexists \acute{x} \in \mathcal{S} : f_j(\acute{x}) < f_j(x), \forall j = 1, 2, ..., m$.*

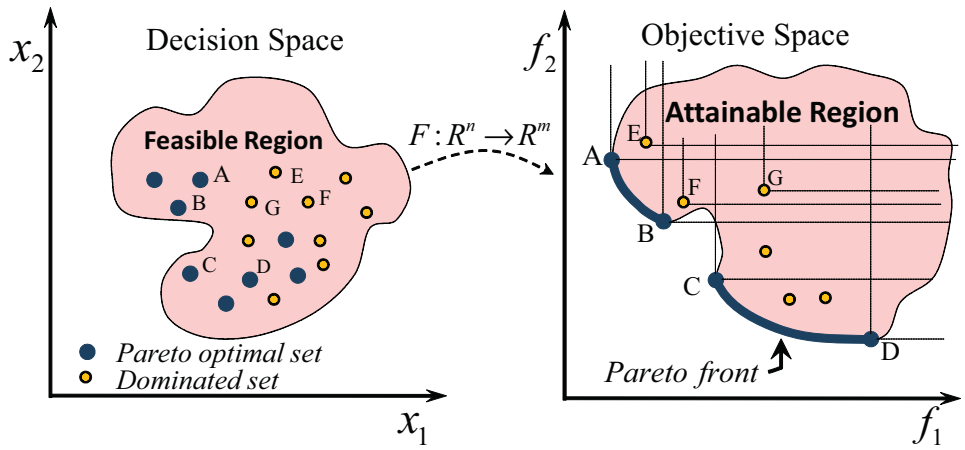


Figure 2.3: Illustration of Pareto optimality concept

Definition 2.11 (Strict Pareto Optimality) [Coello *et al.* 2007]: *A solution x is said to be a strictly Pareto optimal solution of MOP, if there is no $\acute{x} \in \mathcal{S}$, $\acute{x} \neq x$ such that $f_j(\acute{x}) \leq f_j(x)$, for $j = 1, 2, ..., m$.*

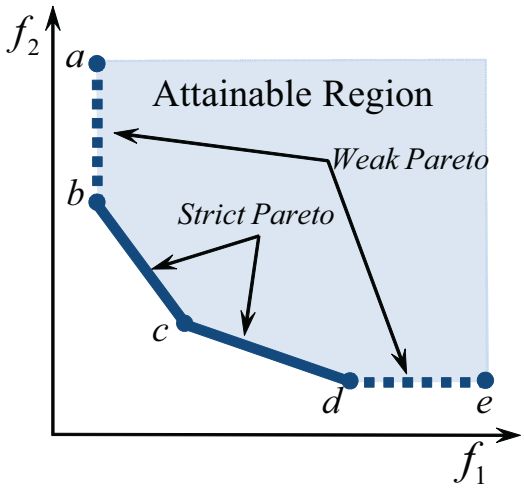


Figure 2.4: Strict and weak Pareto optimality

For convenience, let X^{CO}, X^P and X^{WP} denote complete optimal, Pareto optimal and weak Pareto optimal solution sets, respectively. Then from their definitions, it can be easily understood that the following relation holds:

$$X^{CO} \subseteq X^P \subseteq X^{WP} \tag{2.4}$$

In Fig. 2.4, the dashed line represents the set of weakly Pareto optimal criterion vectors. The Pareto optimal criterion vectors are located at the fat line between

the black points.

2.3.3 Bounds on the Pareto-optimal set

According to the above concepts, we formally define the ideal and nadir points as lower and upper bounds on Pareto-optimal solutions. thes points gives indication of the range of the values which nondominated points can attain. They are often used as reference points for some solutions techniques in order to find a most preferred solutions for a decision maker.

Definition 2.12 (Ideal objective vector) [Miettinen 1999]: *The ideal objective vector* $z^* = (z_1^*, ..., z_m^*)$ *is composed of the best attainable objective function values. Each component* z_j^* *is calculated by minimizing the corresponding objective function* f_j *over the feasible decision space* $\mathcal{S}$*. i.e.*$z_j^* = Min_{x \in \mathcal{S}} f_j(x), \forall j = 1, ..., m$.

Definition 2.13 *A Utopian objective vector* $z^{**} \in \mathbb{R}^m$ *is a vector whose components are slightly smaller than that of the ideal objective vector. i.e.* $z_i^{**} = z_1^* - \varepsilon_i$*, with* $\varepsilon_i > 0$*, for all* $i = 1, ..., m$

Definition 2.14 (Nadir objective vector) [Miettinen 1999]: *The nadir objective vector* $z^{nad} = (z_1^{nad}, ..., z_m^{nad})$ *is constructed using the worst values of objective functions in the complete Pareto-optimal front. i.e.* $z_j^{nad} = Max_x\{f_j(x)|x \in \mathcal{P}^*, \}, \forall j$*, where* $\mathcal{P}^*$ *is the Pareto-optimal set the objective.*

The ideal and Nadir points provide important information about a multiobjective optimization problem (MOP). For a decision maker facing a multiobjective problem, they show the possible range of the objective values of all his criteria over the Pareto set: They are exact lower respectively upper bounds for the set of efficient solutions. Figure (2.5) shows the ideal and nadir and utobian points in the objective function space

2.3.4 Pareto Epsilon optimality

Many of Multiobjective optimization problems may have many or even infinite Pareto optimal solutions. its is very time consuming, if not impossible to obtain the complete $\mathcal{PF}^*$. On the other hand, the decision maker (DM) may not be interested in having a large number of efficient solutions to deal with. Therefore many of solution techniques tends to fined a manageable set of efficient solutions which are a good representative to $\mathcal{PF}^*$. The concept of Pareto epsilon arises to extend the previous definitions to provide a method to control the distances among efficient solutions along $\mathcal{PF}^*$ to reach a manageable set with good representation.

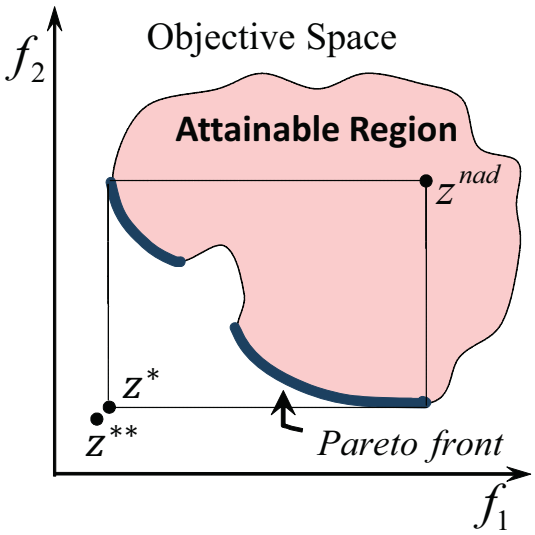


Figure 2.5: Bounds of Pareto optimal set

Definition 2.15 (Pareto ε-dominance) [Coello *et al.* 2007]: *A vector* $u = (u_1, u_2, ..., u_m)$ *is said to epsilon-dominate another vector* $v = (v_1, v_2, ..., v_m)$ *(denoted by* $u \preccurlyeq_\varepsilon v$*) if for some* $\varepsilon > 0$ u_i *is partially less than* $v_i + \varepsilon$*, i.e.* $\forall i \in \{1, 2, ..., m\} : u_i \leq v_i + \varepsilon$ *and* $\exists i \in 1, 2, ..., m : u_i < v_i + \varepsilon$ *where* $\varepsilon > 0$.

Definition 2.16 (Pareto ε-optimality) [Coello *et al.* 2007]: *A solution* $x \in \mathcal{S}$ *corresponding the objective vector* $u = F(x) = (f_1(x), ..., f_m(x)$ *is said to be a Pareto epsilon optimal with respect to* $\mathcal{S}$*, if and only if there is no* $\acute{x} \in \mathcal{S}$ *for which* $v = F(\acute{x}) = (f_1(\acute{x}), ..., f_m(\acute{x}))$ *epsilon dominates* u.

Definition 2.17 (Pareto ε-optimal set) *For a given MOP* $F(x)$*, the Pareto epsilon optimal set* ($\mathcal{P}^*_\varepsilon$) *is defined as:*

$$\mathcal{P}^*_\varepsilon = \{x \in \mathcal{S} | \nexists y \in \mathcal{S} : F(y) \preccurlyeq_\varepsilon F(x)\}. \tag{2.5}$$

Definition 2.18 (Pareto ε-front) *For a given MOP,* $F(x)$ *and Pareto epsilon optimal set* $\mathcal{P}^*_\varepsilon$*, the Pareto epsilon front* ($\mathcal{PF}^*_\varepsilon$) *is defined as:*

$$\mathcal{PF}^*_\varepsilon = \{F(x) = (f_1(x), ..., f_m(x)) | x \in \mathcal{P}^*_\varepsilon\}. \tag{2.6}$$

2.4 Multiobjective Knapsack Problem

2.4.1 Problem Statement

Generally, a 0/1 knapsack problem consists of a set of items, weight and profit associated with each item, and an upper bound for the capacity of the knapsack.

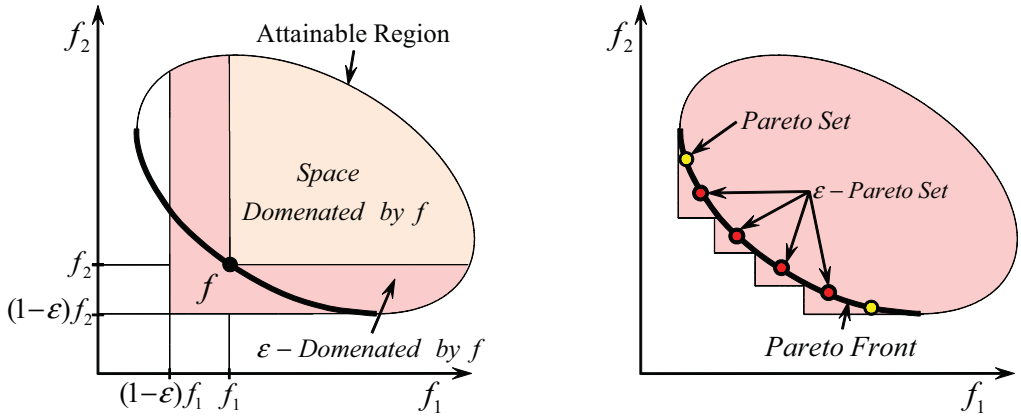


Figure 2.6: Pareto Epsilon optimality

The task is to find a subset of items which maximizes the total of the profits in the subset, yet all selected items fit into the knapsack, i.e., the total weight does not exceed the given capacity [Martello & Toth 1990]. this single-objective problem can be extended directly to an MOP by allowing an arbitrary number of knapsacks. Formally the multiobjective 0/1 knapsack problem considered here is defined in the following way: Given a set of n items and a set of m knapsacks, as follows:

$$\begin{aligned} \underset{x}{\text{Maximize}} \quad & f_i(x) = \sum_{j=1}^{n} c_{ij} x_j, \ \forall i \in \{1, ..., m\} \\ \text{subject to} \quad & \sum_{j=1}^{n} w_{ij} x_j \leq W_i, \ \forall i \in \{1, ..., m\} \\ & x = (x_1, ..., x_n)^T \in \{0, 1\}^n \end{aligned} \tag{2.7}$$

where, $c_{ij} \geq 0$ is the *profit* of the jth item in the ith knapsack, $w_{ij} \geq 0$ is the *weight* of the jth item in the ith knapsack, and W_i is the *capacity* of the ith knapsack. When $x_j = 1$, it means that the jth item is selected and put in all knapsacks. The MOKP is NP-hard and can model a variety of applications in resource allocation. It was formulated and solved by [Zitzler & Thiele 1999] using the strength Pareto evolutionary algorithm (SPEA). Since then the problem has become a standard benchmark that has been solved by many other researchers as [Deb *et al.* 2000, Zhang & Li 2007, Bandyopadhyay *et al.* 2008] and widely used in testing multiobjective metaheuristics.

2.5 Classification of multiobjective methods

Depending on when the Decision-maker (DM) articulate his preference concerning the different objectives and how optimization and the Decision process are

combined, multiobjective optimization methods can be broadly classified into the following categories as summarized in Fig. 2.7

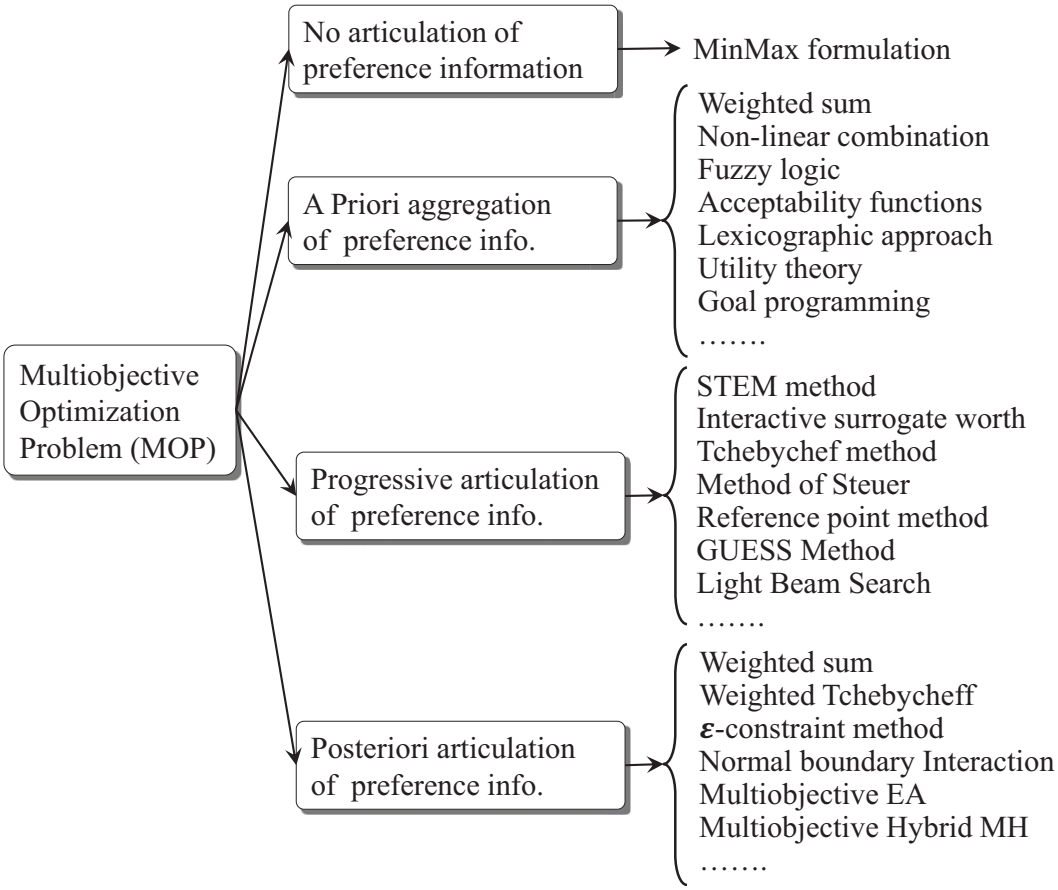


Figure 2.7: Classification of MOP solution methods

- No articulation of preference information (search only): In this method DM's opinions are not taken into consideration, so the multiobjective optimization problems are treated using more relatively simple method and the solution obtained is presented to the DM to accept or reject the solution. Examples are the Min-Max formulation and global criterion method [Hwang *et al.* 1980, Steuer 1986, Miettinen 1999]

- Priori articulation of preference information (Decision before search): The DM's in this method must specify his preference and opinions before solution stage (Fig. 2.8). The difficulty is that the DM does not necessarily know what is possible to attain in the problem and how realistic the expectations are. The most easy and widely used method is the weighted sum approach [Steuer 1986]. They also include several methods such as Goal programming, utility function, lexicographic approach...etc[Miettinen 1999].

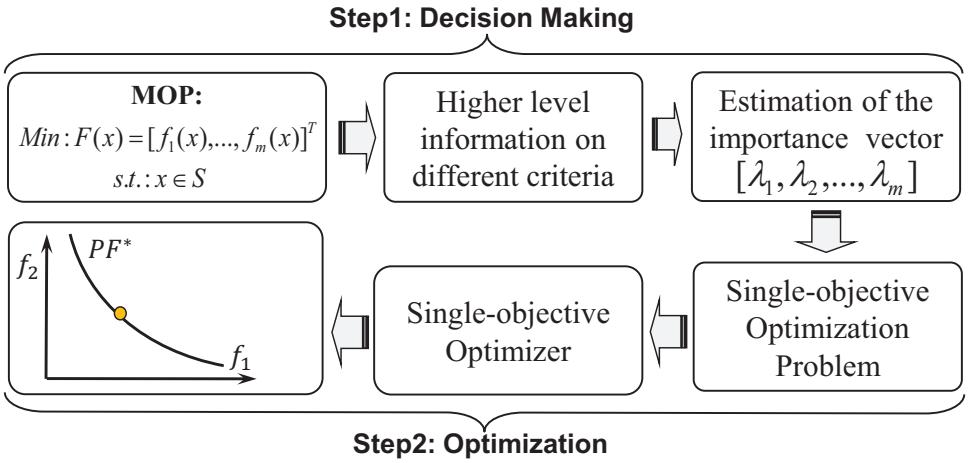


Figure 2.8: Mechanism of Priori methods

- Progressive articulation of preference information (Decision during search): The DM can articulate preferences during the interactive optimization process. After each optimization step, a number of alternative tradeoffs is presented on the basis of which the DM can interact though specification of his farther preference information, respectively guides the search. These methods usually progresses by changing the search direction base on inputs from DM or by progressively reducing the search space. They contains several methods such as STEM method [Benayoun *et al.* 1971], Steuer's Method [Steuer & Choo 1983], interactive surrogate worth method [Miettinen 1999]...etc. The advantages of these types of methods are:

  1. There is no need for "a priori" preference information,
  2. It is a learning process where the DM gets a better understanding of the problem.
  3. Only local preference information is needed.
  4. Its is more likely that the DM accepts the final solution as he takes an active part in the search.

  Some disadvantages are:

  1. The Solutions are depending on how well the DM can articulate his preference.
  2. A high effort is required from the DM during the whole process.
  3. The required computational effort is higher than in the previous methods.

4. In case of changing the DM or even the preferences, the search process has to started.

- Posteriori articulation of preference information (Decision after search): There are a number of techniques that allow to first search the solution space for generating all the Pareto optimal solutions or a representative subset of the Pareto optimal solutions and then present them to the decision maker (Fig. 2.9). These techniques attempts to solve the multiobjective optimization problem by constructing several scalarizations. The solution to each scalarization yields a Pareto optimal solution, whether locally or globally. some of these methods such as normal boundary intersections (NBI) [Das & Dennis 1998] and normal constraints (NC) [Messac *et al.* 2003] are constructed with the target of obtaining evenly distributed Pareto points that give a good evenly distributed approximation of the real set of Pareto points. The big advantages with these types of methods are that the solutions are independent from the DM preferences. However some of these methods suffer from large computation and the DM has too many solutions to choose from.

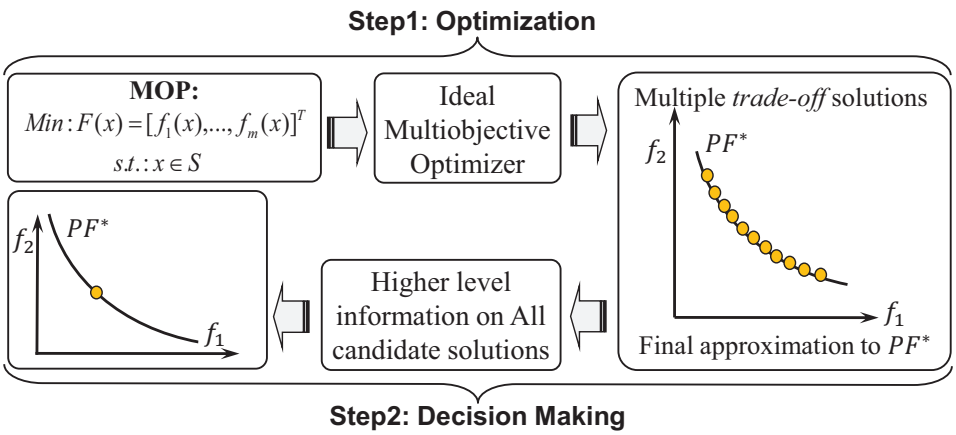


Figure 2.9: Mechanism of Posteriori methods

2.6 Quality Assessment

As mentioned before, the set of Pareto-optimal solutions $\mathcal{P}^*$ is the target output for MOP solution method. Thus, it is so important for multiobjective optimization to assess and compare the relative quality of the respective outputs of their solution methods and algorithms. However, It is not straightforward to measure the quality of a given technique in generation good approximations of Pareto front $\mathcal{PF}^*$ in this section, some concepts and definitions related to performance

assessment in multiobjective optimization methods are introduced. we refer to [Zitzler *et al.* 2008] for more details.

2.6.1 Concepts and definitions

Definition 2.19 (Pareto Front Approximation) *An approximation of a Pareto front in a attainable objective space $\mathcal{Z}$ is a set $A \in 2^{\mathcal{Z}}$ which contains only mutually non-dominated points i.e. $\forall x, \acute{x} \in A, x \prec\succ \acute{x}$. We denote the set of all such approximations as $\phi_{\mathcal{Z}}$.*

Definition 2.20 (Quality Indicator) *A quality indicator is a function $I : \phi_{\mathcal{Z}} \rightarrow \mathbb{R}$ which assigns a real value to each approximation, the smaller is being the better.*

A quality indicator is thus a function which associates a quantitative measure of goodness to each approximating set, in the same way a cost function ranks each solution in single criteria optimization. Many variants have been proposed and we present two of them which are commonly used.

2.6.2 Assessment indicators

In this thesis, two types of assessment indicators are used. The first one is the binary indicators which are used to compare each pair of techniques. The second type is the unary indicators which are used to assess each technique independently of the others. In the following, the assessment metrics which are used in this thesis are briefly discussed.

Let A, $B \subset \mathbb{R}^m$ be two approximations to the Pareto front ($\mathcal{PF}^*$). Let $P^* \subset \mathbb{R}^m$ be a set of uniformly distributed points along the $\mathcal{PF}^*$ or an upper approximation to the $\mathcal{PF}^*$ (e.g. "*Reference Set*"). Given a reference point $r^* \in \mathbb{R}^m$, the following metrics can be expressed as follows:

2.6.2.1 The Set Coverage (I_C)

The I_C indicator [Zitzler & Thiele 1999] is used to compare two approximation sets. In this indicator, the function I_C maps the ordered pair (A, B) to the interval $[0, 1]$ as follows:

$$I_C(A, B) = \frac{|u : u \in B, \exists v : v \in A, v \prec u|}{|B|} \tag{2.8}$$

where $I_C(A, B)$ represents the percentage of the solutions in B that are dominated by at least one solution from A. The value of $I_C(A, B) = 1$ means all solutions in B are dominated by or equal to solutions in A. In contrast, the

value of $I_C(A, B) = 0$ means none of the solution in a set B are covered by the set A. $I_C(B, A)$ is not necessarily equal to $1 - I_C(A, B)$. If $I_C(A, B)$ is large and $I_C(B, A)$ is small, then A is better than B in a sense.

2.6.2.2 The Hypervolume (I_{Hyp})

The hypervolume I_{Hyp} indicator can be used as a unary or a binary indicator. In its unary form, the hypervolume indicator associated with an approximation set A describes the volume of the hyper objective space that is weakly dominated by the points of the approximation set A and dominates the reference point r^* [Zitzler & Thiele 1999] as illustrated in Fig. 2.10 (left). The larger the value of I_{Hyp}, the better is the approximation set A. The I_{Hyp} indicator is mathematically defined as follows:

$$I_{Hyp}(A, r^*) = \mathcal{L}\left(\cup_{u \in A} \{y : u \preccurlyeq y \preccurlyeq r^*\}\right) \tag{2.9}$$

where $\mathcal{L}$ is the Lebesgue measure of a set. The binary form of the hypervolume indicator is usually called the "referenced hypervolume" (I_{Rhyp}). The I_{Rhyp} indicator represents the hypervolume dominated by the reference set P^* and not by A as shown in Fig. 2.10 (right). The more the value of I_{Rhyp} close to 0, the better is the approximation A. The I_{Rhyp} is expressed in terms of the I_{Hyp} as follows:

$$I_{Rhyp}(A, P^*, r^*) = I_{Hyp}(P^*, r^*) - I_{Hyp}(A, r^*) \tag{2.10}$$

Generally, the reference set P^* is considered as the optimal Pareto front or alternatively extracted from the union of all obtained fronts. The reference point r^* may also be expressed using the upper bound of each objective function.

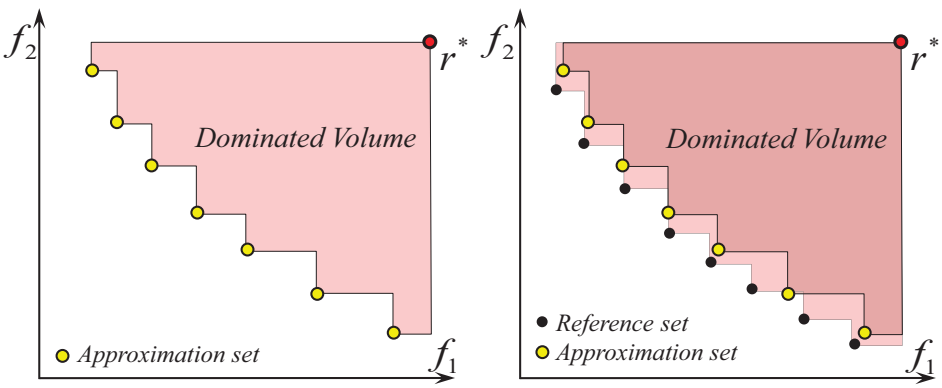


Figure 2.10: Illustration of hypervolume and referenced hypervolume

2.6.2.3 Generational Distance and Inverted Generational Distance

Generational Distance (I_{GD})[Van Veldhuizen & Lamont 2000] and Inverted Generational Distance (I_{IGD}) of a set A are defined as:

$$I_{GD}(A, P^*) = \frac{\sum_{u \in A} \{min_{v \in P^*} d(u, v)\}}{|A|} \tag{2.11}$$

$$I_{IGD}(A, P^*) = \frac{\sum_{u \in P^*} \{min_{v \in A} d(u, v)\}}{|P^*|} \tag{2.12}$$

where $d(u, v)$ is the Euclidean distance between u, v in $\mathbb{R}^m$. The $I_{GD}(A, P^*)$ measures the average distance from A to the nearest solution in P^* that reflects the closeness of A to P^*. If the approximation set A is included in the reference set P^*, the generational distance will be equal to 0. In contrast, the $I_{IGD}(A, P^*)$ measures the average distance from P^* to the nearest solution in A that reflects the spread of A to a certain degree. The lower value of both $I_{GD}(A, P^*)$ and $I_{IGD}(A, P^*)$ means the better quality of A in terms of convergence and diversity respectively.

2.6.2.4 R3-indicator (I_{R_3})[Knowles & Corne 2002]:

The I_{R_3} indicator uses a set of utility functions u, which can be any scalar function that expresses the preferences of the decision maker. Using a sufficiently large set of evenly distributed normalized weight vectors Λ, the utility functions u is defined by weighted sum function or weighted Tchebycheff function or augmented Tchebycheff function as used here. The I_{R_3} indicator can be evaluated as follows:

$$I_{R_3}(A, P^*) = \frac{\sum_{\lambda \in \Lambda} \left[u^*(\lambda, P^*) - u^*(\lambda, A)\right] / u^*(\lambda, P^*)}{|\Lambda|} \tag{2.13}$$

where u^* represents the maximum value of the utility function u using the weight vector λ and the Pareto approximation set A as follows:

$$u^*(\lambda, A) = max_{z \in A} u(\lambda, z)$$

where $u(\lambda, z) = -(max_{1 \leq j \leq m} \lambda_j |z_j^* - z_j| + \rho \sum_{j=1}^{m} |z_j^* - z_j|)$ represents the augmented Tchebycheff utility function, ρ is a small positive integer, z^* is the ideal objective vector. This is for each weight vector $\lambda = [\lambda_1, ..., \lambda_m] \in \Lambda$, such that $\lambda_i \in [0, 1]$ and $\sum_{i=1}^{m} \lambda_i = 1$. The I_{R_3} is a hybrid indicator that measures both convergence and diversity. It gives a scalar value in the range $[0, \infty)$. The smaller the value of $I_{R_3}(A, P^*)$, the better is the approximation set A.

2.6.2.5 Maximum Spread (I_{MS}):

This metric was proposed by [Zitzler *et al.* 2000] to evaluate the maximum extension covered by the non-dominated solutions in A as follows:

$$I_{MS}(A) = \sqrt{\sum_{j=1}^{m} \left[\left(max_{i=1}^{|A|} f_j^i \right) - \left(min_{i=1}^{|A|} f_j^i \right) \right]^2} \tag{2.14}$$

where m is the number of objectives in the given problem. One should note that the higher value indicate the better performance.

2.6.2.6 The unary additive Epsilon ($I_{\varepsilon+}$): [Zitzler *et al.* 2003]

The unary additive ε-indicator $I_{\varepsilon+}$, which is a distance-based indicator, gives the minimum ε value by which each point in an approximation set A has to be translated in the objective space to weakly dominate the reference set P^* as the following formula:

$$I_{\varepsilon+}(A, P^*) = inf_{\varepsilon \in \mathbb{R}} \{ \forall z' \in P^*, \exists z \in A : z \preccurlyeq_{\varepsilon+} z^2 \} \tag{2.15}$$

where $z \preccurlyeq_{\varepsilon+} z'$ if and only if: $z_j - \varepsilon \geq z'_j$, this is for all $j = 1, ..., m$.

2.7 Classical Methods for MODM

Classical methods for generating the Pareto optimal set aggregate the objectives into a single, parameterized objective function by analogy to decision making before search. Several optimization runs with different parameters setting are performed in order to achieve a set of solutions which approximates the Pareto optimal set. This section introduces the main methods for solving the MODM problem which are:

1. Scalarization Techniques
2. Interactive Approach.
3. Goal Programming Approach.
4. Fuzzy programming Approach.

2.7.1 Scalarization Techniques

These methods have been proposed for characterizing optimal solution depending upon the different methods to scalarize the Multiobjective problems. In this section, we review several popular techniques that related to our work in this thesis.

2.7.1.1 Global criterion methods

The main idea behind most global criterion methods [Zeleny 1982, Miettinen 1999] is to use an *exponential sum* to yield a single function. They are based on minimizing the relative distances (typically expressed as $\mathcal{L}_p$-norm) between the candidate solution and the *ideal* point z^*. Thus the MOP stated in (2.1) is reformulated as follows:

$$\begin{aligned} &\underset{x}{\text{Minimize}} \quad \left(\sum_{i=1}^{m} (f_i(x) - z_i^*)^p \right)^{1/p} \\ &\text{subject to} \quad x \in \mathcal{S} \end{aligned} \tag{2.16}$$

where $1 \leq p \leq \infty$. The value of p provides different ways to calculate the distance. the most frequently used values is $p = 1$ for simple formulations ("*Objective sum method*") and $p = 2$ for Euclidean distance, and $p = \infty$ for Min-max formulation ("*Tchebycheff norm*") (see Fig. 2.11). The output is just one point on the Pareto front, which the decision maker has to accept as a final solution. different points in the Pareto front can be obtained by changing the exponent p and by assigning the single objectives different weights.

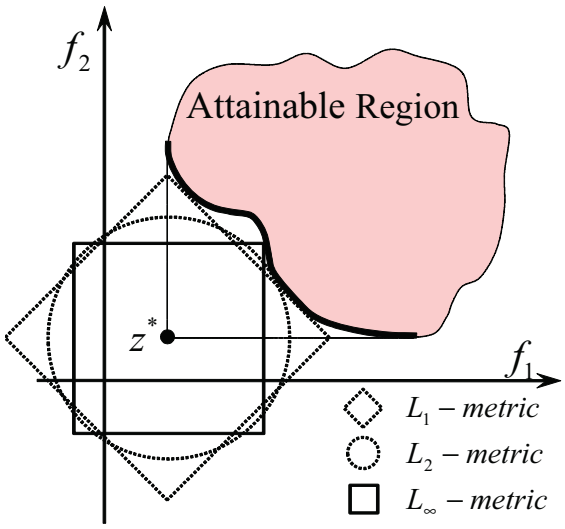


Figure 2.11: Global criterion method

2.7.1.2 The weighted sum approach

The weighted sum approach [Gass & Saaty 1955, Miettinen 1999] was the first technique developed for the generation of *noninferior* solutions for multiobjective optimization. This method is very efficient computationally speaking, and

can be applied to generate a strongly nondominated solutions to be used as an initial solution in other techniques. In this method the MOP stated in (2.1) is transformed into weighted sum problem $F^{ws}(x, \Lambda)$ by associating each objective function with a weighting coefficient and minimize the weighted sum of the objectives as follows:

$$\begin{aligned} \underset{x}{\text{Minimize}} \quad & F^{ws}(x, \Lambda) = \sum_{j=1}^{m} \lambda_j f_j(x) \\ \text{subject to} \quad & x \in \mathcal{S} \end{aligned} \tag{2.17}$$

where $\Lambda = [\lambda_1, ..., \lambda_m]$ is a m-components weight vector such that $\sum_{j=1}^{m} \lambda_j = 1$ and $\lambda_j \geq 0$ for each j. The weighting coefficients of the weighted sum problem give the "trade-off" rate information between the objective functions. This method suffers from some drawbacks. First, a satisfactory, a priory selection of weights does not necessarily guarantee that the final solution will be acceptable. Second, it is impossible to obtain points on non-convex portions in of the Pareto optimal set in the objective space by solving the problem $F(x, \Lambda)$[Das & Dennis 1997]. This is illustrated in Fig. 2.12 for weights λ_1, λ_2, solution x is sought to minimize $y = \lambda_1 f_1(x) + \lambda_2 f_2(x)$. This equation can be reformulated as $f_2(x) = (-\lambda_1/\lambda_2) f_1(x) + y/\lambda_2$, which defines a line with slope $(-\lambda_1/\lambda_2)$ and intercept y/λ_2 (solid line in Fig. 2.12). Graphically, the optimization process corresponds to move this line downwards until no feasible objective vector below it and at least one feasible objective vector is on it. However, the points (A and B) will never be detected whether the slope increased or decreased. Finally, varying the weights consistently and continuously may not necessarily result in an even distribution of Pareto optimal set.

2.7.1.3 The weighted metric method

The weighted metric method [Zeleny 1982, Miettinen 1999] seeks to minimizing the weighted distances to the *ideal* point z^*. Thus the MOP stated in (2.1) is reformulated according to formula (2.18).

$$\begin{aligned} \underset{x}{\text{Minimize}} \quad & F(x, \Lambda, z^*) = \left(\sum_{i=1}^{m} \lambda_i \left| f_i(x) - z_i^* \right|^p \right)^{1/p} \\ \text{subject to} \quad & x \in \mathcal{S} \end{aligned} \tag{2.18}$$

where $\Lambda = [\lambda_1, ..., \lambda_m]$ is a m-components weight vector such that $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$ for each i and $1 \leq p \leq \infty$. As mentioned before, the exponent p provides different ways to calculate the distance. For $p = 1$, the formula 2.18 gives the *weighted sum method*, for $p = 2$ the distance is a weighted Euclidean

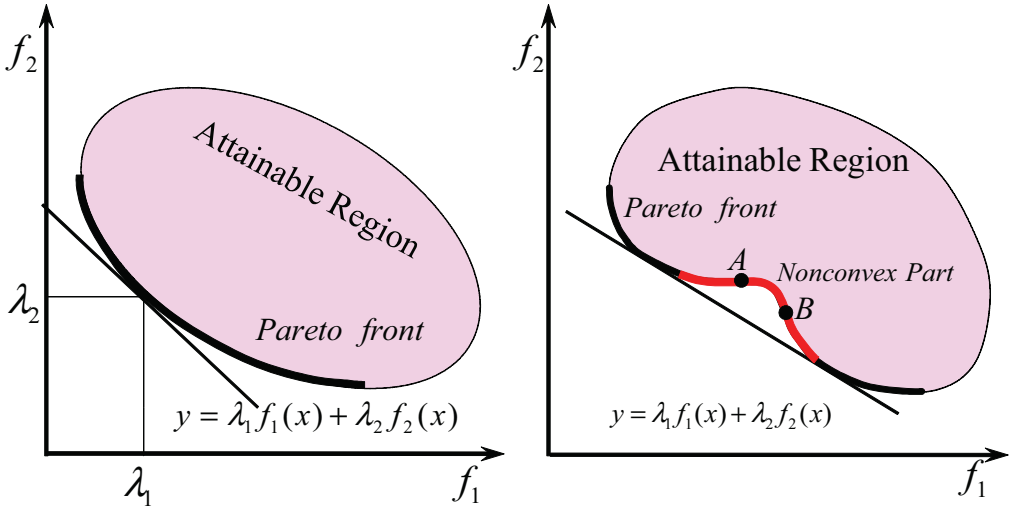


Figure 2.12: Weighted sum approach

distance in $\mathbb{R}^m$. The formulation with $p = \infty$ leads to the so-called *weighted Tchebycheff* problem that is formulated as follows:

$$\begin{aligned} &\underset{x}{\text{Minimize}} \quad F^{tc}(x, \Lambda, z^*) = \underset{1 \le i \le m}{Max}\{\lambda_i \, |f_i(x) - z_i^*|\} \\ &\text{subject to} \quad x \in \mathcal{S} \end{aligned} \tag{2.19}$$

For each Pareto-optimal point x^* there exists a weight vector Λ such that x^* is the optimal solution of $F^{tc}(x, \Lambda, z^*)$ stated in (2.19) and each optimal solution of $F^{TC}(x|\Lambda, z^*)$ is a Pareto optimal solution of MOP stated in (2.1). Therefore, one can obtain different Pareto optimal solution by changing the weight vector. this method is theoretically interesting because it has the ability to find any Pareto-optimal point. even for non-convex problems. The only weakness of this approach is that its aggregation function is not smooth for a continuous MOP.

2.7.1.4 The ε-constraint method

The ε-constraint method [Haimes *et al.* 1971, Miettinen 1999] is one of the best known scalarization techniques to handle MOP. It converts the MOP into its corresponding j^{th} ε-constrained problem that formulated by selecting one of the objective functions as the objective function while the others are used as constrains bounded by some allowable levels ε_i. The ε-constraint problem $F(x, \varepsilon)$ is

defined by formula 2.20 as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & F(x,\varepsilon) = f_j(x) \\ \text{subject to} \quad & f_i(x) \leq \varepsilon_i,\ \forall i = 1,2,...,m.\ i \neq j \\ & x \in \mathcal{S} \end{aligned} \tag{2.20}$$

In order to find multiple Pareto optimal solutions, the upper bounds, ε_i, are var-

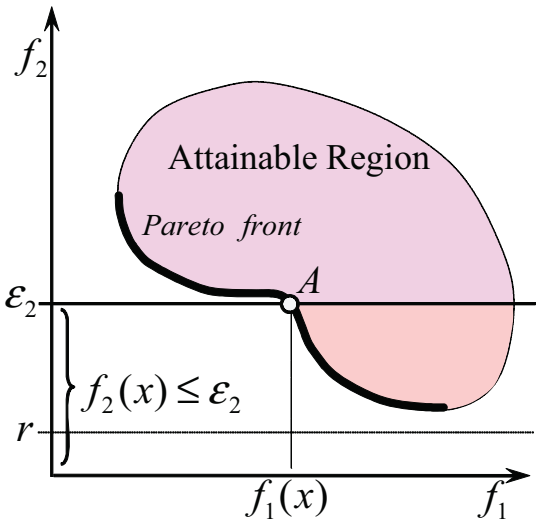


Figure 2.13: Epsilon constrained method

ied by the optimizer. As the converse of the weighting method, the ε-constraint method has the ability to obtain solutions associated with non-convex parts of the Pareto front $\mathcal{PF}^*$. Fig. 2.13 depicts this fact, by setting $j = 1$, the bold line represents the new constraint, while the decision vector related to the point A minimizes $f_1(x)$ among the remaining solutions. It can be seen from Fig. 2.13 the weakness of this method. If the upper bounds are not chosen appropriately (i.e. $\varepsilon_2 = r$), the obtained new feasible set might be empty, i.e. there is no feasible solution to $F(x,\varepsilon)$. To avoid this problem, a suitable rang of values for the ε_i must have been known.

2.7.1.5 Normal Boundary intersection (BI) methods

The Normal boundary intersection (NBI) [Das & Dennis 1998] seeks to find evenly distributed points on the Pareto front in response of deficiencies.in the weighted sum approach. It originally designed for nonlinear continuous optimization problems. the idea is the projecting elements of specific convex hull (CHIM) of points towards the boundary of the attainable region $\mathcal{Z}$ in the objective space.

Definition 2.21 (Convex hull of Individual Minima (CHIM)) *for every* i=1, ..., m,.*let* $x_i^* \in \mathcal{S}$ *be the point in decision space which minimizes the* i_{th} *objective function* $f_i(x)$ *in MOP stated in (2.1), Let* $F_i^* = F(x_i^*)$ *and* z^* *is the ideal point, The convex hull of the individual minima (CHIM) is the set*

$$CHIM = \{M\Lambda : \Lambda = [\lambda_1, ..., \lambda_m]^T, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0\}$$

where M *is* $m \times m$ *pay-off matrix whose* i^{th} *column is* $F_i^* - z^*$. *Thus, the diagonal elements of* M *is* 0.

Fig. 2.14 depicts the CHIM for two-objective problem. The NBI method transforms the MOP into the formula 2.21 as follows

$$\begin{aligned} \underset{x,t}{\text{minimize}} \quad & F^{nbi}(x, \Lambda, z^*) = t \\ \text{subject to} \quad & M\Lambda + t\hat{n} = F(x) - z^*, \\ & x \in \mathcal{S}, t \in \mathbb{R} \end{aligned} \tag{2.21}$$

where Λ is a weight vector, $\hat{n} = -M(\overbrace{-1, ..., -1}^{m})$ is a quasi normal vector points towards the origin, $M\Lambda$ represents a point in the CHIM. $\hat{n}$ gives the NBI method the property that for any Λ, a solution point is independent of how the objective functions are scaled. As Λ is systematically modified, the solution to (2.21) yields an even distribution of Pareto optimal points representing the complete Pareto front. this nice feature makes the method really appealing in practice. one drawback of the NBI technique is that it may return points which are dominated if the boundary is non-convex.

2.7.1.6 Penalty Boundary intersection (PBI) method

The Normal boundary intersection (NBI) [Das & Dennis 1998] and the Normal constraint method [Messac *et al.* 2003] are classified as boundary intersection BI approaches. they were designed for continuous MOP. Geometrically, these BI approaches aim to find intersection points of the most bottom boundary and set of lines. if these lines are evenly distributed in a sense, one can expect that the result intersections points provides a good approximation to the whole $\mathcal{PF}^*$. Mathematically, we consider the following scalar optimization problem

$$\begin{aligned} \text{minimize} \quad & F^{bi}(x, \Lambda, z^*) = d \\ \text{subject to} \quad & F(x) - z^* = d\Lambda \\ & x \in \mathcal{S} \end{aligned} \tag{2.22}$$

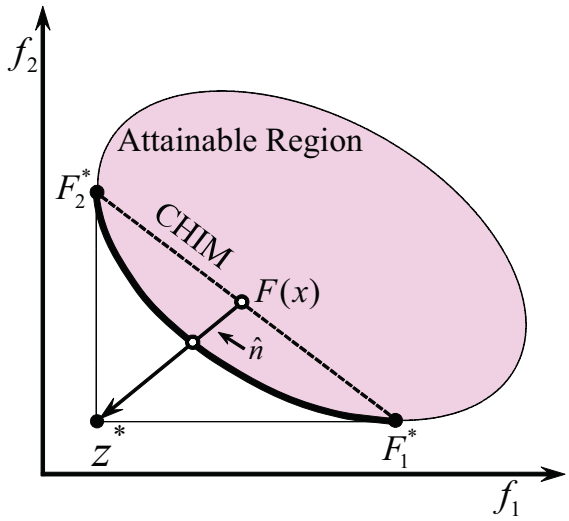


Figure 2.14: Illustration of CHIM on two-objective problem

where Λ and z^* are the weight vector and the ideal point respectively. As illustrated in Fig. 2.15 left, the constraint $F(x) - z^* = d\Lambda$ ensures that $F(x)$ is always in L, the line with direction Λ and passing through z^*. the goal is to push $F(x)$ as low as possible so that i reaches the boundry of the attainable region. One of the drawbacks of this approach is that it has to handle the equality constraints. the penalty can deal with the constrains as follows

$$\begin{aligned} &\text{minimize} && F^{pbi}(x, \Lambda, z^*) = d_1 + \theta d_2 \\ &\text{subject to} && x \in \mathcal{S} \end{aligned} \tag{2.23}$$

where

$$d_1 = \frac{\left\|(F(x) - z^*)^T \Lambda\right\|}{\|\Lambda\|}$$

and

$$d_2 = \|(F(x) - (z^* + d_1 \Lambda)\|$$

θ is a preset penalty parameters. let y is the projection of $F(x)$ on the line L. As shown in Fig. 2.15 right, d_1 is the distance between z^* and y. d_2 is the distance between $F(x)$ and L. if θ is set appropriately, the solutions to (2.22) and (2.23) should be very close. this method is called penalty boundary intersection (PBI) method. the general (BI) approach have advantages over Tchebycheff approach such as in case of more than two objectives, if both BI approaches an Tchebycheff use the same set of evenly distributed weight vectors , the resultant optimal solutions in PBI should much more uniformly distributed than those obtained by Tchebycheff [Das & Dennis 1998]. second, if x dominats y it is still

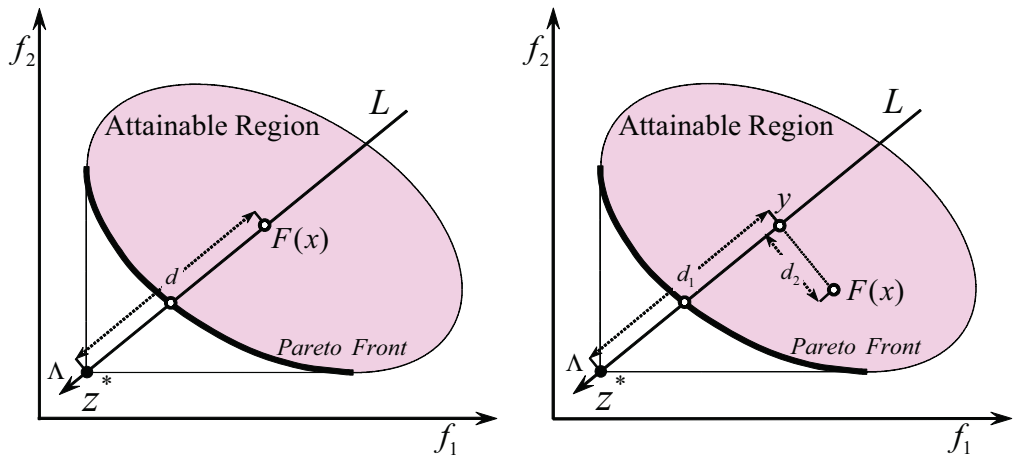


Figure 2.15: Illustration of BI approach (left) and PBI approach (right)

possible that $F^{tc}(x, \Lambda, z^*) = F^{tc}(y, \Lambda, z^*)$ while it is rare for PBI or BI aggregation functions.

2.7.2 Interactive methods

An interactive approach (Fig. 2.16) requires active progressive interaction between the decision maker and the analyst throughout the solution process. The main frame of an interactive approach is normally characterized by three basic steps:

1. Solve the problem based on some initial set of parameters to obtain a feasible, preferable non-inferior solution.
2. Ask the decision maker to evaluate this solution.
3. Use the decision maker's response to formulate a new set of parameters, forming a new problem to be solved.

Step (1)-(3) are repeated until the decision maker is satisfied with the current solution or no further action can be taken by the method. When the interactive algorithms are applied to real life problems the most critical factor is the functional restrictions placed on the objective functions, constraints on the unknown preference function. There exist some methods of interaction the light focused on the following:

- *Feasible region reduction methods:* Each iteration of these methods generally consists of three phases. In the calculation phase, an efficient solution which is nearest to the ideal obtained. In the decision phase, the DM interacts with the method, and his response is used to construct additional constraints in the feasible region as a reduction phase.

- *Feasible direction methods:* These methods starts with a feasible solution and interactively performance to two steps: in the direction finding step, the DM provides the information about his preference in the neighborhood of the current solution by specifying values of local trade-off among criteria. Then the step size is determined from the current solutions along the usable direction.

- *Trade off cutting plane method:* This type is unique in the way. It isolates the best compromise solution by reducing the objective space using cutting plane method.

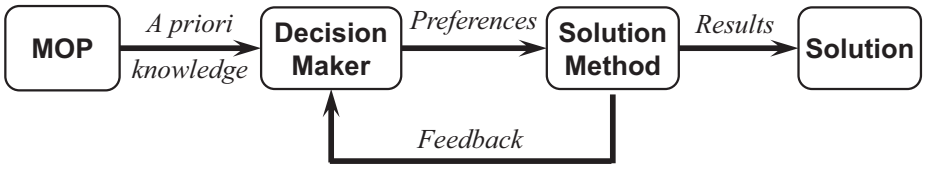


Figure 2.16: Illustration of Interactive approach for MOP

There is a number of real-world situations in which the decision maker cannot easily discriminate between a pair of alternatives because of cognitive strain or because of the lack of sufficient information.

2.7.3 Goal programming methods

In goal programming [Charnes *et al.* 1955, Charnes & Cooper 1961, Ijiri 1965], the goals (aspiration levels) b_j are specified for each objective function $f_j(x)$. Then, the total deviation from the goals $\sum_i^m |d_j|$ is minimized, where d_j is the deviation from the goal b_j for the jth objective. To model the absolute values, d_j is split into two parts d_j^+ and d_j^-, such that $d_j^+ \geq 0$, $d_j^- \geq 0$ and $d_j^+ d_j^- = 0$. Consequently, $|d_j| = d_j^+ - d_j^-$. The deviational variables d_j^+ and d_j^- represents underachievement and overachievement respectively, where achievement implies that a goal has been reached. In general, the MOP is reformulated as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} (d_i^+ + d_i^-) \\ \text{subject to} \quad & f_j(x) + d_j^+ - d_j^- \leq b_j, \forall j = 1, ..., m. \\ & d_j^+, d_j^- \geq 0, \, d_j^+ \times d_j^- = 0, \forall j = 1, ..., m. \\ & x \in \mathcal{S} \end{aligned} \tag{2.24}$$

Lee and Olson [Lee & Olson 1999] provide an extensive review of applications for goal programming. However, despite its popularity and wide range of ap-

plications, there is no guarantee that it provides a Pareto optimal solution. In addition, Equ. 2.24 has additional variables and nonlinear equality constraints, both of which can be troublesome with larger problems. There are several methods for treating the goal programming; some of them are stated here as follows:

- *Archimedean goal programming* (or weighted goal programming) constitutes a subclass of goal programming, in which weights are assigned to the deviation of each objective from its perspective goal [Charnes & Cooper 1977].
- *The preemptive* (or lexicographic) goal programming approach is similar to the lexicographic method in that the deviations $|d_j| = d_j^+ + d_j^-$ for the objectives are ordered in terms of priority and minimized lexicographically.

Archimedean goal programming and preemptive goal programming provide Pareto optimal solutions if the goals form a Pareto optimal point or if all deviation variables, d_j^+ for functions being increased and d_j^- for functions being reduced; have positive values at the optimum [Miettinen 1999]. The latter condition suggests that all of the goals must be unattainable. Generally, however, Archimedean and preemptive goal programming can result in non-Pareto optimal solutions [Zeleny 1982]. Zeleny [Zeleny 1982] briefly mentions multigoal programming, in which various functions of are minimized as independent objective functions in a vector optimization problem. Hwang and Masud [Hwang & Masud 1979] present the goal attainment method, initially proposed by Gembicki [Gembicki & Haimes 1975], which is computationally faster than typical goal programming methods. Finally , we should refer to a recent textbook [Jones & Tamiz 2010] in which a comprehensive overview of the state-of-the-art in goal programming is provided.

2.7.4 Fuzzy programming methods

In Reality, majority of the optimization models in real life contains linguistic terms. Fuzzy sets theory created an opportunity to handle linguistic terms in mathematical programming models. The concept of fuzzy sets [Sakawa 1993] is based on multi valued logic where a statement could be simultaneously party true and party false. In fuzzy logic, a membership function $\mu(.)$ expresses the degree of truthfulness of a statement, in the range from $\mu = 0$, indicating that the statement is false to $\mu = 1$ for truth. This is the opposite of binary logic where a statement can be only false or true. In case of multiobjective problem, the membership function enables the problem solver to associate a normalized value to each objective $\mu_i(f_i(x))$, which expresses the degree of satisfaction of the considered objective . The value of $f_i(x)$ is fuzzified by μ_i to yield a value in

the range [0,1], which quantifies how well a solution satisfies the requirements. An example of a membership function is depicted in Fig. 2.17.

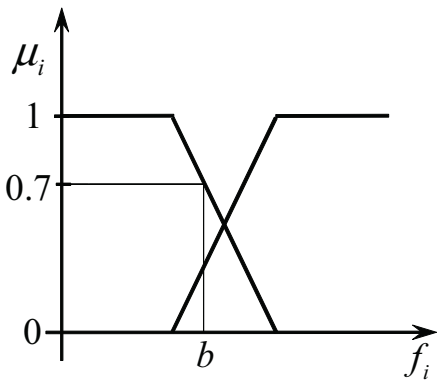


Figure 2.17: Illustration of a membership function

Once the fuzzification has been performed the actual value of each objective has been transformed into logic values. These values have to be aggregated to one in order to get an overall value for the design. In fuzzy logic, this could be implemented by several different rules. The most common are the *min* operator which returns as an output the minimum value of the μ_i on which is operates, and the *product* operator which returns the product of all individual operators. Hence the overall objective function can be expressed as:

$$\begin{aligned} &\underset{x}{\text{Minimize}} \quad \prod_{i=1}^{m} \mu_i(f_i(x)) \\ &\text{subject to} \quad x \in \mathcal{S} \end{aligned} \tag{2.25}$$

2.8 Drawbacks of classical Methods

Although, the conventional methods are widely used in solving MODM problems, they are suffering from some drawbacks as discussed below.

1. There are some assumptions must be valid in the objective functions and/or constraints in the problem under consideration. Such as smoothness, differentiability, continuity, etc. This weak point makes these methods unable to deal with the optimization problems that violates these assumptions.
2. These methods gives only one efficient solution at a time, this means many different optimization runs needed to obtain multiple solutions.
3. When dealing with large-scale problems that contain huge number of constraints and objectives, these conventional methods need a long time expo-

nentially proportion with the size of the problem to reach a set of efficient solution.

4. Most of these classical methods can find Pareto-optimal solutions. However, when dealing with special problems with non-convex Pareto-optimal fronts, they cannot find all the Pareto-optimal solutions.
5. Most of the classical methods and algorithms require some prior knowledge which acts as DM preferences such as suitable weights or ε values etc.[Deb 2001].

To overcome these limitations, the researchers found the metaheuristic based techniques, such as evolutionary computations, are a promising tool that can treat the drawbacks of conventional MODM methods. These techniques have the ability to find a set of optimal solutions in each simulation run. This was made possible during the last decade by using population-based metaheuristics and their hybridization whether with other metaheuristics or with other search techniques. These hybridization schemes are commonly called hybrid evolutionary metaheuristics. In hybrid evolutionary metaheuristics, since a population of solutions is processed in each iteration, the outcome is also a population of solutions. However, if an optimization problem has multiple optimal solutions, the hybrid evolutionary metaheuristic is suitable to capture multiple solutions in its final population. The next chapters will discuss and concentrate those techniques in details.

Chapter 3

Optimization Metaheuristics: A Survey

Contents

3.1 Introduction **36**

3.2 General Overview **36**

3.3 Metaheuristics **39**

3.4 Classification of Metaheuristics **41**

3.5 Single-solution based metaheuristics **43**

3.5.1 Greedy Heuristics 43

3.5.2 Simple Local Search 43

3.5.3 GRASP 46

3.5.4 Iterated Local Search 49

3.5.5 Variable Neighborhood Search 50

3.5.6 Guided Local Search 51

3.5.7 Simulated Annealing 54

3.5.8 Tabu Search 55

3.6 Population-based metaheuristics **57**

3.6.1 Evolutionary computations 57

3.6.2 Other Evolutionary Algorithms 63

3.6.3 Swarm Intelligence 70

3.7 Multiobjective Metaheuristics **75**

3.7.1 Fitness assignment strategies 75

3.7.2 Diversity preservation 77

3.7.3 Elitism 77

3.7.4 Multiobjective Evolutionary Algorithms 78

3.8 Hybrid Metaheuristics **83**

3.8.1 Hybrid Metaheuristics classifications 85

3.8.2 Multiobjective Hybrid Metaheuristics 87
3.9 Summary 90

3.1 Introduction

Nowadays, metaheuristic based techniques, especially hybrid metaheuristics play an important role in solving many real-world problems, which have often multiple-conflict objectives. Due to this importance, a lot of research efforts have been done in this promising area. In this chapter, an overview of metaheuristic search and optimization algorithms is provided. The chapter is organized as follows: Section 3.2 presents an overview of general search and optimization techniques. Metaheuristics and their characteristics are introduced in section 3.3. Section 3.4 discusses the different classifications of metaheuristics. In Sections 3.5 and 3.6, a quick review on some of the important metaheuristic algorithms is outlined with introducing the basic concepts of the algorithms related to the thesis work. In this review, the single solution based metaheuristic is discussed in details in section 3.5. Also, the population based metaheuristics which include Evolutionary Computations (EC) based techniques and Swarm Intelligence (SI) based algorithms are presented in section 3.6. Section 3.7 introduce some basic concepts related multiobjective optimization using metaheuristic approaches with highlighting some of these algorithm, specially those used in the comparative studies provided in this thesis work. Moreover, a review on metaheuristics hybridization is discussed in section 3.8. Finally, a summary for this chapter is provided in section 3.9.

3.2 General Overview

Many real-life optimization problems in diverse fields such as industry, science, engineering, economics, and business are often complex, NP-hard and consequently difficult to solve. These optimization problems usually have several contradictory objectives that are difficult to be satisfied simultaneously. In practice, their modeling is continuously evolving in terms of constraints and objectives. The resolution of these problems cannot be performed in an exact manner within a reasonable amount of time, and their resource requirements are ever increasing. To deal with such an issue, the design of resolution methods must be based on the joint use of advanced approaches from combinatorial optimization, large-scale parallelism and engineering methods.

Global Search and optimization techniques can be classified into two basic classes, deterministic and probabilistic (also called stochastic) as illustrated in Fig. 3.1.

Deterministic algorithms are most often used if a clear relation between the characteristics of the possible solutions and their utility for a given problem exists. Then, the search space can efficiently be explored to find "acceptable" solutions in "acceptable" time by incorporating problem domain knowledge. In deterministic algorithms, each execution step involves at most one way to proceed. If no way to proceed exists, the algorithm will terminate. Deterministic algorithms do not contain instructions that use random numbers in order to decide what to do or how to modify data. As shown in Fig. 3.1, deterministic algorithms include branch and bound search techniques, calculus based techniques and state space search [Russell & Norvig 2010, Bednorz 2008] which involve greedy search, hill-climbing algorithms, depth-first search, breadth-first search, best-first search and A^* search algorithm. These deterministic methods are successfully used in solving a wide variety of problems [Hwang & Masud 1979, Goldberg 1989, Mostaghim & Teich 2003]. If the relation between a candidate solution and its "fitness" are not so obvious or too complicated, or the dimensionality of the search space is very high, it becomes harder to solve a problem deterministically. However, many multiobjective optimization problems are high-dimensional, discontinuous, multimodal, and/or NP-Complete. Deterministic methods are often ineffective when applied to NP-Complete or other high-dimensional problems because they are suffering from their requirement for problem domain knowledge (heuristics) to direct or limit the search in these large search spaces. Problems exhibiting one or more of these above characteristics are termed irregular [Veldhuizen 1999].

Because many real worlds' scientific and engineering problems are irregular, therefore deterministic search techniques are unsuitable. In this case, probabilistic algorithms come into play as alternative algorithms which produce high quality (or near-optimal) solutions in a reasonable amount of time. The initial work in this area started from about six decades ago. Now, it has become one of most important research fields in optimization [Robbins & Monro 1951, Bledsoe & Browning 1959, Bremermann 1962]. Unlike deterministic algorithms, probabilistic search techniques include at least one instruction that acts on the basis of random numbers [Hromkovic 2005]. An especially relevant family of probabilistic algorithms are the Monte Carlo-based approaches. Stochastic search approaches such as Simulated Annealing (SA), Tabu search (TS) and Evolutionary Computation (EC) techniques were developed as alternative approaches for solving these irregular problems [Goldberg 1989, Michalewicz 1996]. Stochastic methods require function which assign fitness value to each possible solution, and a mapping mechanism between the problem and algorithm domains. Although some of them can find the optimum, most cannot guarantee the optimal solution. In general, they provide good enough solutions to a wide range of optimization

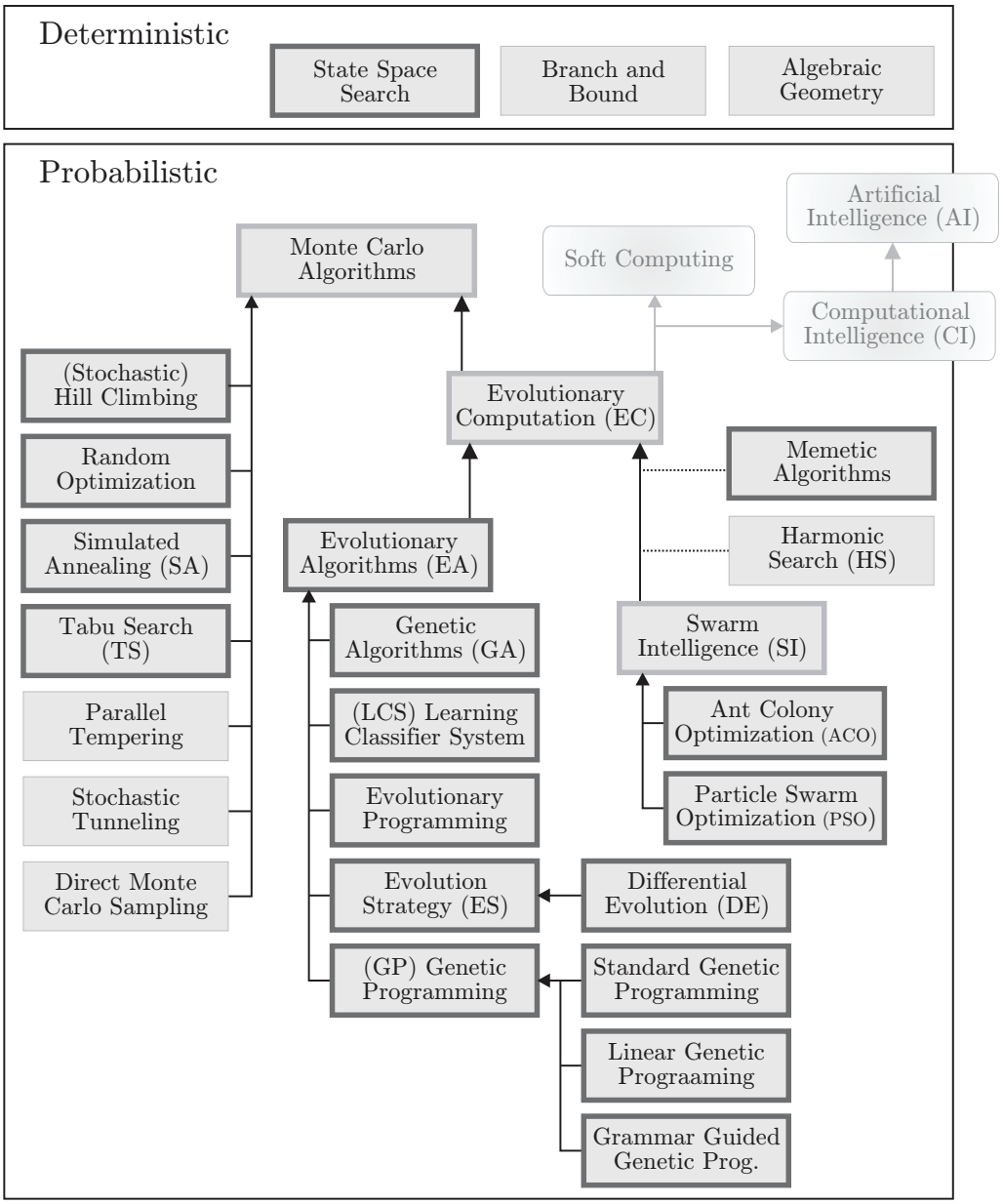


Figure 3.1: The taxonomy of global optimization algorithms [Weise 2009]

problems which traditional deterministic search methods find difficulty in such cases [Goldberg 1989]. In global optimization algorithms, heuristics help to decide which one of a set of possible solutions is to be examined next. Heuristics are usually employed in deterministic algorithms to define the processing order of the candidate solutions, as done in greedy search. Whereas probabilistic methods may only consider those elements of the search space that have been selected by the heuristic in further computations.

Heuristic [Michalewicz & Fogel 2004] can be defined as a "technique which seeks good (near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is" [Reeves 1993]. For example, constructive (greedy) algorithms which construct the solution in a series of steps based on the strategy of making the best decision (according to a certain criterion) at each step. Also, local search which explores neighboring solutions in an attempt to improve the current solution.

In the last three decades, a new advanced heuristic algorithms commonly called "Metaheuristics" have been widely developed and applied to a variety of optimization problems [Reeves 1993, Voss *et al.* 1999, Glover & Kochenberger 2003, Aarts & Lenstra 1997, Osman & Laporte 1996, Rayward-Smith 1996]. The term "Metaheuristics" is firstly introduced by Glover in [Glover 1986]. It can be described as an iterative search strategy that guides the process over the search space in the hope of finding the optimal solution.

According to Voss et al. [Voss *et al.* 1999], a metaheuristic is described as "an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method". In the following sections, the metaheuristic based techniques will be concentrated and described in more details.

3.3 Metaheuristics

As mentioned before, metaheuristics are a new class of approximate algorithms which try to combine basic heuristic methods in higher level frameworks in order to efficiently and effectively explore the search space. This class of algorithms includes, but is not restricted to, Genetic Algorithms (GA) [Holland 1975], Simulated Annealing (SA) [Kirkpatrick *et al.* 1983], Tabu Search (TS) [Hansen 1986], Artificial immune system [Farmer *et al.* 1986, Bersini & Varela 1990], Ant Colony Optimization (ACO) [Dorigo 1992], Particle Swarm Optimization (PSO) [Kennedy & Eberhart 1995], Differential Evolution (DE) [Kennedy & Eberhart 1995], Scatter Search (SS) [Glover 1977], Evolution Strategies (ES) [Rechenberg 1973], Estimation of Distribution algorithm [Baluja 1994] etc. As approximate algorithms, metaheuristics sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. Metaheuristic techniques have received increased interest due to their ability to provide "acceptable" solutions in a reasonable amount of time for solving hard and complex optimization problems. In

general, metaheuristics based techniques are not problem specific and contain mechanisms to avoid getting trapped on local optima. Due to the generality of the metaheuristic concept it is hardly possible to give a precise definition of what a metaheuristic exactly is. Metaheuristic search methods can be simply described as upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems [Talbi 2009]. We refer to [Blum & Roli 2003, Blum & Roli 2008] for several other proposed definitions.

The success of metaheuristics relies on providing a balance between the *exploitation (intensification)* of the accumulated search experiences and the *exploration (diversification)* of the search space to identify regions with high quality solutions in the search space. Intensification refers to focusing the search into certain regions of the solution space to discover the high quality solutions, while diversification refers to expanding the search by exploring unvisited regions of the solution space. Nevertheless of the problem domain, the intensification and diversification mechanisms are fundamental components of any global search method. The metaheuristic should achieve a dynamic and adaptive compromise between intensification and diversification in order to be able to achieve good results. Adequate balance between diversification and intensification is important, on one side to quickly identify regions with high quality solutions, and on the other side not to waste too much time in regions of the search space which are either already explored or which do not provide high quality solutions. The main differences among the existing metaheuristics concerns the particular way in which they try to achieve this balance.

According to [Blum & Roli 2003], metaheuristics are characterized by the following properties:

- Metaheuristics are strategies that "guide" the search process to efficiently explore the search space for finding near-optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics permit an abstract level description.
- Metaheuristics are not problem-specific and may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.

In the following section, the different classifications of metaheuristics will be

discussed

3.4 Classification of Metaheuristics

Metaheuristic approaches can be classified according to different aspects which are usually related to how these approaches operate over the search space during the search process. Some of these aspects are discussed as follows:

- *Nature inspired vs. non-nature inspired*: Many metaheuristics are inspired by natural processes: evolutionary algorithms and artificial immune systems from biology; ants, bees colonies, and particle swarm optimization from swarm intelligence into different species (social sciences); and simulated annealing from physics. Sometimes, it is difficult to identify to which class a particular algorithm can be classified, as illustrated in [Blum & Roli 2003]. Also, this classification is not appropriate or adequate for many hybrid metaheuristics.
- *Memory usage vs. memory-less methods*: Some metaheuristic algorithms use a memory that contains some information extracted online during the search [Taillard *et al.* 2001] such as short-term and long-term memories in tabu search. While other metaheuristics are memoryless; that is, no information extracted dynamically is used during the search. Some examples of this class are local search, GRASP, and simulated annealing.
- *Deterministic vs. stochastic*: A deterministic metaheuristic makes deterministic decisions to solve an optimization problem (e.g., local search, tabu search). In stochastic metaheuristics, some random rules are applied during the search (e.g., simulated annealing, evolutionary algorithms). In deterministic algorithms, using the same initial solution will lead to the same final solution, whereas in stochastic metaheuristics, different final solutions may be obtained from the same initial solution. This characteristic must be taken into account in the performance evaluation of metaheuristic algorithms.
- *Iterative vs. greedy*: Iterative algorithms start with a complete solution (or population of solutions) and apply some search operators on it at each iteration. Whereas greedy algorithms start from an empty solution, and assign a decision variable of the problem at each step until achieving a complete solution. Most of the metaheuristics are iterative algorithms.
- *Single-solution vs. Population-based search*: This is the most commonly used aspect for metaheuristics classifications in the literature [Boussaïd *et al.* 2013]. In single-solution based algorithms (referred to as *trajectory methods*), a single solution is used to explore the search space

which provide more chance to intensify the search in local regions. Whereas in population-based algorithms, a whole population of solutions is evolved during the search process which lead to better diversification in the whole search space. The most common examples of single-solution methods are: basic local search, simulated annealing, tabu search, greedy randomized adaptive search procedure, variable neighborhood search, guided local search, iterated local search and others. Examples for population-based methods include: genetic algorithms, scatter search, ant colony systems, and mimetic algorithms, evolutionary strategies (although some of them are single-solution), particle swarm systems, cultural algorithms, etc. The result of hybridizing single-solution approach with a population based approach is a population-based approach as a mimetic algorithms which incorporate local search into genetic algorithms.

Here, Single-solution vs. Population-based classification is adopted in this context. Moreover, Fig. 3.2 [1], depicts the different aspects used for metaheuristics classifications.

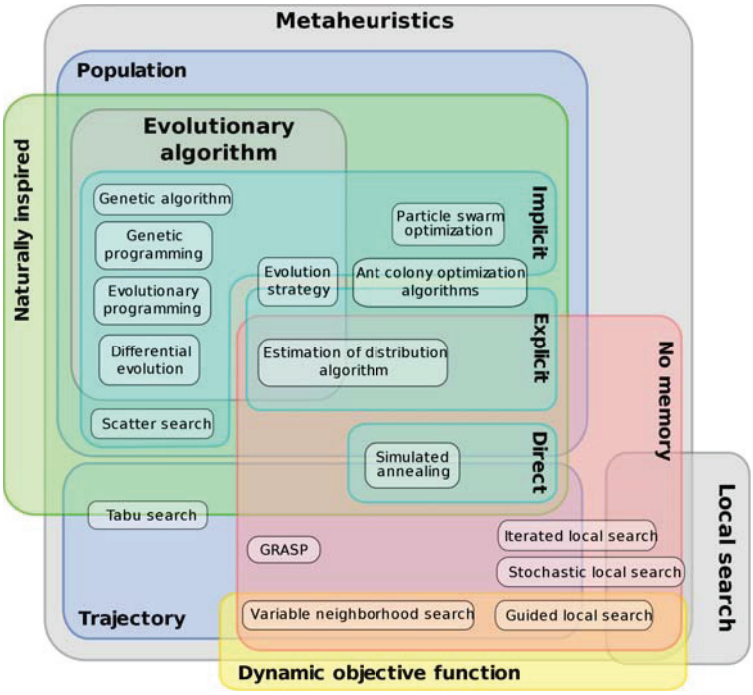


Figure 3.2: The Different classifications of metaheuristics algorithms

[1] http://en.wikipedia.org/wiki/Metaheuristic

3.5 Single-solution based metaheuristics

In this section, single-solution based algorithms are presented. The search process starts by improving a single initial solution which iteratively moves as trajectory in the search space [Crainic & Toulouse 2003]. In the following, we start to present the constructive heuristic algorithm followed by the basic local search procedure. Then, the "intelligent" extensions of local search algorithms which improve its capabilities to escape local optimum are also discussed. These extensions include GRASP, iterated local search, variable neighborhood search, guided local search, simulated annealing and tabu search algorithms.

3.5.1 Greedy Heuristics

A constructive or greedy heuristic [Edmonds 1971] executes a number of iterations to build a complete solution. Starting from an empty solution, greedy algorithms assign a value to one decision variable at a time until constructing a complete solution. Assuming the solution of an optimization problem consists of a set of n components $E = \{e_1, e_2, ..., e_n\}$ where each component e_i represents the i_{th} decision variable. Beginning with an empty solution s, the idea is to use a heuristic to select a new unselected component e_i at each iteration to be included to s according to feasibility conditions until x is completed. Once an element e_i is selected to be part of the solution s, it is never replaced by another element. Alg. 3.1 summarizes the construction process. During the construction the order is maintained static. So, a greedy heuristic is able to generate the same solution every time it is executed on the same problem with the same heuristic. In general, a greedy strategy does not produce an optimal solution in many problems, but nonetheless a greedy heuristic may produce locally optimal solutions that approximate a global optimal solution in a reasonable computation time. However, greedy heuristic can be used as an initial stage. It can be used to generate quite good initial solutions. Then, more intensive search procedures can be used to enhance and upgrade those solutions [Burke *et al.* 1998, Corne & Ross 1996, Kafafy *et al.* 2012a].

3.5.2 Simple Local Search

The basic local search (LS) or *iterative improvement* algorithms starts from an initial candidate solution and attempt to iteratively improve the current solution through exploring its appropriately defined neighborhood [Aarts & Lenstra 1997] (Fig. 3.5 (a)). The neighborhood is formally defined as follows:

Definition 3.1 (Neighborhood structure) [Blum & Roli 2003]: *A neighborhood structure is a function $\mathcal{N} : \mathcal{S} \leftarrow 2^{\mathcal{S}}$ that assigns to every $s \in \mathcal{S}$ a*

Algorithm 3.1 GREEDY HEURISTICS

```
Begin:
s ← ∅;                                   ▷ Begin with an empty solution s
E ← {e_1, e_2, ..., e_n};                ▷ Initialize the set of unselected components E
repeat:
    e_i ←LocalHeuristics(E);             ▷ Use Heuristics to select e_i to be added to s
    if s ∪ e_i is feasible then:         ▷ Check feasibility of adding e_i to s
        s ← s ∪ e_i;                     ▷ Adding e_i to s
        E ← E/{e_i};                     ▷ Delete e_i from E
    end if
until s is completed
End
```

set of neighbors $\mathcal{N}(s) \subseteq \mathcal{S}$. $\mathcal{N}(s)$ is called the neighborhood of s. Often, neighborhood structures are implicitly defined by specifying the changes that must be applied to a solution s in order to generate all its neighbors. The application of such an operator that produces a neighbor $s' \in \mathcal{N}(s)$ of a solution s is commonly called a ***move****.*

Typically, every candidate solution has a neighborhood space that contains more than one neighbor solution. Local search algorithms move from solution to solution in the neighborhood space of candidate solutions by applying local changes, until all candidate neighbors are worse than the current solution, meaning a local search is stuck at a locally minimal[2] point. Termination conditions of local search can also based on CPU time bound, or the maximum number of iterations without improvement reached. The local minimal is the best solution(s) in the defined neighborhood and can be formally defined as follows:

Definition 3.2 (Local Minimum) [Blum & Roli 2003]: *A locally minimal solution (or local minimum) with respect to a neighborhood structure $\mathcal{N}$ is a solution $\hat{s}$ such that $\forall s \in \mathcal{N}(\hat{s}) : f(\hat{s}) \leq f(s)$ (Fig 3.3). We call $\hat{s}$ a strict locally minimal solution if $f(\hat{s}) < f(s)$, $\forall s \in \mathcal{N}(\hat{s})$.*

Algorithm 3.2 LOCAL SEARCH (LS)

```
Begin:
s ←GenerateInitialSolution( );           ▷ By either greedy or random methods
while ∃s' ∈ N(s) such that f(s') < f(s) do:
    s ←ChooseImprovingNeighbor(N(s));    ▷ select a neighbor solution to improve the
    current one
end while
End
```

[2]Without loss of generality, Minimization case is considered

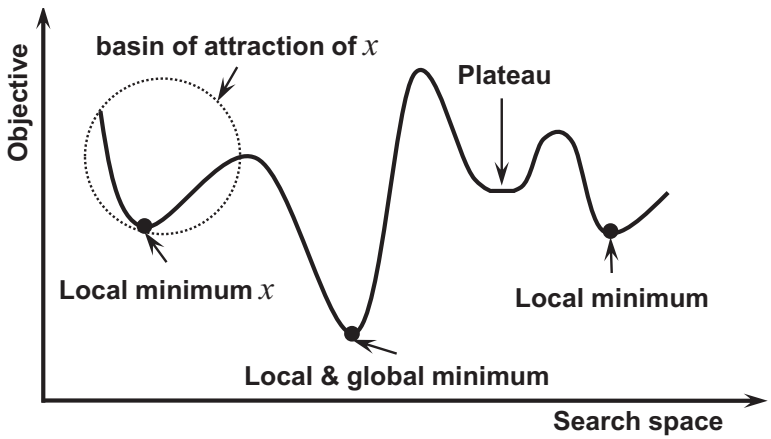


Figure 3.3: Local optimum and global optimum in a search space.

The template of the local search algorithm is described in Alg. 3.2. The initial solution can be generated either using random or greedy approaches. There is always a trade-off between the use of random and greedy initial solutions in terms of the quality of solutions and the computational time. The performance of local search is affected by tow issues. The first one is the choice of the neighborhood structure. A strong neighborhood produces local optima that are largely independent of the quality of the initial solution while a weak neighborhood produces local optima that is highly correlated to the initial solution [Papadimitriou & Steiglitz 1998]. The second issue is how to explore solutions in the neighborhood(s), some of the possible strategies used to select the candidate solutions are explained as follows:

- *Deterministic Iterative Improvement:* In this strategy, each iteration is responsible for generating one neighboring solution. The new generated solution replaces the current solution only if it is better. The search stops when no better neighboring solution is found. This strategy leads to partition the search space $\mathcal{S}$ into so-called *basins of attraction* of local minima [Prestwich & Roli 2005]. The basin of attraction of a local minimum $s' \in \mathcal{S}$ is the set of all solutions s for which the search terminates in s' when started from the initial solution s.
- *Best Iterative improvement:* In this strategy, the neighborhood structure is exhaustively explored to return one of the solutions with the lowest objective function value. Hence, all possible moves are tried for a solution to select the best neighboring solution. This type of exploration may be time-consuming for large neighborhoods.
- *First improvement:* In this strategy, the first improving neighbor that is better than the current solution is selected to improve the current solution.

This means a partial evaluation of the neighborhood. When no improvement is found, a complete evaluation of the neighborhood is performed.

The improvement strategies above can be implemented in the ChooseImprovingNeighbor($\mathcal{N}(s)$) module. The performance of these strategies strongly depends on the definition of the neighborhood structure $\mathcal{N}$. A compromise in terms of quality of solutions and search time may consist in using the first improvement strategy when the initial solution is randomly generated.

Although, simple iterative improvement local search heuristics are simple and easy to implement, they have unsatisfactory performance due to producing local minimum. The quality of the obtained local minimum depends on the initial solution used. To avoid getting stuck in poor local optima, various strategies have been incorporated into local search producing a number of metaheuristics approaches. The main goal of these strategies is to achieve good balance between *intensification* and *diversification* as mentioned. According to [Talbi 2009], these strategies can be classified into different families as in the following as well as depicted in Fig. 3.4:

- *Iterating from different initial solutions*: as applied in multistart local search, iterated local search(ILS), GRASP.
- *Accepting non-improving neighbors*: as applied in simulated annealing (SA) and tabu search (TS). This can be done through allowing moves that deteriorate the current solution to have a chance to escape local optimum.
- *Changing the landscape of the problem*: as applied in both variable neighborhood search strategies (VNS) and guided local search (GLS). In VNS, the neighborhood structure is changed during the search process, whereas in GLS, the objective function or the constraints of the problem on hand is perturbed.

In the following, we present some of the most important local search based metaheuristics.

3.5.3 GRASP

The Greedy Randomized Adaptive Search Procedure (GRASP) is a memory-less multi-start metaheuristic algorithm, which combines a randomized constructive heuristics with local search [Glover & Kochenberger 2003]. It was proposed by Feo and Resende in [Feo & Resende 1989, Feo & Resende 1995]. The main idea of GRASP is based on repeatedly improving starting solutions by local search. Each iteration of the GRASP algorithm consists of two phases: construction

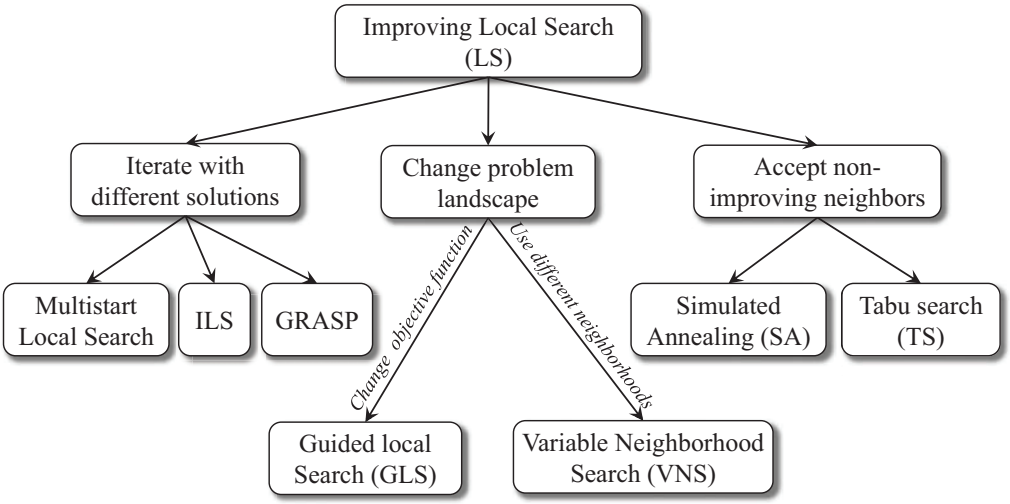


Figure 3.4: Local Search improvement Strategies

and local search. In the construction phase, a feasible completed solution is built using greedy randomized heuristics. Since this solution is not guaranteed to be locally optimal, a local search is performed to improve this solution in the second phase. This process is repeated until stopping criterion is met and the best solution found is taken as a result. Alg. 3.3 illustrates the GRASP procedure. In the following two subsections, each phase is briefly explained.

Algorithm 3.3 GRASP(α)

Inputs:
$\alpha \in (0, 1]$**:** *A parameter for constructing the restricted candidate list RCL*

1: **Begin:**
2: $s^* \leftarrow \varnothing$; ▷ *The best solution found so far*
3: **repeat**:
4: $s \leftarrow \varnothing$; ▷ *Begin with an empty partial solution*
5: $s' \leftarrow$CONSTRUCTION(s, α); ▷ *Apply greedy randomize construction to s*
6: $s \leftarrow$LOCALSEARCH(s'); ▷ *Apply local search to s′*
7: **if** $(F(s) < F(s^*))$ **then**: ▷ *Apply fitness comparison to get the best*
8: $s^* \leftarrow s$; ▷ *Update the best solution*
9: **end if**
10: **until** termination criterion is satisfied
11: **return** s^*; ▷ *Return the best solution*
12: **End**

3.5.3.1 Greedy Randomized Construction

The greedy randomized construction procedure merges randomize to greedy algorithm for producing a divers set of good quality initial solutions, from which to start local search. In this phase, a candidate solution is built iteratively by

incorporating an element into a partial solution at each iteration until a complete solution is built. This means the solution must be defined as a set of elements. At each iteration of the construction phase, a partial solution s (usually empty) is taken. The candidate list (CL) is formed by all the elements which can be included to the partial solution without violating feasibility. The CL is decreasingly ordered according to the values of a greedy function which used to evaluate the benefit of selecting each element. The so-called restricted candidate list (RCL) is formed by the best candidates of the CL. Then, the element to be added to the partial solution is randomly picked from the RCL, which represents the probabilistic aspect of GRASP. The restriction of the RCL may depend on either the number of elements (cardinality-based) or their quality (value-based) that is the most used strategy [Resende & Ribeiro 2003]. Cardinality-based restriction means the RCL list consists of the p best candidate in CL. Whereas, value-based restriction means RCL consists of all elements with values greater than or equal to $c_{min} + \alpha \times (c_{max} - c_{min})$, where p and α are parameters to control the balance between greediness and randomness components of the partial solution, and c_{min} and c_{max} are the values of the best and worst elements, respectively. In practice, the parameter α may be either fixed or adaptive. Algorithm 3.4 summarizes the construction procedure.

Algorithm 3.4 CONSTRUCTION(s, α)

Inputs:
$\alpha \in (0, 1]$**:** *A parameter for constructing the restricted candidate list RCL*

1: **Begin:**
2: **repeat**:
3: $RCL \leftarrow$BUILDRCL(s, α); ▷ *Construct RCL*
4: $e \leftarrow$SELECTELEMENTATRANDOM(RCL); ▷ *Choose an element e to be added to s*
5: $s \leftarrow$ADDELEMENT(e); ▷ *Add element e to s*
6: **until** s becomes a completed solution
7: **return** s;
8: **End**

3.5.3.2 Local Search Phase

After the construction step, the constructed solutions are not guaranteed to be local optima. So, it is useful to apply local search step to improve the constructed solutions. Traditionally, a simple local search algorithm is applied. Nevertheless, more sophisticated local search methods with good global search ability, such as simulated annealing and tabu search, variable neighborhood search etc, can also be used to improve the constructed solutions in GRASP [de la Peña 2004, Resende 2008, Villegas *et al.* 2010]. Path-relinking strategy can also used to improve search capabilities of GRASP [Festa & Resende 2013]. Alg. 3.5 illustrates the general procedure of GRASP local search phase which

applied to improve the constructed solution s.

Algorithm 3.5 LocalSearchPhase(s)

1: **Begin:**
2: $s^* \leftarrow s$; ▷ *set s as initial point to start the search*
3: **while** $(\exists s' \in \mathcal{N}(s^*) : F(s') < F(s^*))$ **do**: ▷ *Improve the current point by neighborhood investigation*
4: $s^* \leftarrow s'$;
5: **end while**
6: **return** s^*;
7: **End**

In general, GRASP performance is highly sensitive to the parameter α, it is important to select its values carefully. Thus, several strategies are proposed to fit α [Talbi 2009], such as initializing to a constant value, dynamically changing according to some probability distribution or automatically adaptation during the search process. As an example, in reactive GRASP [Prais & Ribeiro 2000], α is periodically updated based on the quality of the obtained solutions. However, GRASP can be more effective if the construction procedure samples the most promising regions of the search space, and able to produce solutions belong to basins of attraction of different local optima. More extensive details of GRASP and its applications on combinatorial optimization can be found in[Festa & Resende 2002, Festa & Resende 2009a, Festa & Resende 2009b, Resende & Ribeiro 1997]. Moreover, GRASP adaptations to continuous optimization is also proposed in [Hirsch *et al.* 2007].

3.5.4 Iterated Local Search

Iterated Local search (ILS) is a simple metaheuristic that exploits both local search and a perturbation operator [Glover & Kochenberger 2003]. The idea is instead of repeatedly applying a local search procedure to randomly generated starting solutions, ILS generates the starting solution for the next iteration by perturbing the local optimum found at the current iteration (see Fig. 3.5 (b)). The aim of this idea is that the perturbation mechanism produces a solution that is located in the basin of attraction of a local minimum that is better than and closed to the current solution. The importance of perturbation mechanism due to twofold: firstly, a too weak perturbation may not be sufficient to escape from the basin of attraction of the current local optimum; secondly, a too strong perturbation would make the algorithm similar to a multistart local search with randomly generated starting solutions. Thus, perturbation mechanism must be designed in such a way that achieve this aim. Alg.3.6 summarizes the steps of this metaheuristic.

First, the algorithm starts with an initial solution s and perform local search

Algorithm 3.6 ITERATED LOCAL SEARCH (ILS)

```
1: Begin:
2: s ←GenerateInitialSolution( );                    ▷ Internalize the current solution
3: s′ ←LocalSearch(s);
4: repeat:
5:     p ←Perturbation(s′,history);                  ▷ Perturb the obtained local optimum s′
6:     ŝ ←LocalSearch(p);
7:     s′ ←ApplyAcceptanceCriterion(ŝ,s′,history);
8: until termination condition is met
9: End
```

to improve s until a local optimum s' is found. Then, the current solution s' is perturbed to obtain the solution p and a different local optimum $\hat{s}$ is obtained by performing local search on p. Finally, the resulting solution $\hat{s}$ may either be accepted as new current solution, or not according to the function ApplyAcceptanceCriterion($\hat{s}, s'$),*history* which implements the acceptance criterion. The balance between intensification and diversification can be controlled by both the acceptance criterion and the perturbation mechanism. For example, an extreme acceptance criterion in terms of intensification is to accept only improving solutions. Another extreme criterion in terms of diversification is to accept any solution, regardless its quality. Many acceptance criteria that balance the two goals may be applied [Talbi 2009]. A recent review of ILS, its extensions and its applications is found in [Lourenço *et al.* 2010]. Also, adaptive perturbation strategy is found in[Benlic & Hao 2013].

3.5.5 Variable Neighborhood Search

The variable neighborhood search (VNS) metaheuristic is introduced in [Mladenović & Hansen 1997]. VNS adopts the strategy of using more than one neighborhood structure during the search process in order to escape local minimum (see Fig. 3.5(d)). This is based on the fact that a local minimum with respect to a certain neighborhood function $\mathcal{N}_1$ is not necessarily a local minimum with respect to a different neighborhood function $\mathcal{N}_2$. To develop such idea, VNS has to define a set of neighborhood structures, and to dynamically swap between these different neighborhood structures during the search process. Alg. 3.7 summarizes the basic steps of VNS metaheuristic.

At the initialization step, a set of neighborhood structures $\mathcal{N}_k$ ($\forall k = 1, ..., k_{max}$) are defined. Then, the current solution s is initialized. Each iteration of the algorithm involves three basic steps: *shaking*, *local search*, and *move*. At each iteration, the shaking step is carried out through randomly selecting the solution s' form the kth neighborhood structure of the current solution $\mathcal{N}_k(s)$. Then, local search procedure is applied to improve the solution s', for producing the solution

Algorithm 3.7 VARIABLE NEIGHBORHOOD SEARCH (VNS)

```
Begin:
Define a set of neighborhood structures N_k, ∀k = 1, ..., k_max
s ←GenerateInitialSolution( );                    ▷ Initialize the current solution
repeat:
    k ← 1;                                        ▷ Intialize with 1_st neighborhood structure
    while k < k_max do:                           ▷ Inner Loop
        s' ←PickAtRandom(N_k(s));                 ▷ Shaking phase
        s'' ←LocalSearch(s');                     ▷ Local search phase
        if f(s'') < f(s) then:                    ▷ if s'' better than s
            s ← s'';                              ▷ Update the current solution
            k ← 1;
        else:
            k ← k + 1;                            ▷ Update neighborhood structure
        end if
    end while
until termination condition is met
End
```

s''. The local search can use any neighborhood structure and is not restricted to the set $\mathcal{N}_k$, $\forall k = 1, ..., k_{max}$. After local search step, in case of the new local optima s'' is better than the current solution s, Then, s'' replaces s and the same search procedure is thus restarted from the solution s'' in the first neighborhood $\mathcal{N}_1$. Otherwise, the algorithm moves to the next neighborhood $\mathcal{N}_{k+1}$ and a new shaking phase starts by randomly generating a new solution in this neighborhood and it attempts to improve it using local search. The process of changing neighborhoods in case of no improvements corresponds to a diversification of the search. For recent surveys of VNS extensions and its adaptations to tackle continuous optimization problems, we refer to [Hansen *et al.* 2008, Hansen *et al.* 2010] and [Liberti & Drazic 2005, Mladenović *et al.* 2008, Carrizosa *et al.* 2012]. Also, hybridization of VNS with other metaheuristics, such as GRASP, can also be found in [Villegas *et al.* 2010, Salehipour *et al.* 2011]

3.5.6 Guided Local Search

Guided local search (GLS) [Voudouris 1997, Voudouris & Tsang 1999] is a metaheuristic which adopts the strategy of modifying the search landscape through dynamically changing the objective function (see Fig. 3.5(c)). The idea of using modified objective functions in GLS is to escape from the local optimal by gradually reducing its attractiveness. Changing objective functions dynamically can be achieved by penalizing solution features that occur frequently in visited solutions. The penalized features leads to increasing the objective function of solutions that contain these features. This cause changing the search landscape and thus gradually reducing the attractiveness of the current local minimum.

As described in Alg. 3.8, GLS starts with an initial solution s that is improved by local search until a local optima s^* is found. Then, in each iteration the original objective function $f(s)$ is adapted to obtain the so-called "*augmented*" objective function $f'(s)$. Then, the local search is restarted using the augmented function, which is designed to bring the search out of the local optimum. First, a set of features ft_k, $k = 1, ..., n$ has to be defined to distinguish between solutions with different characteristics, so that regions of similarity around local minimum can be recognized and avoided. For each feature ft_i a cost function c_i is defined. Each feature is also associated with a penalty p_i to record the number of occurrences of the feature in local minimum. The penalties are initialized to 0 and updated when the local search reaches a local minimum. Given an objective function f and a solution s, the augmented function $f'(s)$ is given by:

$$f'(s) = f(s) + \lambda \sum_{i=1}^{n} p_i I_i(s) \tag{3.1}$$

where λ represents the regularization parameter and, $I_i(s)$ is an indicator function that indicates whether the feature ft_i is present in the solution s:

$$I_i(s) = \begin{cases} 1 & \text{if } ft_i \in s \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

GLS uses the augmented function above to allow it to guide the Local Search algorithm out of the local minimum, through penalizing features present in that local minimum. Each time a local optimum is found by the local search, GLS intends to penalize the most "unfavorable features" of this local optimum. The idea is to make the local minimum more costly than the surrounding search space, where these features are not present. This way, solutions exhibiting other features become more attractive, and the search can escape from the local minimum. When a local optimum s^* is reached, the utility u_i of penalizing a feature ft_i is calculated as follows:

$$u_i(s^*) = I_i(s^*) \frac{c_i}{1 + p_i} \tag{3.3}$$

where c_i is the cost of the feature ft_i. This means, if a given feature ft_i is not present in the local minimum s^*, the utility of penalizing feature ft_i is 0. The higher the cost c_i of this feature, the greater the utility of penalizing it. Besides, the more times it has been penalized (the greater p_i), the lower the utility of penalizing it again. Then, the feature with the highest utility will be penalized by increasing its penalty p_i by 1, and the scaling of the penalty is normalized by λ. The balance between intensification an diversification is controlled by λ. Large values of λ encourage diversification, while small values

intensify the search around the local optimum [Talbi 2009]. More details about GLS adaptation for continuous optimization and its extensions to population based metaheuristic can be found in [Voudouris 1998] and [Zhang *et al.* 2003, Tairan & Zhang 2010]. Other GLS extensions to Multiobjective optimization can be found in [Alsheddy 2011].

Algorithm 3.8 GUIDED LOCAL SEARCH (GLS)

```
Begin:
{ft_i | i = 1, ..., n} ←DefineFeatures( );                ▷ Select a set of features
s ←GenerateInitialSolution( );                            ▷ Initialize the current solution
p_i ← 0; ∀i = 1, ..., n                                   ▷ Initialize penalties
repeat:
    f'(s) ←AugmentedFunction(s, λ);                       ▷ Compute f' as in Equ. (3.1)
    s* ←LocalSearch(s, f');                               ▷ Improve s using the augmented f'
    u_i(s*) ←ComputeUtilizes(s*, i); ∀i = 1, ..., n       ▷ Compute u_i as in Equ. (3.3)
    for each feature ft_i with maximum u_i(s*) do:        ▷ penalize features with max. util.
        p_i ← p_i + 1;
    end for
until the stopping condition is satisfied
End
```

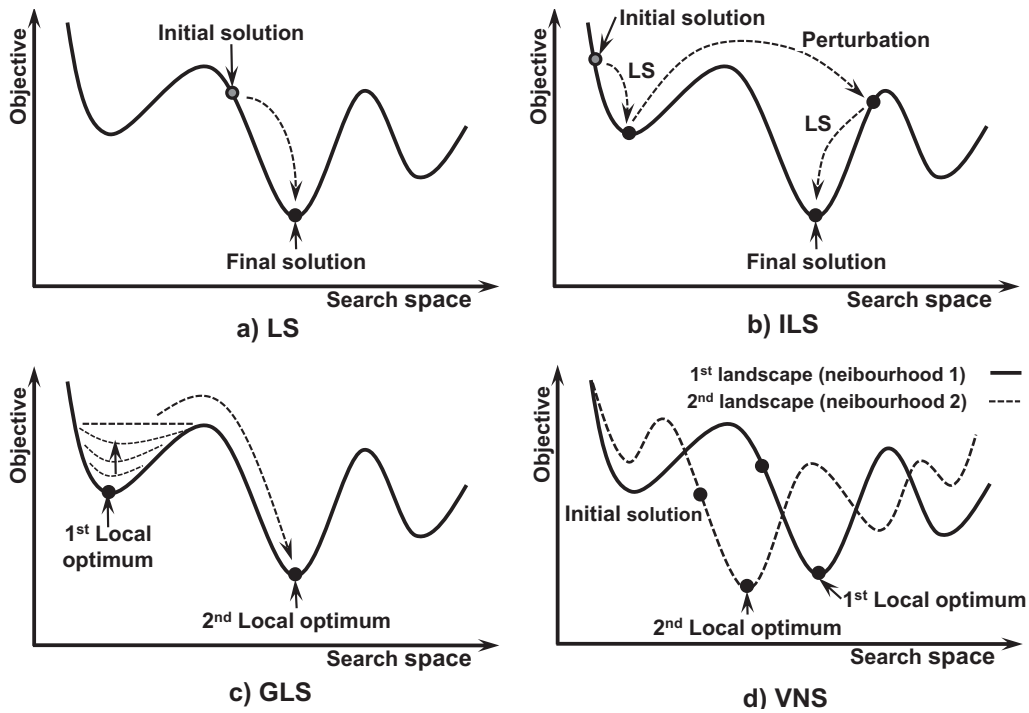


Figure 3.5: Local Search and its alternatives

3.5.7 Simulated Annealing

Simulated annealing (SA) is a metaheuristic that explores new areas of the solution space by allowing the degradation of a solution under some conditions, in order to escape from local optimum. The idea of SA is inspired by the annealing process of metal and glass, which assumes a low energy configuration when first heated up and then cooled down sufficiently slowly. SA transposes the process of the annealing to the solution of an optimization problem: the objective function of the problem, similar to the energy of a material, is then minimized, by introducing a dummy temperature T, which is a simple controllable parameter of the algorithm. SA was first presented as a search algorithm for CO problems in [Kirkpatrick *et al.* 1983] and [Černỳ 1985]. The main principle is that improving candidate solutions are always accepted while non-improving candidate solutions are accepted with a certain probability. This probability of accepting non-improving solutions is calculated according to the current temperature of the algorithm. The search process is stared with a high initial temperature T, which means a high probability for accepting non-improving solutions. The temperature is gradually decreased during the search process, thus, the probability to accept non-improving solutions is also reduced. The algorithm accepts only improving solutions when temperature goes to zero until the stopping condition.

Simulated annealing framework is described in Alg.3.9. Firstly, an initial solution s is generated either randomly or using constructive heuristic. The temperature T is also initialized. At each iteration k, a solution s' is randomly selected from the neighborhood $\mathcal{N}(s)$ of the current solution s. The generated neighbor s' is always accepted as new current solution if it improves the objective function. Otherwise, s is accepted with a probability p which is a function of both the current temperature T_k and the amount of degradation of the objective function $(f(s') - f(s))$. In general, this probability is computed based on the Boltzmann distribution:

$$p(s'|T_k, s), f(s)) = e^{-\frac{f(s')-f(s)}{T_k}} \tag{3.4}$$

The current temperature T_k is adapted at each iteration according to the so-called *cooling schedule* that defines the value of T_k at each iteration k, $T_{k+1} = Q(T_k, k)$, where $Q(T_k, k)$ is a function of current temperature and the iteration number. The selection of an adequate cooling schedule is crucial for the performance and highly dependent on the problem domain. One of the most used cooling schedules follows the geometric low $T_{k+1} = \alpha T_k$, where $\alpha \in (0, 1)$ is the cooling factor, which correspondence to exponential decay of the temperature. The cooling rule may be varied during the search for tuning the balance between intensification and diversification. In other words, the search process may start with linearly decreasing T in order to sample the search space. Then, T might follow a rule

such as the geometric one to converge to local minimum at the end of the search.

SA has been successfully applied to several discrete or continuous optimization problems. The adaptation of SA to continuous optimization problems has been particularly studied [Courat *et al.* 1994]. A wide bibliography can be found in [Fleischer 1995, Beyer & Schwefel 2002, Alba 2005]. Several interesting variants of SA have been proposed in the literature as Threshold Accepting [Dueck & Scheuer 1990] and Extremal Optimization [Boettcher & Percus 2002] etc.

Algorithm 3.9 SIMULATED ANNEALING (SA)

1: **Begin:**
2: $s \leftarrow$GENERATEINITIALSOLUTION( ); ▷ *Initialize s the current solution*
3: $k \leftarrow 0$; ▷ *Initialize the iteration counter*
4: $T_k \leftarrow$SETINITIALTEMPERATURE( ); ▷ *Specify initial temperature*
5: **while** termination conditions not met **do**:
6: $s' \leftarrow$PICKNEIGHBORATRANDOM($\mathcal{N}(s)$); ▷ *Randomly select a neighbor solution*
7: **if** $(f(s') < f(s))$ **then**:
8: $s \leftarrow s'$; ▷ *Update current solution*
9: **else**:
10: Accept s' as new solution with probability $p(s'|T_k, s) = e^{-\frac{f(s')-f(s)}{T_k}}$
11: **end if**
12: $k \leftarrow k + 1$ ▷ *Update iteration counter*
13: $T_k \leftarrow$ADAPTTEMPERATURE( ); ▷ *Adapt temperature using cooling schedule*
14: **end while**
15: **End**

3.5.8 Tabu Search

Tabu search (TS) is a metaheuristic that exploits the search history through using flexible and adaptive memory structures, both to escape from local minimum and to implement an exploitative strategy. The use of memory, which stores information related to the search process, represents the particular feature of tabu search. The basic ideas of Tabu search were first introduced in [Glover 1986] and the algorithm was proposed in [Glover 1989]. Basically, Tabu Search behaves like a deterministic local search strategy where, at each iteration, the best solution in the neighborhood of the current solution is selected as the new current solution, but it accepts non-improving solutions to escape from local optima when all neighbors are non-improving solutions. Various types of memory structures are commonly used to remember specific properties of the trajectory through the search space that the algorithm has undertaken. A short-term memory, known as the *tabu list*, stores recently visited solutions (or attributes of recently visited solutions) to forbid revisiting these solutions and therefore to prevent from endless cycling and forces the search to accept even deteriorating moves. Long-term

memory is used to collect information during the overall search process that permits the identification of common properties in good visited solutions and also to attempt to visit solutions with varying properties from those already visited. In the following, the basic framework of tabu search procedure is described in Alg. 3.10.

Firstly, the algorithm uses the tabu lists which store the features of recently visited solutions. A lot of tabu lists may be used corresponding to the different types of considered features. At the beginning of the search process, an initial solution s is generated and the tabu lists (TL_1,...,TL_k) are initialized as empty sets. To apply a move, the neighborhood of the current solution $\mathcal{N}(s)$ is checked to exclude those neighbor solutions which contain solution features currently stored in the tabu lists. These solutions are said to violate tabu conditions. Now, we have a restricted set of neighbors $\mathcal{N}a(s)$. It must be noted that storing only features of solutions may allow excluding some of high quality unvisited solutions. This problem is tackled by defining aspiration criteria which allow to include a solution in the restricted set of neighbors even though it violates a tabu condition. At each iteration, the best solution s' is chosen from $\mathcal{N}a(s)$ as the new current solution. Moreover, the corresponding features of this solution are added to the tabu lists with removing the oldest solution features if the tabu lists have reached their maximally allowed sizes. Finally, the algorithm stops when a termination condition is met.

Algorithm 3.10 Tabu Search (TS)

Begin:
$s \leftarrow$GenerateInitialSolution( );
InitializeTabuLists(TL_1,...,TL_k);
repeat:
  $\mathcal{N}_a(s) \leftarrow \{s' \in \mathcal{N}(s) | s'$ does not violate a tabu condition, or it satisfies at least one
  aspiration condition $\}$
  $s' \leftarrow argmin\{f(s'') | s'' \in \mathcal{N}_a(s)\}$
  UpdateTabuLists(TL_1,...,TL_k,s,s');
  $s \leftarrow s'$
until termination condition is met
End

In tabu search, the search process is affected by the length of the tabu list which controls the memory of algorithm during the search process. The small length of the tabu list leads to concentrating the search on small area in the search space. While large length prevents revisiting a large number of solutions, and thus forces the search to explore large regions in the search space. The tabu list length can also be changed during the search, leading to more robust algorithms, like reactive tabu search [Battiti & Tecchiolli 1994]

An extensive description of TS and its concepts can be found in

[Glover & Laguna 1997]. TS was designed for, and has predominately been applied to combinatorial optimization problems. However, adaptations of TS to continuous optimization problems have been proposed [Cvijovic & Klinowski 1995, Chelouah & Siarry 2000]. The interested reader can find representative applications of TS in [Taillard 1991, Battiti & Protasi 1996].

3.6 Population-based metaheuristics

In Population-based metaheuristics algorithms, a population of solutions is adopted rather than operating on single solution. The most commonly used population based algorithms are evolutionary computations (EC) based algorithms and swarm intelligence(SI) based algorithms. EC algorithms simulate the natural evolution process which are based on the Darwinian concept of "Survival of the Fittest". They tackle hard optimization problems through improving a population of initial solutions by using selection, recombination and mutation operators. In SI algorithms, the idea is to produce computational intelligence by exploiting simple analogs of social interaction, rather than purely individual cognitive abilities.

3.6.1 Evolutionary computations

Evolutionary Computation (EC) is the general term expresses several optimization algorithms that are inspired by the Darwinian principles of natural selection. EC Algorithms are usually called evolutionary algorithms (EA) as they are based on the evolutionary principles in solving optimization problems. EAs involve the domains of genetic algorithms (GA) [Holland 1975], evolutionary strategies (ES) [Rechenberg 1973], evolutionary programming (EP) [Fogel *et al.* 1966], and genetic programming (GP) [Koza 1992] which share the same principle of producing better solution by simulating the evolution of individual structures via processes of selection, recombination, and mutation reproduction. In general, EAs operates on an initial population of solutions to the problem on hand. At each algorithm iteration, a number of Evolutionary operators is applied to the individuals of the current population to generate the individuals of the population of the next generation. the new offspring is generated by applying both recombination (crossover) on the parent individuals, then mutation operator is also applied to cause a self-adaptation of the offspring to promote diversity. The new offspring is evaluated to identify their fitness. Finally, the new population is formed by selection from both the parents individuals and the new generated offspring according to their fitness which is matched with the principle "Survival of the Fittest" in nature evolution. This process is continued until stopping condition is satisfied. The general framework of EA is described in Alg.3.11. In this

algorithm, P denotes the population of individuals. A population of offspring (P'') is generated by the application of recombination and mutation operators and the individuals for the next population are selected from the union of the old population P and the offspring population P''. EAs have more research

Algorithm 3.11 EVOLUTIONARY COMPUTATION (EC)

```
Begin:
P ←GenerateInitialPopulation( );
P ←EvaluateFitness(P);
while termination condition not met do
    P' ←SelectParents(P);                          ▷ Select parents individuals
    P'' ←Recombination(P');                        ▷ Generate new offspring from P'
    P'' ←Mutation(P'');                            ▷ Apply mutation to P''
    P'' ←EvaluateFitness(P'');                     ▷ Evaluate the offspring population
    P ←SelectNextPop(P ∪ P'');                     ▷ Select the next population from union of P ∪ P''
end while
End
```

attention over the last decades. Many surveys about EAs and its successful application in combinatorial optimization problems are found in [Bäck 1996] and [Bianchi *et al.* 2009]. There are also many successful applications in constrained optimization [Coello Coello 2002] and multibjective optimization which are one of the current trends in developing EAs [Coello *et al.* 2007]. In the following sections, a brief discussion on both genetic algorithms and evolutionary strategies is involved.

3.6.1.1 Genetic Algorithms

Genetic algorithms (GA) [Holland 1975] are considered as the most well known evolutionary optimization techniques. The typical Genetic Algorithm (GA) consists of a number of steps on its way to finding the preferred solutions which include the characteristic procedures that an EA runs through [Coello 2000a, Deb 2001, Fonseca & Fleming 1995, Goldberg 1989, Michalewicz 1996]. These steps can be summarized as follows:

- Representing (encoding) solutions of the problem on hand as a population of individuals.
- A way to initialize the population of individuals.
- A fitness function which returns a rating for each individual.
- Evolutionary operators (selection, crossover and mutation) that may be applied to parents individuals to create offspring.
- Termination criteria for the algorithm.

Here, the basic steps of GA are described in more details as follows.

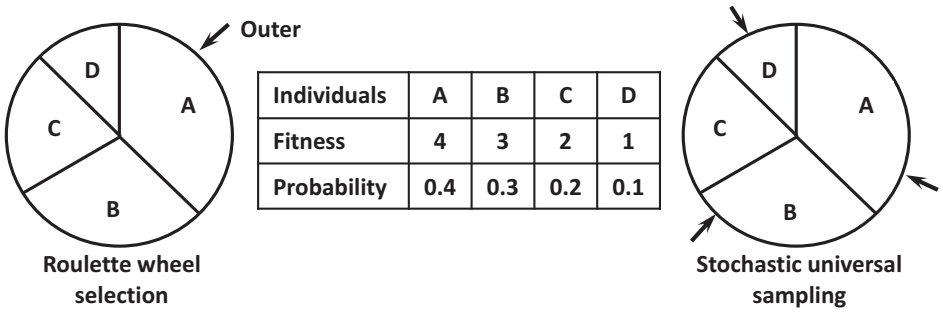
1. Encoding solutions and initializing population: GAs needs a way to represent information in the individual. According to the tackled problem, there are several ways for representation. The most commonly used representation techniques are binary encoding in which the individual is represented by string bits, 0 or 1, floating point encoding in which the real values is used directly to represent the decision variables and permutation encoding in which a sequence of numbers is used to represent the decision variables. After choosing the encoding method, a population of solutions must be initialized to operate on it. The initial population should be as diverse as possible to cover the different regions in the search space. The most common ways to initialize populations is random generation or using the constructive heuristics. There are no hard rules for determining the size of the population. Generally, larger populations guarantee greater diversity, but they use more computer resources.

2. Fitness evaluation: In this step, the quality of an individual x, is measured by a fitness function, $f(x)$. The fitness value of each individual is calculated by the fitness function, and is not necessarily equal to the objective value. This step plays the most important role for selection in the next step. The constraints can also be handled in this step by using penalty functions to penalize potential solutions that are infeasible by degrading their fitness values.

3. Selection: The selection operator is intended to improve the average quality of the population by giving individuals of higher quality a higher probability of survival. Thereby, selection acts as an intensification strategy through focusing the search on promising regions in the search space. The objective of the selection method is to assign individuals with the largest fitness a higher probability of reproduction. There are different kinds of selection operators, e.g., Tournament and Roulette Wheel Selection [Goldberg 1989, Deb 2001]. In Tournament Selection, k individuals are chosen at random and the best one is selected to survive into the next generation, whereas in Roulette Wheel Selection, each population member is assigned a proportion of the Roulette Wheel equal to the ratio of its fitness to the sum of the entire population's fitness as depicted in Fig.3.6. To reduce the bias of the roulette selection strategy, the stochastic universal sampling (SUS) may be used. An outer roulette wheel is placed around the pie with k equally spaced pointers. In the SUS strategy, a single spin of the roulette wheel will simultaneously select all the k individuals for reproduction.

Depending on the problems and their restrictions, certain characteristics can be applied on the selection process. If elitism is implemented, some high quality individuals will be selected to survive from one generation to the next one. If the selection operator uses much selection pressure, i.e., it emphasizes the population's best solution by assigning many copies of it, the algorithm will lose the

diversity of its solutions very quickly. On the other hand, if the selection pressure is very low, the method will behave like a random search. The handling of the constraints can also be carried out in the selection step [Deb 2001]. This method compares two selected individuals, if both of them are feasible, the individual with "better" fitness value is selected. In the case that one of them is infeasible, the feasible individual is selected and when both of them are infeasible, the individual with smaller overall constraint violation is chosen.



Individuals	A	B	C	D
Fitness	4	3	2	1
Probability	0.4	0.3	0.2	0.1

Figure 3.6: Roulette Wheel Selection vs. stochastic universal sampling

4. Crossover: Crossover is a process in which new individuals are generated by exchanging features of the selected parents to producing new offspring that explore new points in the search space, with the intent of improving the fitness of the next generation. The primary exploitation operator in genetic algorithms is crossover, a version of artificial mating. There are various types of crossover operators, depending on the encoding of individuals, the problem, the search space, and many other parameters. For example, the crossover operators for string (bit-string or integer value) individuals are: single point, multiple point, and uniform [Michalewicz 1996], whereas, for real-valued individuals the crossover operators such as: the average crossover[Nomura 1997], the heuristic crossover [Wright *et al.* 1991], the flat crossover [Radcliffe 1991], the blend crossover (BLX-α) [Eshelman & Schaffer 1993], the intermediate crossover [Voigt *et al.* 1995], the simulated binary crossover(SBX) [Deb & Agrawal 1995], the uni-modal distribution crossover (UNDX) [Ono *et al.* 2003], the simplex crossover(SPX) [Tsutsui *et al.* 1999a] and the parent centric crossover(PCX) [Deb *et al.* 2002]. Indeed, the main idea in utilizing crossover is examining the search space locally. This is particularly achieved for bit-string coding of individuals. The crossover operator is usually applied on two randomly selected individuals with a probability of PC [Spears 2000].

- Single point crossover: This is achieved by randomly choosing a crossing point along the string. All the string elements (bits) following the crossing point are exchanged. Fig.3.7 shows this type applied on bit-string.

- n-point crossover: This operator functions similarly to the single point cross-over operator, except that n crossover points are randomly chosen. The segments of the string between the n points are exchanged. Fig.3.7 depicts this type applied on permutation string with $n = 2$.

- Uniform crossover: This operator is the extreme version of the previous operators. It randomly chooses different bits from each of the parents, with equal probabilities.

- Simulated binary crossover (SBX): This operator is introduced by Deb et al. [Deb 2001] for real vector individuals. Indeed, SBX simulates the single point crossover on real vectors.

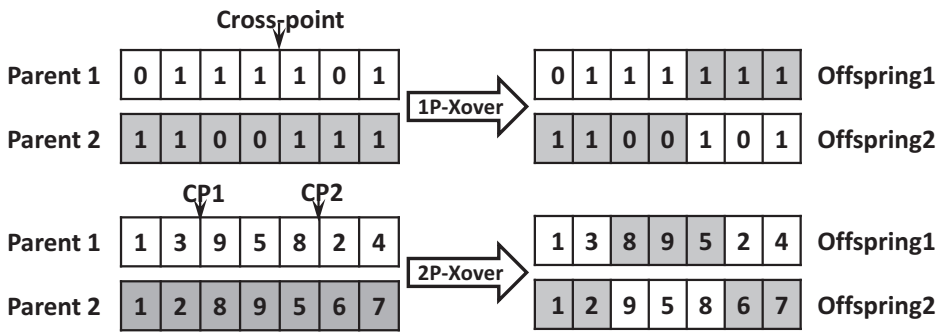


Figure 3.7: One and two points crossover for binary and permutation strings

5. Mutation: Mutation operators are applied on the individuals of a population with a certain probability P_m. There are different mutation operators depending on the encoding of the individuals [Spears 2000]. For bit-string individuals, Mutation changes a bit from 0 to 1 and vice versa, whereas in real individuals a mutated variable is replaced by a new random variable as depicted in Fig.3.8. There are many mutation operators reported in the literature, such as uniform, non-uniform, whole non-uniform, boundary mutation [Michalewicz 1996], and polynomial mutation [Deb & Agrawal 1995]. The aim of mutation is to apply a sudden large change on the individuals for performing global searching.

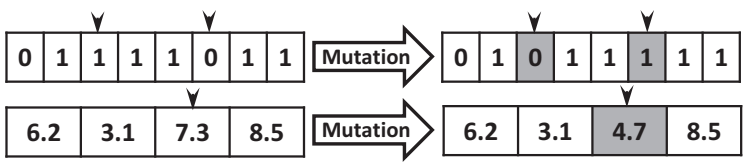


Figure 3.8: Mutation on binary and real value encoding

6. Termination: Generally in EAs, the algorithm still run until satisfying a stopping condition. Termination criteria can be defined by a maximum number of generations (or function evaluations) or when there has been no change in the population for a given number of generations. In this case, reproduction operators will have no further effect.

The general GA structure is described in Alg.3.12. Firstly, GA encodes and initializes a population of individuals to start the search process. Each individual in the initial population is evaluated through computing its fitness. At each iteration, the genetic operators is applied to the current population for generating new individuals. First, the parent individuals are chosen by selection operator. Then, the new generated individuals are created by applying crossover in which two or more selected parent individuals are recombined to produce one or more new individuals (offspring). Mutation operator is also applied to make a self-adaptation of the offspring. The resulting offspring are evaluated and a suitable selection strategy is then applied to determine which individuals will be maintained into the next generation. This evolutionary process is continued until a termination condition is satisfied. Many variants of GAs have been devel-

Algorithm 3.12 Genetic Algorithm (GA)

1: **Begin:**
2: $P \leftarrow$EncodeAndInitializePopulation( );
3: $P \leftarrow$FitnessEvaluation( );
4: **repeat**:
5: $Parents \leftarrow$Selection(P);
6: $P' \leftarrow$Crossover($Parents$);
7: $P'' \leftarrow$Mutation(P');
8: $P'' \leftarrow$FitnessEvaluation( );
9: $P \leftarrow$SelectTheFittest(P,P'');
10: **until** termination condition is satisfied
11: **End**

oped and applied to a wide range of optimization problems including hybrid GAs [Sinha & Glodberg 2003] and multiobjective optimization [Konak *et al.* 2006].

3.6.1.2 Evolutionary Strategies

Like genetic algorithms, Evolutionary Strategies (ES) are a subclass of Evolutionary Algorithms which are originally developed in [Rechenberg 1965, Rechenberg 1973]. While genetic algorithms emphasize recombination (high crossover probability) as the main search mechanism and usually use self-adaptation (low mutation probability) only as a supportive mechanism, evolutionary strategies emphasize both mechanisms as fundamental for searching. In most cases, ESs are applied to tackle continuous optimization where real-values are used to represent individuals. ESs operate on two distinct postulations, one

for the parents individuals and the other for the generated offspring. They usually depends specific Gaussian distributed mutation and sometimes Crossover to generate of offspring from parents population. An elitist replacement is then also used to keep the best individuals to be the parents for the next generation. The basic notation $(\mu + \lambda)$ES where μ is the number of parents and λ is the number of offspring (usually $\lambda \geq \mu$), represents an evolutionary strategy that in each generation selects the best μ individuals from the $\mu + \lambda$ individuals (parents and offspring) in total. The modified notation (μ, λ)ES indicates that λ offspring are generated from the μ parents but the best μ individuals are selected only from the λ offspring. Other two well-known ES notation are known as $(\mu/\rho + \lambda)$ and $(\mu/\rho, \lambda)$ in which the parameter ρ refers to the number of parents involved in the procreation of one offspring. The genral frame work of ES is described in Alg.3.13

As mentioned, the mutation in ES follows normal distribution with zero mean and standard deviation r, which controls the mutation step size. The parameter r can be kept constant or dynamically adjusted by assigning different values depending on the number of generations or by incorporating feedback from the search process.

Algorithm 3.13 EVOLUTIONARY STRATEGIES (ES)

Inputs: μ: *number of parents*, λ: *number of offspring*

1: **Begin:**
2: $P \leftarrow$ INITIALIZEPOPULATION(μ); ▷ *Generate μ parent individuals*
3: $P \leftarrow$ EVALUATION(P); ▷ *Evaluate the μ individuals*
4: **while** stopping criterion not met **do**:
5: $P' \leftarrow$ GENERATEOFFSPRING(P, λ); ▷ *Generate λ offspring from μ parents*
6: $P' \leftarrow$ EVALUATION(P'); ▷ *Evaluate the λ individuals*
7: $P \leftarrow$ ELITISM(P, P'); ▷ *Choose the best μ individuals from parents and offspring*
8: **end while**
9: **End**

More details about theoretical investigation of evolutionary search strategies on a variety of problem classes that have contributed can be found in [Beyer & Schwefel 2002, Kramer 2010].

3.6.2 Other Evolutionary Algorithms

In this section, some of the others evolutionary algorithm are described. these algorithms includes differential evolution, scatter search, path relinking and mimetic algorithms.

3.6.2.1 Differential Evolution

Differential evolution (DE) is considered as one of the most powerful approaches for tackling continuous global optimization [Price *et al.* 2005]. DE adopts the concept of a larger population from GA beside self-adapting mutation from ES [Storn & Price 1995, Storn & Price 1997]. Since It was proposed to solve Chebycheff polynomial fitting problem [Storn & Price 1997], DE has become a very reliable strategy for many different optimization tasks. The basic idea of DE is based on the so-called *differential mutation* which refers to using vector differences for perturbing the vector population. The differential mutation has been integrated in a novel recombination operator of two or more solutions in order to direct the search towards high quality solutions.

As any evolutionary algorithm, DE operates on a population of candidate solutions for the optimization problem to be solved. The initial population is randomly initialized in the beginning of the evaluation process. At each iteration, the recombination operator which composed of differential mutation followed by crossover is applied for generating new individuals which are evaluated to determine their fitness. Each individual (*target individual*) of the population competes against a new generated individual (*trial individual*) to determine which one will be maintained into the next generation. The trial individual is produced by recombining a target individual with the so-called "*mutant individual*" which is created by applying the differential mutation. In order to explain this process in details, Assume $P = \{x_1, x_2, ..., x_N\}$ is a population of N target individuals where each ith individual $x_i = (x_i^1, x_i^2, ..., x_i^n)$ consists of n components, $\forall i \in \{1, ..., N\}$. For each target individual $x_i \in P$, a new trial individual u_i is generated through applying the differential mutation on a number of selected distinct target individuals, followed by crossover as explained in the following subsections:

Differential mutation

The simplest form of this operation is that a mutant individual v_i for each target individual $x_i \in P$ is generated by multiplying the amplification factor F by the difference of two random individuals (x_{r2}, x_{r3}), and the result is added to another third random individual x_{r1}, as shown below:

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3}) \tag{3.5}$$

where r_1, r_2, $r_3 \in \{1, 2, ..., N\}$are random indexes such that $r_1 \neq r_2 \neq r_3 \neq i$. The scaling factor $F \in [0, 1]$ is a control parameter for scaling the difference vector. This operation enables DE to explore the search space and maintain diversity. There are many strategies for implementing differential mutation according to which individual to be mutated and the number of difference vector

used. The general mutation strategy denoted as "DE/x/y", where x represents the individual to be mutated, y denotes the number of difference vectors used. For instance, DE/rand/1 [Storn 1996], DE/best/1 [Storn 1996], DE/rand-to-best/2 [Qin *et al.* 2009], rand/2/dir [Mezura-Montes *et al.* 2006], DE/current-to-rand/1[Iorio & Li 2004], DE/current-to- best/1 [Price *et al.* 2005]. There are also other mutation strategies, such as best/n/bin, rand/n/bin, best/n/exp, rand/n/exp, "current-to-rand/n", and "rand-to-current/n", where n can be equal to any integer value [Price *et al.* 2005].

Crossover

After producing the mutant individual v_i corresponding to the target individual x_i, Crossover operator is applied on both to yield the trial individual u_i. The two different schemes to implement crossover are exponential and binary crossover as described in the following paragraphs. The DE family of algorithms commonly uses one of those schemes although it is known that binomial is generally better than exponential.

Exponential crossover: In this crossover scheme, a random index $l \in \{1, ..., n\}$ is selected, where n represents the individual length. This index acts as a starting point in the target individual, from where the crossover or exchange of components with the donor individual starts. Another integer $L \in \{1, ..., n\}$ is also chosen to represent the number of components that the donor individual actually contributes to the target. After the generation of l and L, the trial individual is obtained as:

$$u_i^j = \begin{cases} v_i^j & \text{for } j = \langle l \rangle_n, \langle l+1 \rangle_n, ..., \langle l+L-1 \rangle_n \\ x_i^j & \text{for all other } j \in \{1, ..., n\} \end{cases} \tag{3.6}$$

where $i \in \{1, 2, ..., N\}$, is individuals' index, $j \in \{1, 2, ..., n\}$, is components' index, the angular brackets $\langle l \rangle_n$ denote a modulo function with modulus n, with a starting index l.

Binomial crossover: In this crossover scheme, for each component $j \in \{1, ..., n\}$, a random number $rand(j) \in [0, 1]$ is generated. The jth component is exchanged if the random number $rand(j)$ is less than or equal to a crossover rate (CR). In this case, the number of components inherited from the donor individual has a (nearly) binomial distribution

$$u_i^j = \begin{cases} v_i^j & \text{if } rand(j) \leq CR, or j = j_{rand} \\ x_i^j & \text{otherwise, } \forall j = 1, ..., n. \end{cases} \tag{3.7}$$

where $rand \in [0, 1]$, and $j_{rand} \in \{1, ..., n\}$ is a randomly chosen index, which ensures the trial individual u_i gets at least one component from the mutant individual v_i.

Finally, after obtaining the trial individual u_i, its important to decide whether or not it should become a member of the current population. To do so, the trial vector u_i^j is compared to the target vector x_i^j using the fitness function evaluation as descried in formula (3.8). The whole DE algorithm process is also showed in Alg. (3.14)

$$x_i = \begin{cases} u_i & \text{if } f(u_i) < f(x_i) \\ x_i & \text{otherwise} \end{cases} \tag{3.8}$$

Algorithm 3.14 Differential Evolution (DE)

Inputs: N: *Population size*, F: *Scaling factor*, CR: *Crossover rate*

1: **Begin:**
2: $P = \{x_1, ..., x_N\} \leftarrow$InitializePopulation( );
3: **while** termination conditions not met **do**:
4: **for** each $i \in \{1, ..., N\}$ **do**: ▷ *for each target individual* x_i
5: $x_{r1}, x_{r2}, x_{r3} \leftarrow$SelectAtRandom($P$); ▷ *Where:* $i \neq r1 \neq r2 \neq r3$
6: $v_i \leftarrow$ApplyDiffrentialMutation($x_{r1}, x_{r2}, x_{r3}, F$); ▷ *mutant individual* v_i
7: $u_i \leftarrow$ApplyCrossover(x_i, v_i, CR); ▷ *trial individual* u_i
8: **if** $f(u_i) < f(x_i))$ **then** ▷ *if* u_i *is better than* x_i
9: $x_i \leftarrow u_i$; ▷ *update* P *by replacing* x_i *with* u_i
10: **end if**
11: **end for**
12: **end while**
13: **End**

According to the strategy implemented in differential mutation and also the type of crossover used, different variants of DE have been suggested [Price *et al.* 2005]. As mentioned in differential mutation, the General notation used to express a DE variant is "DE/x/y/z", where x specifies the individual to be mutated, which currently can be *rand* (a randomly chosen individual) or "best" (the best individual in the current population), y is the number of difference vectors used, and z denotes the crossover scheme. The simplest and the most commonly used DE variant is "DE/rand/1/bin".

As discussed above, DE has only three input control parameters, the population size N, the scaling factor F, and the crossover control parameter CR. These parameters my be kept fixed during the optimization process. Therefore, various strategies is proposed [Liu & Lampinen 2005, Teng *et al.* 2009] to make the setting of the parameters self-adaptive according to the learning experiences. Due to the success of DE in tackling single-objective optimization problems in continuous search spaces, It is extended to handle other problems as multi-

objective optimization [Mezura-Montes *et al.* 2008] and combinatorial problem [Zhang *et al.* 2009, Kafafy *et al.* 2012a]. Several modified versions of DE are available in literature for improving the performance of basic DE including hybridized versions, where DE is combined with some other algorithm to produce a new algorithm. we refer to [Chakraborty 2008, Neri & Tirronen 2010] for a more details of many of the existing variants and applications of DE.

3.6.2.2 Scatter Search

Scatter search (SS) is a population based metaheuristic in which the search process depends on maintaining a set of high-quality and diversified solutions called "*reference set*". The new solutions are generated through combining their parents solutions which are selected from the reference set [Laguna *et al.* 2003]. In SS algorithm, an initial population satisfying the conditions of diversity and quality is generated. The reference set of small size is then constructed by selecting good representative solutions from the population. The reference set is used to construct subsets of selected solutions for combination. The selected solutions are combined to provide starting solutions to an improvement procedure by means of local search before using them to update the reference set. According to the result of the improvement procedure, the reference set and even the population of solutions are updated to incorporate both high-quality and diversified solutions. The process is iterated until a stopping criterion is satisfied. SS provides diversification and intensification to the search process through using the diversified reference set and applying local search to improve the solutions' quality.

Algorithm 3.15 SCATTER SEARCH (SS)

```
 1: Begin:
 2: P ←CreateDiversifiedPopulation( );          ▷ By using greedy heuristics
 3: P ←Improvement(P);                           ▷ By using local search means
 4: RefSet ←UpdateRefrenceSet(P);                ▷ keep high quality and diversified sols.
 5: while termination conditions not met do:
 6:     SubSets ←SubsetsGeneration(P);
 7:     repeat:
 8:         Sols ←SolutionCombinations(SubSets);
 9:         P ←Improvement(Sols);                ▷ By using local search means & update P
10:     until termination condition 1 is met
11:     RefSet ←UpdateRefrenceSet(P);
12: end while
13: End
```

The main steps of the Scatter Search algorithm are presented in Alg. 3.15. They involve different procedures which are explained in the following:

- CreateDiversifiedPopulation(): This procedure generates a set of

diverse initial solutions using for example greedy procedures to diversify the search.

- IMPROVEMENT(): In this procedure, local search strategies are applied on trial solutions for obtaining local optima.
- UPDATEREFRENCESET(): This procedure collects the best solutions in terms of both quality and diversity to update the reference set $RefSet$.
- SUBSETSGENERATION(): This procedure operates on the reference set $RefSet$ to produce subsets of solutions for creating new trial solutions.
- SOLUTIONCOMBINATION(): In this procedure, a given subset of solutions are combined using for example linear combinations or generalized rounding to discrete variables, for producing new trial solution. The linear combinations are chosen to produce points both inside and outside the convex regions spanned by the reference solutions in Euclidean space. The path relinking strategies can also be attended in this task. That is, the path between two solutions will generally yield solutions that share common attributes with the input solutions in the neighborhood space.

SS has been successfully applied to a wide range of applications, a detailed overview and references on these methods are provided in [Glover *et al.* 2003]. The basic principles of SS and its recent works can be found in [Laguna *et al.* 2003] and [Resende *et al.* 2010b] respectively.

3.6.2.3 Path Relinking

Path relinking (PR) metaheuristic was proposed in the context of scatter search and tabu search [Glover 1996, Glover *et al.* 2000]. The main idea of path relinking is to investigate the trajectory in the search space connecting a starting solution s and a target solution t. This can be accomplished by selecting moves that introduce attributes contained in the target solutions, and incorporating them in an intermediate solution initially originated in the starting solution. Thus, the path between two solutions in the search space (neighborhood space) will generally provide solutions that share common attributes with the input solutions. However, PR attempts to find the best solution found in the sequence of neighboring solutions in the decision space which are generated from the starting solution to the target solution. The general template of the PR procedure is presented in Alg. 3.16. Initially, an intermediate solution x is initialized by the starting solution s. At each iteration, the best move m in terms of the objective function and decreasing the distance between the two solutions x and t is selected and applied to x. This is repeated until the distance is equal to 0. finally, the best solution found in the explored trajectory is returned. Fig. 3.9 depicts this process.

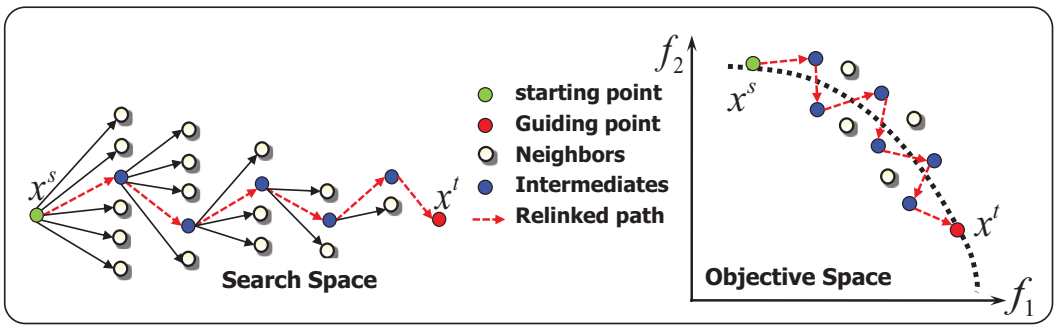


Figure 3.9: Path Relinking strategies

Algorithm 3.16 PATH RELINKING (PR)

Inputs: s: *starting solution*, t: *target solution*

1: **Begin:**
2: $x \leftarrow s$; ▷ *initialize intermediate solution* x
3: **while** $dist(x,t) \neq 0$ **do**:
4: $m \leftarrow$FINDTHEBESTMOVE(x,t); ▷ *the best move that decreases* $dist(x,t)$
5: $x \leftarrow$APPLYMOVE(x,m); ▷ *Apply the move* m *to* x
6: **end while**
7: **End**

In PR metaheuristic, for each pair of solutions s and t, several strategies to select the starting and the target solution are found. Some of these ways are described as follows:

- *Forward*: In this strategy, the relinking process starts from the worst solution among s and t.
- *Backward*: The relinking process in this strategy starts from the better solution among s and t. As starting from the best gives the algorithm a better chance to investigate in more details the neighborhood of the most promising solutions, the backward strategy is in general better than the forward one [Ribeiro *et al.* 2002].
- *Backward and forward relinking*: In this strategy, two paths are constructed in parallel, using alternatively s as the starting and the target solutions. This means, an additional computation time overhead which is not surely balanced by the quality of the obtained solutions.
- *Mixed relinking*: As in the backward and forward relinking strategy, two paths are constructed in parallel from s and t but the guiding solution is an intermediate.

Moreover, Path-relinking extensions to multiobjective are found in [Zeng *et al.* 2013a, Zeng *et al.* 2013b].

3.6.2.4 Memetic Algorithms

The term memetic algorithms (MA) has been used to identify a broad class of hybrid metaheuristics: evolutionary algorithms that incorporate local search heuristics, specialised recombination/mutation operators and/or other "helpers" specifically designed to exploit the knowledge of the problem domain [Moscato 1989, Moscato 1999]. While genetic algorithms are inspired by the metaphor of genes, memetic algorithms are inspired by the metaphor of memes. A gene is the unit of genetic information that is propagated biologically between generations during the evolution process. A meme is the unit of conceptual information (knowledge, ideas, behaviour, customs, etc.) that is transmitted by imitation from one generation to the next one. Then by incorporating the available knowledge about the problem into an evolutionary algorithm, the working metaphor is that of evolving a population both biologically and culturally. Since the term memetic was introduced some time after researchers have started to study this kind of hybrids, it is common that names such as genetic local search, hybrid genetic algorithms and others are used when referring to memetic algorithms [Reeves 1996, Falkenauer 1996, Burke & Smith 2000, Jaszkiewicz 2002a].

3.6.3 Swarm Intelligence

Swarm Intelligence (SI) algorithms[Pinto *et al.* 2005, Runkler 2008] is an innovative distributed intelligent paradigm for solving optimization problems that takes inspiration from the collective behavior of a group of social insect colonies and of other animal societies. SI Algorithms consist of a population of simple agents that cooperate by an indirect communication medium, and do movements in the decision space. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global and self-organized behavior. Among the most successful swarm intelligence inspired optimization algorithms are ant colony and particle swarm optimization.

3.6.3.1 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population-based metaheuristic inspired from swarm intelligence [Eberhart *et al.* 2001]. It was initially introduced in [Kennedy & Eberhart 1995] as a global optimization technique and soon became a very popular global optimizer, mainly in problems in which the decision variables are real numbers [Eberhart *et al.* 2001, Engelbrecht 2007]. PSO simulates the social behavior of natural organisms such as bird flocking and fish schooling to solve optimization problems. It has been successfully designed for continuous optimization problems. PSO consists of a population of particles, which contrary

to EA, survive up to the last generation. Each particle is a candidate solution to the problem, and is represented by a velocity, a position in the search space and has a memory which helps it in remembering its previous best position. The particles search the decision space by flying with a special speed (velocity) towards their "guide" by using their experience from the past generations. Moreover, every particle swarm has some sort of topology describing the interconnections among the particles. The set of particles to which a particle i is topologically connected is called $i's$ neighborhood. Thus, the particle in the swarm can interchange the information with every particle belongs to its neighborhood. The neighborhood may be the entire population or some subset of it. The two most commonly used neighborhood topologies are known as *gbest* (for "global best") and *lbest* (for "local best"). In the global one, particles are moving to the best particle that has been found so far. This leads the algorithm to have a quick convergence pattern. However, it may thus become trapped in local optima. On the other hand, in the local PSO, each particle's velocity is adjusted according to both its personal best and the best particle within its neighborhood. In this variant, the algorithm can escape from getting stuck in local optima, but it may suffer from a slower convergence pattern [Liang & Suganthan 2006]. The effects of various population topologies on the particle swarm algorithm were investigated in [Kennedy & Mendes 2002].

A PSO method can be formulated as follows. A set of N particles are considered as a population P_t in the generation t. Each particle i in n-dimensional decision space $\mathcal{S}$ has a position defined by $x_i = (x_{i1}, ..., x_{in})$ and a velocity defined by $v_i = (v_{i1}, ..., v_{in})$. At each generation t, Each particle i successively adjusts its position x_i toward the global optimum according to the following two factors: the best position visited by itself (personal best) denoted as $x_i^{pb} = (x_{i1}^{pb}, ..., x_{in}^{pb})$ and the best position visited by the whole swarm (*gbest*) (or *lbest*, the best position visited by its neighborhood denoted as $x_i^{lb} = (x_{i1}^{lb}, ..., x_{in}^{lb})$) denoted as $x_i^{gb} = (x_{i1}^{gb}, ..., x_{in}^{gb})$. Thus, the velocity v_i and the position x_i at generation $t+1$ can be calculated as follows:

$$v_i(t+1) = w \times v_i(t) + c_1 r_1 (x_i^{pbest} - x_i(t)) + c_2 r_2 (x_i^{gbest} - x_i(t)) \tag{3.9}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.10}$$

where, $v_i(t)$ is the velocity of particle i at generation t, and $v_i \in [-v_{max}, v_{max}]$ keep the particles from flying out of the problem space, w is the inertia weight factor, c_1, c_2 are the acceleration coefficients, r_1, r_2 are uniform random numbers within $[0, 1]$. The *lbest* PSO can be obtained by replacing x_i^{gbest} in formula (3.9) with x_i^{lbest} for each particle i.

However, there are three fundamental elements for the calculation of the next

position of a particle: according to its own velocity, towards it best performance and the best performance of its best informant. The way in which these three vectors are combined linearly via confidence coefficients is the bases of all the versions of the classic PSO. The three fat arrows represent such combination, which will give the next position of the particle. Fig.3.10 shows this process.

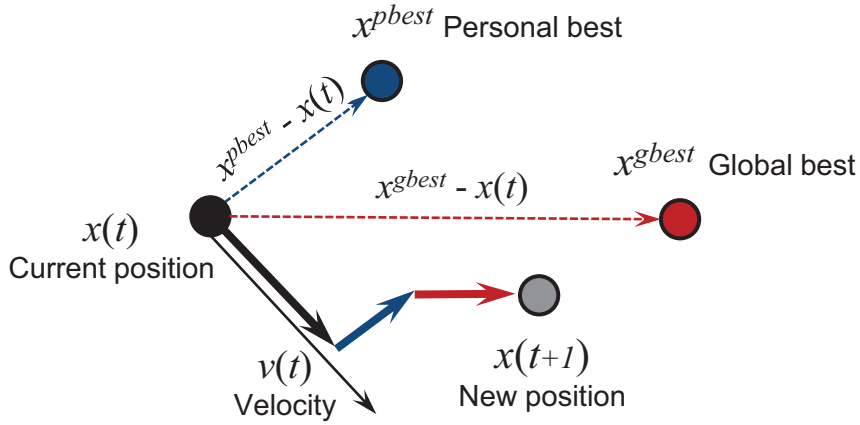


Figure 3.10: Changing particles' positions in PSO

In PSO, the performance of each particle is measured according to a predefined fitness function, which is related to the problem to be solved. Alg.3.17 summarizes the structure of the general PSO algorithm.

Algorithm 3.17 PARTICLE SWARM OPTIMIZATION(PSO)

Begin: $t \leftarrow 0$;
for each Particle $i \in \{1, ..., N\}$ **do**: ▷ *for each particle* x_i
  $x_i(t) \leftarrow$ INITIALIZEPARTICLE( ); ▷ *initialize particles positions*
  $v_i(t) \leftarrow$ INITIALIZEVELOCITY( ); ▷ *initialize velocity*
  $f(x_i(t)) \leftarrow$ EVALUATION$(x_i(t))$; ▷ *Evaluate particle* x_i
end for
while $t < t_{max}$ **do**:
  for each Particle $i \in \{1, ..., N\}$ **do**: ▷ *for each particle* x_i
    $v_i(t+1) \leftarrow$ ADAPTVELOCITY$(v_i(t), x_i(t), x_i^{pbest}, x_i^{gbest})$; ▷ *use formula 3.9*
    $x_i(t+1) \leftarrow$ UPDATEPOSITIONS$(v_i(t+1), x_i(t))$; ▷ *use formula 3.10*
    $f(x_i(t+1)) \leftarrow$ EVALUATION$(x_i(t+1))$; ▷ *Evaluate particle* x_i
    if $f(x_i(t+1)) < f(x_i^{pbest})$ **then**: $x_i^{pbest} \leftarrow x_i(t+1)$; ▷ *update personal best*
    if $f(x_i(t+1)) < f(x_i^{gbest})$ **then**: $x_i^{gbest} \leftarrow x_i(t+1)$; ▷ *update global best*
  end for
  $t \leftarrow t+1$; ▷ *update iteration counter*
end while
End

The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thus to influence the trade-off between global

and local exploration abilities of the particles [Shi & Eberhart 1998]. A larger inertia weight w facilitates global exploration (i.e., diversifies the search in the whole search space) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area (i.e., intensifies the search). Suitable selection of the inertia weight w can provide a balance between global and local exploration abilities and thus requires less iterations on average to find the optimum. Rather than applying inertia to the velocity memory, an alternative version of PSO proposed in [Clerc & Kennedy 2002], in which a constriction factor τ is applied for the velocity adjustment as given as follows:

$$v_i(t+1) = \tau \times [v_i(t) + c_1 r_1 (x_i^{pbest} - x_i(t)) + c_2 r_2 (x_i^{gbest} - x_i(t))] \tag{3.11}$$

where $\tau = \frac{2}{|\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}|}$ and $\varphi = \varphi_1 + \varphi_2$, $\varphi > 4$. This version of PSO eliminates the parameter v_{max}, where $\varphi = 0.729$, while $c_1 = c_2 = 1.49445$.

The effectiveness of the particle swarm algorithm comes from interactions of particles with their neighbors. As one particle discovers an optimum, it becomes the best in the neighborhood and attracts (guides) the other particles to itself. Indeed, there is no selection process in a PSO method, unlike EAs. One of the drawbacks of the standard PSO is premature convergence and trapping in local optima. More research has been developed to provide PSO convergence results through understanding theoretically how PSO algorithm works and why under certain conditions it might fail to find a good solution [Clerc & Kennedy 2002, van den Bergh & Engelbrecht 2006]. Moreover, considerable research has been also conducted into further refinement of the original formulation of PSO in both continuous and discrete problem spaces[Kennedy & Eberhart 1997], and areas such as parallel implementation [Banks *et al.* 2007] and MultiObjective Optimization [Reyes-Sierra & Coello 2006]. Modified versions of PSO based on diversity, mutation, crossover and efficient initialization using different distributions and low-discrepancy sequences are discussed in [Pant *et al.* 2009]. Finally, a large number of hybrid variants have been proposed, such as [Valdez *et al.* 2011, Idoumghar *et al.* 2011, Dor *et al.* 2012].

3.6.3.2 Ant Colony Optimization (ACO)

The ant colony optimization (ACO) [Dorigo *et al.* 1996] is a metaheuristic inspired by the collective behavior of ants when finding the shortest path between a food source and their nest. When searching their food, these ants initially explore the area surrounding their nest by performing a randomized walk. while exploring paths, ants deposit a volatile substance called pheromone for marking some favorable paths that should guide other ants to the food source. They use the amount of pheromone deposited to decide which path to follow. After some

time, and due to the pheromone evaporation as time passes, the shortest path between food source and the nest will have high level of pheromone concentrations, and therefore it becomes more attractive to more ants. An ant colony optimization algorithm exploits this idea to tackle hard combinatorial optimization problems. The algorithm emulate the idea through developing the so-called a parametrized probabilistic model (*pheromone mode*) that consists of a set of parameters (pheromone values) act as the memory that keeps track of the search process. ACO consists of a set of artificial ants that incrementally construct solutions by adding components to their solutions. This can be done through an iterative process composed of two steps. In the first step, solutions are constructed using a pheromone model, that is, a parametrized probability distribution over the solution space. The constructed solutions and possibly solutions that were constructed in earlier iterations are used to modify the pheromone values in a way that is deemed to bias future sampling toward high quality solutions.

The general framework of a basic ACO metaheuristic is described in Alg. 3.18. After initializing pheromone values, ACO iterates over three steps until stopping criterion. The first step is the module AntBasedSolutionConstruction() in which a set of solutions to the tackled problem is probabilistically constructed from a set of solution components using artificial ants. Then, the pheromone values are updated by the module PheromoneUpdate() in the second step. A standard pheromone update consists of two mechanisms: *pheromone evaporation* and *pheromone deposit*. First, a pheromone evaporation, which uniformly decreases all the pheromone values, is performed. Practically, pheromone evaporation is needed to avoid a premature convergence of the algorithm to suboptimal solutions and then favoring the exploration of unvisited areas of the search space. The pheromone deposit is applied after all ants have finished constructing a solution. The pheromone values are increased on solution components that are associated with a chosen set of high quality solutions. The goal is to make these solution components more attractive for ants in the following iterations. Many different schemes for pheromone update have been proposed within the ACO framework. Finally, Daemon actions are optional steps that refer to any centralized operation which cannot be performed by a single ant. The most used daemon action consists in the application of local search to the constructed solutions.

A recent overview of ACO can be found in [Dorigo & Stutzle 2010]. It reveals that the majority of the currently published articles on ACO are clearly on its application to computationally challenging problems. For more details on ACO algorithm and its applications, we refer to [Dorigo *et al.* 1996, Blum & Roli 2001] and [Gambardella & Dorigo 2000, Merkle *et al.* 2002, Shmygelska & Hoos 2005]

Algorithm 3.18 ANT COLONY OPTIMIZATION(*sol*)

```
1: Begin:
2: InitializePheromoneValues( );
3: repeat:
4:     AntBasedSolutionConstruction( );
5:     PheromoneUpdate( );
6:     DaemonActions( );
7: until termination condition is met
8: End
```

3.7 Multiobjective Metaheuristics

Solving multiobjective optimization problems using metaheuristics becomes an active research filed, especially population-based metaheuristics as they can simultaneously explore the search space with achieving convergence to the true Pareto front and uniform diversity. The multiobjective metaheuristic algorithm involves three basic search components. These components are the fitness assignment strategy, diversity preservation and elitism. Fitness assignment assigns a scalar-valued fitness to a vector objective function in order to guide the search algorithm toward Pareto optimal solutions. Diversity preservation strategy helps to generate a diverse set of Pareto solutions in the objective and/or the decision space. The elitism strategies tries to maintain and use the elite solutions for improving the performance of a metaheuristic algorithm. These components are briefly discussed in the following.

3.7.1 Fitness assignment strategies

Fitness assignment is used to map the vector fitness to a scalar value which is used to measure the solution quality. The fitness assignment strategies used in multiobjective metaheuristics contain four basic categories as explained in the following.

- *Scalar approaches*: These strategies are essentially based on converting the MOPs into single objective problems and then the metaheurstic algorithms are applied to the single objective problem. These strategies include the scalarization techniques explained in section 2.7.1. As mentioned, these methods need the decision maker to have good information about the problem. The most common metaheuristics based on these strategies are MOSA (multiobjective simulated annealing) algorithm [Ulungu *et al.* 1999] and MOTS (Multiobjective Tabu Search) algorithm which is a one of the first scalar approaches adapted to tabu search [Hansen 1997].

- *Criterion-based approaches*: In these strategies, the search is performed

by treating the various objective functions separately such as the parallel selection in evolutionary algorithms, parallel pheromone update in ant colony optimization, and lexicographic ordering. The most common parallel selection in evolutionary algorithms designed in the vector evaluated GA [Schaffer 1985], in which the individuals are selected from the current population according to each objective, independently from the other ones. In sequential or Lexicographic Approaches, the search process is performed according to a given preference order [Fishburn 1974] of the objective defined by the decision maker.

- *Pareto-based approaches*: Unlike the other approaches that use scalarization or treat the objectives separately, these strategies use the concept of dominance and Pareto optimality to guide the search process as introduced in [Goldberg 1989]. Thus, there is no need to transform the MOP into a single objective one. These approaches deal simultaneously with a set of solutions that allow to find several members of the Pareto optimal set in a single run of the algorithm. They also have the ability to reach Pareto solutions in the non-convex portions of the Pareto front. The commonly used approaches use these fitness assignment are NSGA-II and SPEA2. To guide the search toward the true Pareto front, several Pareto ranking procedures are applied. The most popular dominance-based ranking procedures are dominance rank, dominance depth and dominance count [Zitzler *et al.* 2004]. In dominance rank, each solution is ranked based on number of solutions in the population that dominate it. Whereas in dominance depth, the population is decomposed into several fronts. The first front which has rank 1 is constructed from the nondominated solutions of the population. This front is eliminated form the population. The second front which has rank 2 is calculated by the same manner and the process is continued until obtaining empty population. In the dominance count, each solution takes a rank based on the number of solutions dominated by it (see Fig.3.11).

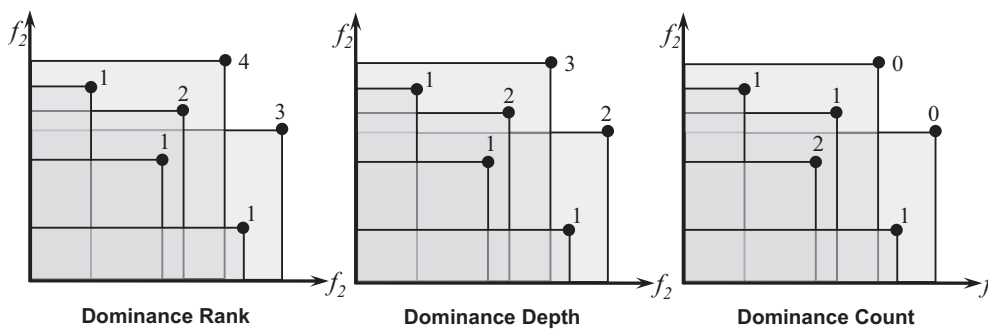


Figure 3.11: Some dominance-based ranking methods

- Indicator-based approaches: In these strategies, a performance quality indicator [Zitzler *et al.* 2004] is adopted to guide the search process toward the Pareto front. Several advantages gained by these approaches as incorporating the decision maker preferences into the optimization algorithm and there is no need to use diversity maintenance.

3.7.2 Diversity preservation

In order to obtain a good approximation to Pareto front, it is important in population based metaheuristic to preserve the diversity of the population during the search process. The fitness assignment methods tend to favor the convergence toward the Pareto optimal front. However, these methods are not able to guarantee that the approximation obtained will be of good quality in terms of diversity, either in the decision or objective space. There are several methods used to preserve the diversity in population based metaheuristics. These methods can be classified into three categories, kernel methods, Nearest-Neighbor Methods and Histogram methods. In kernel methods, a kernel function which takes the distance between solutions as argument, is used to estimate the density of the solution. Whereas in Nearest-Neighbor methods, the distance between a given solution i and its k_{th} nearest neighbors are taken into account to estimate the density of the solution. In histograms methods, the search space is divided into several hypergrids defining the neighborhoods. The density around a solution is estimated by the number of solutions in the same box of the grid, see Fig.3.12.

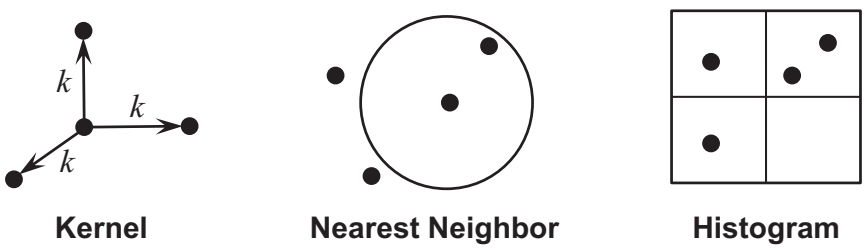


Figure 3.12: Diversity maintaining strategies

3.7.3 Elitism

Elitism or archiving strategy is a mechanism used to maintain the high quality solutions encountered during the search process. By using elitism, it can be guaranteed that the maximum fitness of the population never decreases from one generation to the next, and hence, a faster convergence of the population can be achieved. Elitism strategy can be designed as either *passive* or *active*. In passive elitism, a secondary population, called *archive*, is used only to store these

high-quality solutions without any participation on the search process. Whereas in active elitism, the archived solutions are used to generate new solutions. Active elitism leads to achieve faster convergence toward the Pareto front for a better approximation of the Pareto front [Zitzler & Thiele 1999]. The archive updating strategy may depend on size, convergence and diversity criteria. In size criteria, a specified number of nondominated solutions are stored in the archive. If the archive reaches its maximum size, the convergence and diversity are used to update it. In convergence criteria, fitness assignment strategies such as scalar fitness, dominance based or indicator based strategies are used to update the archive. Dominance based strategies are the most common used methods in which nondominated solutions are stored in the archive to reach its maximum size and then diversity criteria should be applied. The ε-dominance concept in which the objective space is divided into hyper boxes with a certain size ε can be applied to update the archive [Laumanns *et al.* 2002]. Each hyper box maintains only one solution. The Pareto adaptive $pa\varepsilon$-dominance which allows adaptive size to each hyper box can also used [Hernández-Díaz *et al.* 2007]. Fig.3.13 depicts updating the archive using both ε-dominance and $pa\varepsilon$-dominance. Generally, the archive updating process is designed based on both convergence and diversity criteria.

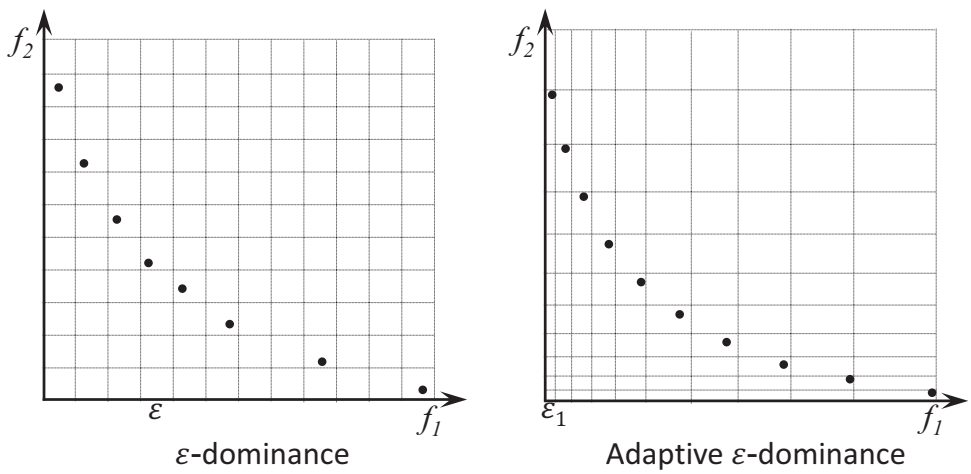


Figure 3.13: Updating archive by ε-dominance vs. $pa\varepsilon$-dominance

3.7.4 Multiobjective Evolutionary Algorithms

One of the goals of multiobjective optimization is to approximate the set of Pareto optimal solutions. Sets of solutions can be easily found in one simulation run by EAs. Indeed, EAs explore the search space by several individuals in the population. Therefore, they can simultaneously explore the search space and under certain additional assumptions can converge to a set of optimal solutions. In other

words, in single-objective optimization the whole population converges to a single minimum, whereas in multiobjective optimization they can theoretically converge to the set of Pareto-optimal solutions [Rudolph 1998, Rudolph & Agapie 2000]. This advantage has attracted many researchers to solve MOPs using Multiobjective Evolutionary Algorithms (MOEAs). In the following sections, an introduction to MOEA history and a brief summary for some of these methods are introduced.

3.7.4.1 MOEA history

The MOEAs' history goes back to the first MOEA, called Vector Evaluated Genetic Algorithm (VEGA) that introduced in [Schaffer 1985]. This method adopts the criterion-based selection. VEGA is the simplest possible MOEA and is a straightforward extension of a single-objective EA. In [Kursawe 1991], Kursawe introduced his method which adopts the same idea as VEGA. Later, Fonseca and Fleming (MOGA) [Fonseca *et al.* 1993], Horn and Nafpliotis (NPGA) [Horn *et al.* 1994], and Srinivas and Deb (NSGA) [Srinivas & Deb 1994] introduced other kinds of MOEAs which use Pareto-based selection, niching and visual comparisons. These methods eliminate the limitations of VEGA in achieving a good diversity of solutions by introducing the non-domination concept and an explicit diversity-preservation operator. After, Zitzler and Thiele added elitism as an important part in MOEA [Zitzler & Thiele 1999], which can guarantee convergence. Their method, Strength Pareto Evolutionary Algorithm (SPEA) and also later methods from Knowles and Corne (PAES,PESA) [Knowles & Corne 1999, Corne *et al.* 2000], Deb et al. (NSGAII) [Deb *et al.* 2000] and Zitzler et al. (SPEA2) [Zitzler *et al.* 2001] made another category of MOEA based on archiving, elitism, and quantitative performance metrics. These methods are supposedly faster and better than the previous methods. This is due to the elite-preserving operator. An elite-preserving operator gives the elite solutions to be directly carried over to the next generations. Therefore, it ensures that the fitness of the best solution does not deteriorate. Recently, instead of using Pareto concepts, some researchers introduce MOEAs which adopts the decomposition of the original MOP into a set of single-objective subproblems based on the conventional mathematical programming in approximating the PF [Miettinen 1999]. Some of MOEAs adopt this idea such as MOGLS [Jaszkiewicz 2002b, Ishibuchi & Murata 1998] and MOEA/D [Zhang & Li 2007].

3.7.4.2 MOEA methods

The MOEA methods are arranged into Pioneers (VEGA) , Classic (First generations), and Elitist (second generation) categories. These methods constitute the

basic part of the MOEA, however there are many other recorded MOEA methods for particular problems [Deb 2001, Coello *et al.* 2007]. The first and the second generations are based on the Pareto dominance concept. Here, some of the most popular of these methods are briefly discussed.

Vector Evaluated Genetic Algorithm (VEGA): In VEGA [Schaffer 1985], the objective-wise selection is adopted. It works efficiently for some generations, but in some cases suffers from the principal of not using the domination criterion in fitness evaluation. Therefore, it is not able to deliver a good spread of solutions in some cases. In VEGA only the selection operator was modified so that at each generation, a number of sub-populations were generated by performing proportional selection according to each objective function in turn. Thus, for a problem with k objectives, k sub-populations of size N/k each would be generated, assuming a total population size of N. These sub-populations would be shuffled together to obtain a new population of size N, on which the GA would apply the crossover and mutation operators in the usual way.

MOEAs First Generation: The major step towards the first generation of MOEAs is given by David Goldberg [Goldberg 1989]. Goldberg analyzes VEGA and proposes a selection scheme based on the concept of Pareto optimality. Goldberg not only suggested what would become the standard first generation MOEA, but also indicated that stochastic noise would make such algorithm useless unless some special mechanism was adopted to block convergence. First generation MOEAs typically adopt niching or fitness sharing for that sake. Also, first generation MOEA is characterized by the use of selection mechanisms based on Pareto ranking. The most representative algorithms from the first generation include the Nondominated Sorting Genetic Algorithm (NSGA) [Srinivas & Deb 1994], the Niched Pareto Genetic Algorithm (NPGA) [Horn *et al.* 1994] and the Multiobjective Genetic Algorithm (MOGA) [Fonseca *et al.* 1993]. Here, NSGA is introduced as follows:

Nondominated Sorting Genetic Algorithm (NSGA): It was proposed in [Srinivas & Deb 1994]. This approach is based on several layers of classifications of the individuals as suggested by Goldberg [Goldberg 1989]. Before selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diversity of the population, these classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of nondominated individuals is considered. The process continues until all individuals in the population are classified. Stochastic proportionate selection is adopted for this technique. Since

individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. This allows searching for nondominated regions, and results in convergence of the population toward such regions. Sharing by its part helps to distribute the population over the Pareto frontier of the problem.

MOEAs Second Generation: The second generation of MOEAs was born with the introduction of the notion of elitism. In the context of multiobjective optimization, elitism usually (although not necessarily) refers to the use of an external population to retain the nondominated individuals. The use of this external file raises several questions:

- How does the external file interact with the main population?
- What do we do when the external file is full?
- Do we impose additional criteria to enter the file instead of just using Pareto dominance?

Elitism can also be introduced through the use of a selection in which parents compete with their children and those which are nondominated (and possibly some additional criterion such as providing a better distribution of solutions) are selected for the following generation. In the following, some of the second generation MOEAs implement elitism will be discussed.

Strength Pareto Evolutionary Algorithm (SPEA): SPEA [Zitzler & Thiele 1999] is conceived as a way of integrating different MOEAs. It uses an archive containing nondominated solutions previously found (called external nondominated set). At each generation, nondominated individuals are copied to the external nondominated set. For each individual in this external set, a strength value is computed. This strength is similar to the ranking value of MOGA [Fonseca *et al.* 1993], since it is proportional to the number of solutions to which a certain individual dominates. It should be obvious that the external nondominated set is in this case the elitist mechanism adopted.

In SPEA, the fitness of each member of the current population is computed according to the strengths of all external nondominated solutions that dominate it. In Addition, a clustering technique is used to keep diversity.

Strength Pareto Evolutionary Algorithm 2 (SPEA2): SPEA2 [Zitzler *et al.* 2001] is an improved version of SPEA. It has three main differences with respect to its predecessor:

1. SPEA2 incorporates a fine-grained fitness assignment strategy which takes into account for each individual the number of individuals that dominate

it and the number of individuals by which it is dominated.

2. It uses a nearest neighbor density estimation technique which guides the search more efficiently.
3. It has an enhanced archive truncation method that guarantees the preservation of boundary solutions.

Therefore, in this case the elitist mechanism is just an improved version of the previous. The procedure which describes SPEA2 steps is presented in Alg. 3.19 as follows:

Algorithm 3.19 SPEA2($N, ArchSize$)

Inputs:
N: *Population size*
$ArchSize$: *Archive size*

1: **Begin:**
2: $P_0 \leftarrow$ InitializeRandomPopulation( ); ▷ *Initial Population of size N*
3: $Arch_0 \leftarrow \phi$; ▷ *Begin with an empty archive*
4: $t \leftarrow 0$
5: **while** $t \leq MaxGen$ **do**:
6: FitnessAssignment($P_t, Arch_t$); ▷ *Calculate fitness values of individuals in $P_t, Arch_t$*
7: $Arch_{t+1} \leftarrow$ CollectNondominatedIndividuals(P_t, $Arch_t$);
8: $Arch_{t+1} \leftarrow$ ArchiveTruncation($Arch_{t+1}, ArchSize$);
9: $MatingPool \leftarrow$ BinaryTournamentSelection($Arch_{t+1}$);
10: $P_{t+1} \leftarrow$ ApplyCrossoverMutation($MatingPool$);
11: $t \leftarrow t + 1$;
12: **end while**
13: **return** $Arch_t$; ▷ *Return the best Pareto rank*
14: **End**

Elitist Nondominated Sorting GA (NSGA-II): NSGA-II is proposed in [Deb *et al.* 2000] as a revised version of the NSGA [Srinivas & Deb 1994], which uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters. Alg. 3.20 provides the NSGA-II procedure. In this algorithm, the offspring population Q_t of size N is created from the parent population P_t of size N. First, these two populations are combined to form a population R_t of size $2N$. Then, a nondominated sorting procedure is used to classify the entire population R_t as follows:

1. Nondominated individuals are calculated from R_t and are called nondominated solutions of the first frontier. Then, these are temporarily disregarded from R_t and the nondominated solutions of the remaining elements of R_t are then determined and called non dominated solutions of 2nd frontier. This procedure is continued until all the members of R_t are classified into a non-dominated frontiers. This is called sorting process.

2. After the sorting process, the new population is filled with solutions of different nondominated fronts, one at a time. The filling starts with the best nondominated front. Since R_t has $2N$ solutions and the new population must have N solutions, just N best solutions appear in the new population (N solutions from the best frontiers). In the case of inadequate available space in the new population to accommodate all solutions of a nondominated set, a crowding strategy is used to identify solutions which reside in less crowded areas (in the objective space).

The NSGA-II does not use an external memory as the previous algorithms. But, the elitist mechanism consists of combining the best parents with the best offspring obtained (selection). Fig. 3.14 depicts the procedure of NSGAII. A new variant of NSGA2 is called CNSGA2 [Deb *et al.* 2003], which uses a clustering technique instead of a crowding strategy in the second step.

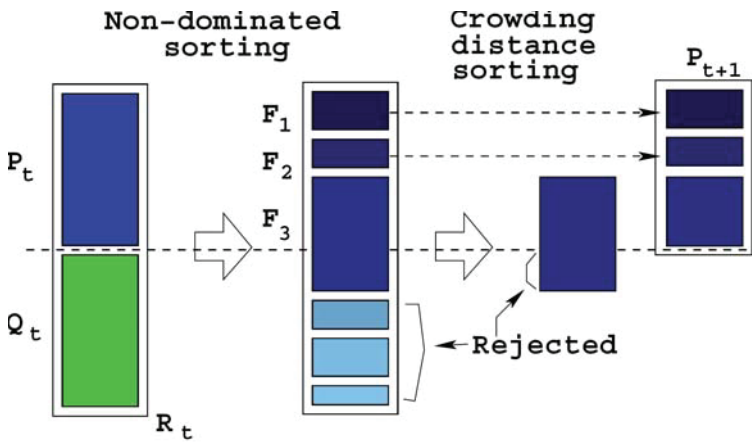


Figure 3.14: NSGAII procedure [Deb *et al.* 2003]

3.8 Hybrid Metaheuristics

Over the last few years, hybrid metaheuristics based algorithms become an important research issue in the filed of optimization. Due to this importance, a lot of research efforts have been employed to investigate and improve the performance of these algorithms. With the exception of memetic algorithms, the metaheuristics described above can be considered pure in the sense that they are not a combination of two or more approaches. Hybrid metaheuristics have proven to be successful in many optimization problems and particularly in practical or real-world problems. The main motivation for the hybridization of different algorithmic concepts has been to obtain better performing systems that exploit and combine advantages of the individual pure strategies, that is, hybrids are believed to benefit from synergy. Choosing an adequate combination of multiple

Algorithm 3.20 NSGA-II(N)

Inputs:
N: *Population size*

Begin:
$P_0 \leftarrow$ MakeInitalRandomPopulation( ); ▷ *Initial Population of size N*
$Q_0 \leftarrow$ MakeNewPopulation(P_0); ▷ *Apply crossover & mutation to construct new population*
$R_0 \leftarrow \phi \wedge t \leftarrow 0$
while $t \leq MaxGen$ **do**:
  $R_t \leftarrow P_t \cup Q_t$ ▷ *Combine parent and offspring populations*
  $\mathcal{F} \leftarrow$ FastNonDominatedSort(R_t); ▷ *Divide the combined population into fronts*
  $P_{t+1} \leftarrow \phi \wedge i \leftarrow 1$;
  while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**: ▷ *While population size is not full*
    CrowdingDistance($\mathcal{F}_i$); ▷ *Calculate crowding measure in $\mathcal{F}_i$*
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$; ▷ *Include the ith rank into the population*
    $i \leftarrow i + 1$;
  end while
  Sort($\mathcal{F}_i$,$\prec_n$); ▷ *Sort in descending order using $\prec_n$*
  $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$; ▷ *Fill population untill size N*
  $Q_{t+1} \leftarrow$ MakeNewPopulation(P_{t+1});
  $t \leftarrow t + 1$;
end while
return $\mathcal{F}_1$; ▷ *Return the best Pareto rank*
End

algorithmic concepts is the key for achieving top performance in solving many hard optimization problems.

In Hybrid metaheuristics, two or more algorithms are combined to develop a hybrid approach better suited for the given problem [Glover & Kochenberger 2003]. The hybridization of metaheuristics has been proposed at various levels and in different ways. For example, the components of one metaheuristic can be embedded into another (using tabu lists within a genetic algorithm) or one metaheuristic can be used as a component to enhance the performance of another (simulated annealing as the local search phase in variable neighborhood search or in GRASP). The most common different types of combinations are considered as follows:

- Combining metaheuristics with (complementary) metaheuristics.
- Combining metaheuristics with exact methods from mathematical programming approaches that are mostly used in operations research.
- Combining metaheuristics with constraint programming approaches developed in the artificial intelligence community.
- Combining metaheuristics with machine learning and data mining techniques.

In the following, the classifications of hybrid metaheuristic are discussed.

3.8.1 Hybrid Metaheuristics classifications

Many classification schemes for hybrid metaheuristics have been proposed in literature [Cotta-Porras 1998, Calégari *et al.* 1999, Hertz & Kobler 2000, Talbi 2002, Puchinger & Raidl 2005, El-Abd & Kamel 2005]. For example, the framework proposed in [Hertz & Kobler 2000] which describes a wide range of evolutionary algorithms including their hybrids with local search. The authors illustrate their framework by using it to describe various evolutionary algorithms including genetic algorithms, scatter search and ant systems. A similar taxonomy called Table of Evolutionary Algorithms (TEA) was proposed in [Calégari *et al.* 1999] to compare the principles of various evolutionary algorithms also including some hybrids. Another taxonomy of hybrid metaheuristics presented in [Talbi 2002] (See Fig. 3.15). In this taxonomy, The hybridization is classified hierarchically into two levels. The first one distinguishes between low-level combination in which a given function of a metaheuristic is replaced by another metaheuristic vs. high-level combination in which the different metaheuristics are self-contained. The second level distinguishes between rely hybridization in which a set of metaheuristic is applied one after another vs. teamwork hybridization in which many parallel cooperating agents, where each agent carries out a search in a solution space. Consequently, there are four classes according to this hierarchy as follows:

- *low-level Relay hybridization*: This class of hybrids represents algorithms in which a given metaheuristic is embedded into a single-solution metaheuristic.
- *low-level teamwork hybridization*: In this class, a metaheuristic as local search is embedded into a Population based metaheuristic such as evolutionary algorithms. This class of hybrid algorithms is very popular and has been applied successfully to many optimization problems.
- *high-level Relay hybridization*: In this class, the self-contained metaheuristics are executed in a sequence. For example, the initial solution of a given metaheuristic may be generated by another optimization algorithm.
- *high-level teamwork hybridization*: This class involves several selfcontained algorithms performing a search in parallel and cooperating to find an optimum.

In [Raidl 2006], the most important aspects of some different classifications [Cotta-Porras 1998, Talbi 2002, BLUM *et al.* 2005] are combined with both the classifications with particular respect to parallel metaheuristics [El-Abd & Kamel 2005, Puchinger & Raidl 2005] and with respect to the

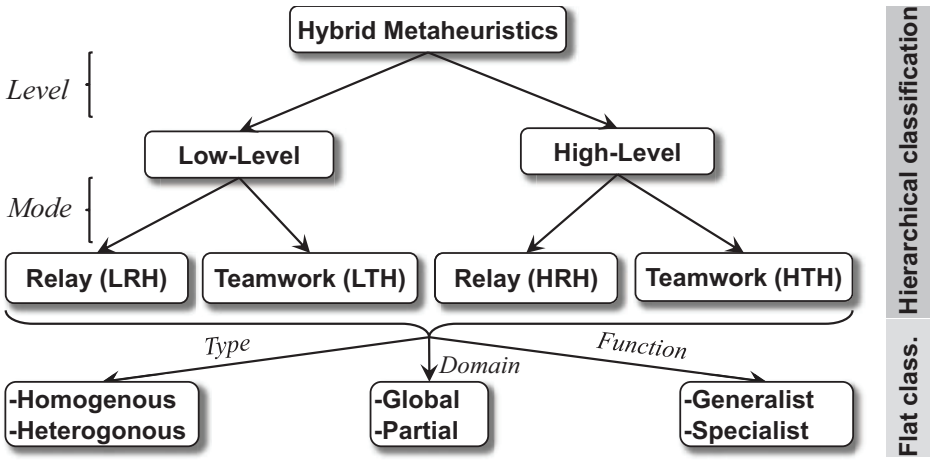


Figure 3.15: Hybrid metaheuristics classification (design issue) [Talbi 2002]

hybridization of metaheuristics with exact optimization techniques from [Puchinger & Raidl 2005]. Fig.3.16 depicts the various classes and properties that used to categorize hybrids of metaheuristics. As can be seen, the first branch describes which types of algorithms are hybridized as discussed above. The second branch distinguishes the level at which the different algorithms are combined that may be High-level combinations or low level. In high level combination, the original algorithms are just cooperate over a relatively well defined interface; there is no direct relationship of the internal workings of the algorithms. In contrast, algorithms in low-level combinations strongly depend on each other i.e. a given function of a metaheuristic can be replaced by another metaheuristic. According to the order of execution, the hybrid metaheuristic can be classified to relay, interleaved and parallel models. On relay model, one algorithm is strictly performed after the other, and information is passed only in one direction i.e. each using the output of the previous as its input. Whereas in interleaved and parallel models, there are many cooperating and parallel agents, where each agent carries out a search in a solution space. Parallel algorithms can be classified according the architecture (SIMD: single instruction, multiple data streams versus MIMD: multiple instruction, multiple data streams), the granularity of parallelism (fine- versus coarse-grained), the hardware (homogeneous versus heterogeneous), the memory strategy (shared versus distributed memory), the task and data allocation strategy (static versus dynamic), and whether the different tasks are synchronized or run in an asynchronous way. More details about hybrid parallel metaheuristic can be found in [El-Abd & Kamel 2005, Cotta *et al.* 2005]. Based on the control strategy used, hybrid metaheuristic contains integrative and collaborative (cooperative) combinations. In integrative approaches, an algorithm is considered as an embedded components in another. For example in memetic algorithms in which the local search strategy is embedded in evolu-

tionary algorithms. Also, in Very large scale neighborhood search (VLSN) approaches [Ahuja *et al.* 2002], exact techniques such as dynamic programming to efficiently find best solutions in specifically designed large neighborhoods within a local search based metaheuristic. Also embedding more powerful algorithms like path-relinking [Glover *et al.* 2000] or by exact techniques based on branch and bound on population based metaheuristic. In collaborative combinations, algorithms exchange information, but are not part of each other. For example, the popular island model [Goldberg 1989] for parallelizing evolutionary algorithms falls into this category. We can further classify the traditional island model as a homogeneous approach since several instances of the same metaheuristic are performed.

3.8.2 Multiobjective Hybrid Metaheuristics

Metaheuristic hybridization for multiobjective optimization follows the same classifications above with considering the additional search components used in multiobjective case. The hybridization of Metaheuristics for multiobjective optimization field usually follows three basic schemes. The first one is the combination among different metaheuristics which provids more efficient behavior and higher flexibility than using pure metaheuristics. The second schemes is to combine metaheuristics with Exact methods such as branch and bound techniques. The third scheme is to combine metaheurstics with data mining techniques. in the following, each scheme will be presented in some details.

3.8.2.1 Combining different metaheuristics

The strate forward method for metaheuristic hybridization is to combine different types of metaheurstics to improve the effectiveness of the pure ones. These schemes of hybridization follow the hierarchical classification provided in [Talbi 2002] that explained above. In the Low level relay class, an adaptive hybrid approach involves different single solution based metaheristics which are adaptively used according to their performance during the search process. In low level team work class, the cooperation is achieved between population based metaheuristics that have good diversification capabilities and the single solution based metaheuristics that marked by good intensification capabilities. the common example that follows this class are embedding the local search into the genetic algorithms to handle multiobjective optimization problems constructing the so-called MOGLS [Ishibuchi & Murata 1998, Jaszkiewicz 2002b]. The local search algorithm can also be adapted for multiobjective optimization by considering the Pareto dominance relation to check the neighborhood of each solution of the Pareto approximation set. The new generated solutions are used to update the Pareto approximation set with eliminating the dominated solutions. This method

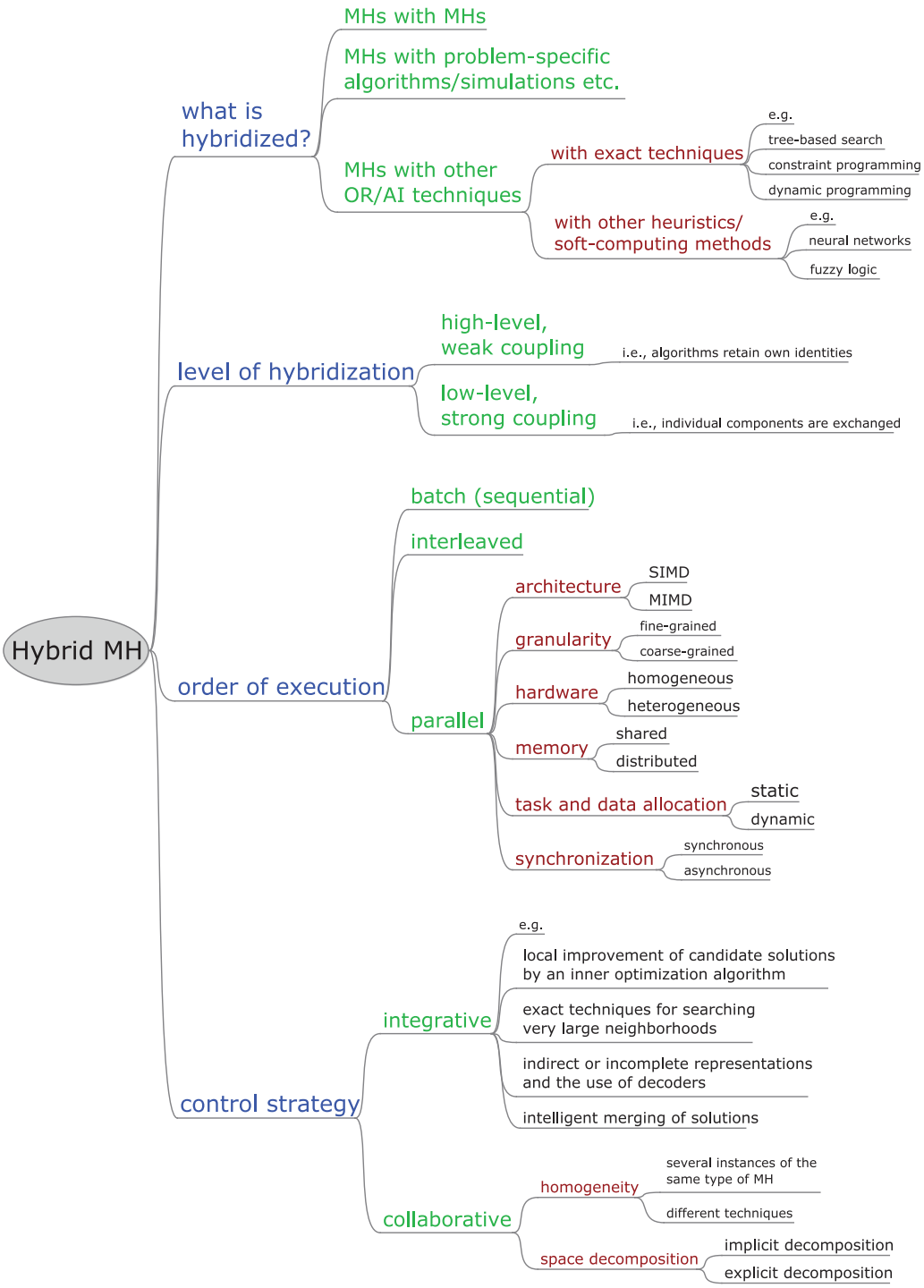


Figure 3.16: A summarized classification of hybrid metaheuristics [Raidl 2006]

is called Pareto local search (PLS). In high level relay hybrids, a self contained multiobjective metaheurstics are executed in a sequence such as using intensification strategies to improve the approximation set obtained by a population

based metaheurstics [Deb & Goel 2001]. Also, path relinking can be combined with any multiobjecive metaheurstics to fill the gaps and to intensify the search around the Pareto approximation set. Finally, in the high level cooperative class, a several multiobjective metaherstics are run in parallel and cooperative form to find the Pareto approximation set. The most work in this class is related to evolutionary algorithms [Golovkin *et al.* 2002, Meunier *et al.* 2000] and there are also works related to alternative methods as tabu search [Al-Yamani *et al.* 2002].

3.8.2.2 Combining metaheuristics with exact methods

In these schemes, the exact methods are cooperated with metaheurstics to improve their performance through either minimizing the time of obtaining the set of Pareto optimal solutions or improving the quality of the obtained solutions. The most common example is the cooperation between branch and bound approach and Genetic Algorithms to solve bi-objective flow shop problems [Basseur *et al.* 2004]. As considered in [Basseur *et al.* 2004], the cooperation between exact methods and metaheurstics can be achieved according to one of the following forms. The first one is to use metaheuristics to generate the upper bounds of the exact algorithms such as using the Pareto optimal approximation set as an upper bound to the exact algorithms. The second form is to use exact approaches to explore the very large neighborhoods which leads to reduce the search space explored by the exact algorithm through pruning the nodes when the solution in the construction is far from the initial Pareto solutions. The third form is to use exact algorithms to explore a given regions in the search space. This can be preformed by using exact algorithms to solve subproblems which generated by metaheurstics.

3.8.2.3 Combining metaheuristics with data mining

In this hybridization schemes, data mining techniques such as feature selection, association rules, clustering and classifications are incorporated into metaheuristics to improve their capabilities in tackling multiobjective optimization problems. These techniques can be applied to help the search operators to generate offspring sharing the characteristics of nondominated solutions and avoiding those dominated solutions [Jourdan *et al.* 2005]. Another important issue is to use data mining techniques to adapt parameter settings to make better convergence in metaheuristics. This can be achieved by exploiting the information gained and regarding the current ability of each operator to produce better quality solutions. There are other methods used to adjust parameters settings according to the diversity criterion [Coyne & Paton 1994]. As an example, incorporating the adaptive mutation operator in MOEAs. This can be done by using different mutation operators simultaneously and try to adapt the probability of

selecting each operator according to its performance during the search process [Basseur *et al.* 2003]. This means that the algorithm will adopt the the operator with better performance according to to the tackled problem.

3.9 Summary

This chapter presents an overview on some key stones needed to achieve our thesis objectives. At the begining, an overview of the global search and optimization techniques is provided which can be classified into two basic classes, deterministic and probabilistic. Where, probabilistic methods are more suitable to solve irregular multiobjective optimization problems that are high-dimensional, discontinuous, multimodal, and/or NP-Complete. Hence, stochastic algorithms come into play as alternative algorithms which produce high quality solutions in a reasonable amount of time. In global optimization algorithms, heuristics help to decide which one of a set of possible solutions is to be examined next. Heuristics are usually employed in deterministic algorithms to define the processing order of the candidate solutions, as done in greedy search. Whereas probabilistic methods may only consider those elements of the search space that have been selected by the heuristic in further computations. Then, an overview of metaheuristics is presented. Metaheurstics are new class of approximate algorithms which try to combine basic heuristic methods in higher level frameworks in order to efficiently and effectively explore the search space. They have been widely developed and applied to a variety of optimization problems. As approximate algorithms, metaheuristics sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. The main classification for metaheuristics is single-solution vs. population-based search. This chapter explains in details some of those search techniques. The multiobjective metaheuristic algorithm involves three additional basic search components. These components are the fitness assignment strategy, diversity preservation and elitism. Two or more metaheuristic algorithms can be combined to develop a hybrid approach better suited for a given problem. The main motivation for metaheuristic hybridization concepts is to obtain better performing systems that exploit and combine advantages of the individual pure strategies. The hybridization of Metaheuristics for multiobjective optimization field usually follows three basic schemes. The first one is the combination among different metaheuristics. The second schemes combine metaheuristics with exact methods. The third schemes combine metaheurstics with data mining techniques. In this thesis, some proposed hybrid metaheuristics for solving multiobjective optimization problems will be considered. The next chapter presents the first proposed hybrid evolutionary metaheuristic approach called HEMH.

Chapter 4

HEMH: A Hybrid Evolutionary Metaheuristics

Contents

4.1 Introduction 91

4.2 A review of GRASP and data mining 93

4.2.1 GRASP Algorithm 94

4.2.2 GRASP with Data Mining (DM-GRASP) 94

4.3 Path Relinking 95

4.4 The MOEA/D 96

4.4.1 MOEA/D Framework 96

4.4.2 MOEA/D Features 97

4.5 HEMH: Hybrid Evolutionary Metaheuristics 98

4.5.1 Motivations 98

4.5.2 The Proposed HEMH 98

4.6 Experimental design 107

4.6.1 Tested Algorithms and instances 107

4.6.2 Parameters settings 109

4.6.3 Performance Assessment metrics 111

4.7 Experimental Results 112

4.8 Summary 116

4.1 Introduction

Many of real world optimization problems can be modeled as multiobjective combinatorial optimization (MOCO) problems, which are often characterized by their large size and the presence of multiple, conflicting objectives. In general, the basic task in multiobjective optimization is the identification of the set of Pareto

optimal solutions or even a good approximation set to the Pareto front (PF). Despite progress in solving multiobjective combinatorial optimization problems exactly, the large size often means that metaheuristics are required for their solution in a reasonable time. However, multiobjective metaheuristics efficiently solve MOCO problems generating sets of approximately Pareto optimal solutions [Lamont & Veldhuizen 2002, Deb 2001]. Many of metaheuristics have been introduced in the last thirty years [Glover *et al.* 2003] such as evolutionary algorithms, simulated annealing, tabu search, variable neighborhood search, scatter search, path relinking, multistart and iterated local search, guided local search, differential evolution, ant colony optimization, particle swarm optimization and greedy randomized adaptive search procedure (GRASP)...etc. An overview of various of these methods was presented in chapter 3.

Multiobjective evolutionary algorithms (MOEAs) are a very active research area. They have recently received increased interest because they offer practical advantages in facing difficult optimization problems including their simplicity, their flexibility, and their robust response for changing circumstances. They also are able to capture several Pareto-optimal solutions in a single run. Recently, solving multiobjective optimization problems and their applications using evolutionary algorithms have been investigated by many authors [Lamont & Veldhuizen 2002, Coello 2000b, Jaszkiewicz 2003, Stewart *et al.* 2004, Farina *et al.* 2004]. Pareto dominance based algorithms such as NSGA-II [Deb *et al.* 2000] and SEPA2 [Zitzler *et al.* 2001] have been dominantly used in recent studies in this area. Based on many traditional mathematical programming methods for approximating the PF [Miettinen 1999], the approximation of the PF can be decomposed into a number of single objective optimization subproblems. Some of MOEAs adopt this idea for their fitness assignment such as MOGLS [Jaszkiewicz 2002b, Ishibuchi & Murata 1998] and MOEA/D [Zhang & Li 2007]. They optimize aggregations of the objectives with selected aggregation coefficients, which make them very easy, at least in principle, to use single objective local search for improving individual solutions. By using aggregations of the objectives, a decision maker's preference has been incorporated into MOEAs [Deb & Sundar 2006].

Many of the search algorithms attempt to obtain the best from a set of different MHs that perform together, complement each other and augment their exploration capabilities. They are commonly called *Hybrid* metaheuristics [Raidl 2006]. Intensification and diversification are the two major issues when designing a global search method [Blum & Roli 2003]. Diversification generally refers to the ability to visit many and different regions in the search space whereas, intensification refers to the ability to obtain high quality solutions within those regions. A search algorithm must balance between sometimes-

conflicting two goals. It is a complicated task for metaheuristics to provide adequate balance between intensification and diversification, but the hybrid metaheuristics can give the ability to control the balance between intensification and diversification through involving the design of hybrid metaheuristics with search algorithms specializing in intensification and diversification, which combines these types of algorithms with the objective of compensating each other and put together their complementary behaviors [Lozano & García-Martínez 2010].

This chapter tends to study the hybridization of different metaheuristics and analyze its effect on handling MOCO problems. It develops a Hybrid Evolutionary Metaheuristics (HEMH) which combines different metaheuristics integrated with each other to enhance the search capabilities. It incorporates both of DM-GRASP [Ribeiro *et al.* 2006] and Path-relinking [Glover 1996] within the MOEA/D [Zhang & Li 2007] framework. The main goals are to capture the benefits of those techniques with providing cooperation, integration and adequate balance between intensification and diversification to improve the search capabilities and to concentrate the search efforts to investigate the promising regions in the search space. This can be achieved by applying Path-relinking or reproduction operators on high quality solutions obtained by DM-GRASP. The search process is divided into two phases. In the first one, the DM-GRASP is applied to obtain an initial set of high quality solutions uniformly dispersed along the Pareto front. Then, the search efforts are intensified on the promising regions around these solutions through the second phase. The greedy randomized path-relinking with local search or reproduction operators are applied to improve the quality and to guide the search to explore the non-discovered regions in the search space. The two phases are combined with a suitable evolutionary MOEA/D framework supporting the integration and cooperation. Moreover, the efficient solutions explored over the search are collected in an external archive. The rest of the chapter is organized as follows: In section 4.2, an overview of GRASP and data mining is highlighted. In section 4.3 the Path relinking strategy is discussed, the MOEA/D framework and its features are presented in section 4.4. In section 4.5, the proposed HEMH is motivated and presented. In additions, experimental design and experimental results are involved in sections 4.6 and 4.7 respectively. Finally, summary and some directions for further research are presented in section 4.8.

4.2 A review of GRASP and data mining

In this section, a quick review of GRASP procedure and its cooperation with data mining techniques is provided in the following.

4.2.1 GRASP Algorithm

As mentioned in section 3.5.3, GRASP [Feo & Resende 1989] is a multi-start metaheuristics algorithm, which repeatedly improves starting solutions by local search. It is a two phase iterative process. The first phase of any GRASP iteration is called the greedy randomized construction, in which an initial complete solution is built. Since this solution is not guaranteed to be locally optimal, a local search strategy is performed in the second phase. This process is repeated until stopping criterion is met and the best solution found is taken as a result. The greedy randomized construction seeks to produce a divers set of good quality starting solutions from which to start local search. This can be achieved by adding randomize to the simple greedy algorithm as explained in section 3.5.3.1. Following the construction step, local search is invoked to improve starting solutions. In local search phase, two basic strategies are often considered to accept local search moves, first improvement and best improvement. The first improvement accepts the first neighbor with better quality examined as a new current solution. In contrast, the best improvement examines all neighbors and accepts the best one as a new current solution. As mentioned in section 3.5.3.2, more local search methods with good global search ability, such as simulated annealing and tabu search, etc have been suggested to improve the starting solutions [de la Peña 2004]. The whole GRASP process is summarized by the procedure in Alg. 3.3. Also, the construction and the local search procedures are described in Alg. 3.4 and Alg. 3.5 respectively.

4.2.2 GRASP with Data Mining (DM-GRASP)

In the original GRASP, iterations are performed independently from each other. Consequently, the knowledge acquired in the past iterations is not exploited in the subsequent iterations. Data mining techniques have recently begun to find their way into metaheuristics [Jourdan *et al.* 2006]. These techniques can be used to extract solutions patterns that reoccur in high quality solutions. The idea is that patterns found in good quality solutions could be used to guide the search, leading to a more effective exploration of the solution space. The basic concept of incorporating data mining in GRASP is that patterns found in the high quality solutions obtained in earlier iterations can be used to improve the search process, leading to a more effective exploration of the search space, and consequently, a cooperative behavior is achieved instead of building each solution independently. The resulting heuristic is the so-called "DM-GRASP" [Ribeiro *et al.* 2006] which has the ability to achieve promising results not only in terms of solution quality but also in terms of execution time required to obtain satisfactory solutions. DM-GRASP involves two basic phases [Santos *et al.* 2008]. The first one is re-

sponsible for generating an elite set $\mathcal{D}$ through executing pure GRASP for n iterations and selecting the best solutions found. Then, data mining is applied on $\mathcal{D}$ to extract a set of patterns $\mathcal{P}$. Next, the hybrid phase is performed in which a number of slightly different iterations are executed. In these iterations, the construction receives a pattern $p \in \mathcal{P}$ as a partial solution from which a complete solution will be built. This process is completely summarized in Alg. 4.1.

Algorithm 4.1 DM-GRASP(n)

```
Begin:
s* ← ∅; D ← ∅;
for i ← 1 to n do:                                  ▷ Apply pure GRASP
    s' ←Construction(s);
    s'' ←LocalSearchPhase(s');
    D ←UpdateEliteSet(s'');
    if (F(s'') < F(s*)) then:                       ▷ Fitness comparison
        s* ← s'';
    end if
end for
P ←PatternMining(D);                                ▷ Extract patterns from Elite set
p ←GetPattern(P);                                   ▷ Select one pattern
repeat:                                             ▷ Apply the hybrid phase
    s ←Construction(p);
    s' ←LocalSearchPhase(s);
    if (F(s') < F(s*)) then:                        ▷ Fitness comparison
        s* ← s';
    end if
until termination criterion is met
return s*;
End
```

4.3 Path Relinking

As discussed in section 3.6.2.3, Path relinking (PR) generates new solutions by exploring trajectories that connect high-quality solutions by starting from one of these solutions, called starting solution x^s, and generating a path in the neighborhood space that leads toward the other solution, called guiding solution x^t. Starting the relinking process from the best of x^s, x^t provides a better chance to investigate in more detail the neighborhood of the most promising solutions [Ribeiro *et al.* 2002]. Path-relinking represents a major enhancement to the basic GRASP procedure, leading to significant improvements in both time and solutions quality. Two basic strategies are used, in one; path-relinking is applied as intensification strategy to each local optimum obtained after the local search phase. In the other, path-relinking is applied to all pairs of elite solutions, either periodically during the GRASP iterations or after all GRASP iterations have been performed as a post-optimization step. The use of

path-relinking within a GRASP procedure, as an intensification strategy applied to each locally optimal solution, was first proposed in [Laguna & Martí 1999]. It was followed by several extensions, improvements, and successful applications [Canuto *et al.* 2001, Resende & Ribeiro 2003, Resende & Werneck 2004, Resende & Ribeiro 2005, Resende & Werneck 2006, Resende *et al.* 2010a]. In this work, path relinking will be used as an essential part of the proposed HEMH as intensification strategy to improve the performance and enhance the efficiency.

4.4 The MOEA/D

The MOEA/D [Zhang & Li 2007] is a recently developed MOEA in which the decomposition idea is applied instead of dominance relation. The MOEA/D simultaneously optimizes a set of N single objective subproblems obtained by decomposing the MOP formulated in Equ. 2.1 according to corresponding set of N uniformly distributed weight vectors $\{\Lambda^1, ..., \Lambda^N\}$ by using a scalarization technique as explained in Equ. 2.17 and Equ. 2.19. Each weight vector $\Lambda = [\lambda_1, \lambda_2, ..., \lambda_m]$ has m component corresponding to the m objectives. Equ. 4.1 depicts how Λ can be generated whereas, Equ. 4.2 defined the number of different weight vectors could be obtained.

$$\sum_{i=1}^{m} \lambda_i = 1, \forall \lambda_i \in \{0/H, 1/H, ..., H/H\}, \forall i = 1, 2, ..., m \tag{4.1}$$

$$N = \begin{pmatrix} H + m - 1 \\ m - 1 \end{pmatrix} \tag{4.2}$$

where H is a positive integer. Thus, The MOEA/D population size (N) is controlled by both H and m.

4.4.1 MOEA/D Framework

The MOEA/D framework can be explained as a cellular MOEA [Ishibuchi *et al.* 2009] with a neighborhood structure in the m-dimensional weight space. A single cell with a single individual is located at the same place as each weight vector in the m-dimensional weight space. That is, each cell has its own weight vector, which is used in the scalarizing function for evaluating the individual in that cell. Neighbors of a cell are defined by the Euclidean distance between cells in the weight space. The number of neighbors of each cell (including itself) is prespecified. The nondominated solutions obtained over the search process are maintained in external archive. To generate an offspring for a cell, two parents are randomly selected from its neighbors. The offspring is generated by crossover and mutation, which is compared with the individual in the

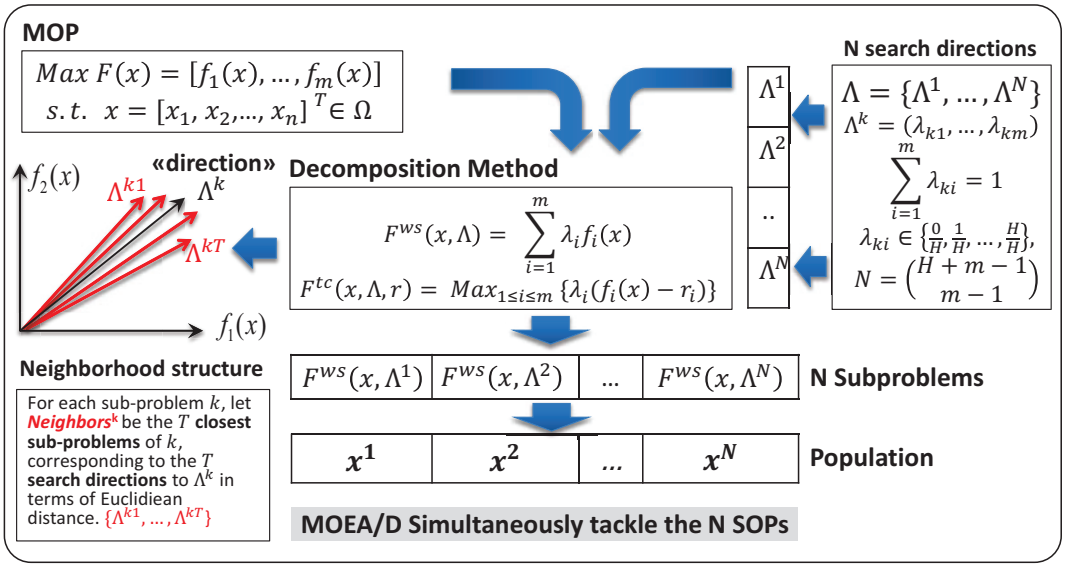


Figure 4.1: The MOEAD structure

current cell using the scalarizing function. If the offspring is better, the current individual is replaced with the offspring. The offspring is also compared with each neighbor. The scalarizing function with the weight vector of each neighbor is used in the comparison. All neighbors, which are inferior to the offspring, are replaced with the offspring (i.e., local replacement). MOEA/D has lower computational complexity at each generation than NSGA-II [Deb *et al.* 2000] and MOGLS [Ishibuchi & Murata 1998].

4.4.2 MOEA/D Features

MOEA/D framework is characterized by the following:

- MOEA/D depends on scalar optimization methods since each scalar optimization problem has its best solution found so far in the current population.
- Each individual has a different weight vector corresponding to a scalar optimization subproblem. Thus the number of the weight vectors is the same as the population size.
- Each scalar optimization problem has several neighboring subproblems.
- The optimal solutions of two neighboring subproblems should be similar.
- Each subproblem is optimized in MOEA/D by using information from its neighboring subproblems.

Based on the above framework of the MOEA/D and its features, it is clear that is the most suitable evolutionary framework to carry out the proposed hybridization with DM-GRASP and Path relinking metaheuristics to develop the proposed HEMH to enhance the performance and improve the capabilities.

4.5 HEMH: Hybrid Evolutionary Metaheuristics

In the following, the motivations of this research are discussed. Then, the HEMH applied on 0/1 MOKP will be described in more details.

4.5.1 Motivations

The Motivations of this work are expressed as follows:

- It is well known that, GRASP discovers the search space with multistart independent manner through building each solution point independently. Thus, we have the motivations of using data mining to extract good patterns form high quality solutions obtained in earlier GRASP iterations. Then these patterns is used to build new solutions for the next GRASP iterations. Hence, achieving the cooperation among GRASP iterations.
- The second motivation is related to use genetic operators for increasing the smoothness; as a result of using high quality solutions as inputs for genetic operators, we can deduce the generated offspring solutions will also have high quality.
- Incorporating path-relinking helps to discover solutions beyond elite points as a post optimization strategy. This means, on the one hand, to increase the intensification in the promising regions, on the other hand, we don't need to use a large set of predefined weight vectors (search directions) for good discovering to the Pareto front and decreasing the bad effects of scalarization techniques used in case of non-convex Pareto front shapes.
- Path relinking give the ability to investigate the non-convex regions and to discover the promising solutions lies on them due to its ability to tracing most of solutions lies on the path between starting and guiding solutions.

4.5.2 The Proposed HEMH

In this context, the HEMH approach will be described in more details. Like MOEA/D [Zhang & Li 2007], HEMH needs a decomposition technique to convert MOKP formulated in Equ. 2.7 into a set of single objective subproblems.

In this chapter, the weighted sum approach described in Equ. 2.17 was used because it performed better than weighted Tchebycheff described in Equ. 2.19 on 0/1 MOKP [Ishibuchi *et al.* 2009]. However, if we have a set of N uniformly distributed weight vectors $\{\Lambda^1, ..., \Lambda^N\}$, correspondingly, we have N single objective subproblems where the ith subproblem is formulated according to Equ. 2.17 with $\Lambda = \Lambda^i$. HEMH attempts to simultaneously optimize these N subproblems. Each subproblem has a set of subproblems located in its own neighborhood. The set of neighborhoods of the ith subproblem includes all the subproblems with the T closest weight vectors $\{\Lambda^1, ..., \Lambda^N\}$ to the weight vector Λ^i according to the Euclidean distance in the weight space.

In the next subsections, The HEMH framework is discussed with illustrations for its components and modules. Then, the whole procedure is explained.

4.5.2.1 HEMH framework

The HEMH framework contains two populations, main population and Archive. The main population consists of N members in which a solution is maintained for each search direction (subproblem) i.e. the ith member in the population represents the best solution discovered for the ith subproblem. The main population is also used to define the neighborhoods for each subproblem. The Archive collects all efficient solutions explored over the search process. It is periodically updated by new explored solutions. In HEMH, the search process consists of two basic phases, "*initialization*" and "*main loop*". Initialization is responsible for obtaining an initial set of high quality solutions dispersed into Pareto front, whereas the main loop is responsible for discovering more new solutions in the most promising regions through applying greedy randomized path-relinking or reproduction on the set of high quality solutions previously obtained in the initialization phase. Figure 4.2 depicts the HEMH flow diagram which clarifies the whole process. In the following, the two phases above will be described in more details.

Initialization phase

In this phase, DM-GRASP is applied to generate an initial set of high quality solutions that represent an initial approximation to the Pareto front to fill the main population. Firstly, original GRASP is applied on each objective function separately to construct a set of elite solutions from which a set of good patterns is extracted using data mining. Then, for each subproblem, one of the extracted patterns is selected as a partial solution to construct the current solution. DM-GRASP consists of three basic modules, Construction, LocalSearch and Pattern-Mining. The procedures of both Construction and LocalSearch were proposed in [Vianna & Arroyo 2004]. In the following,

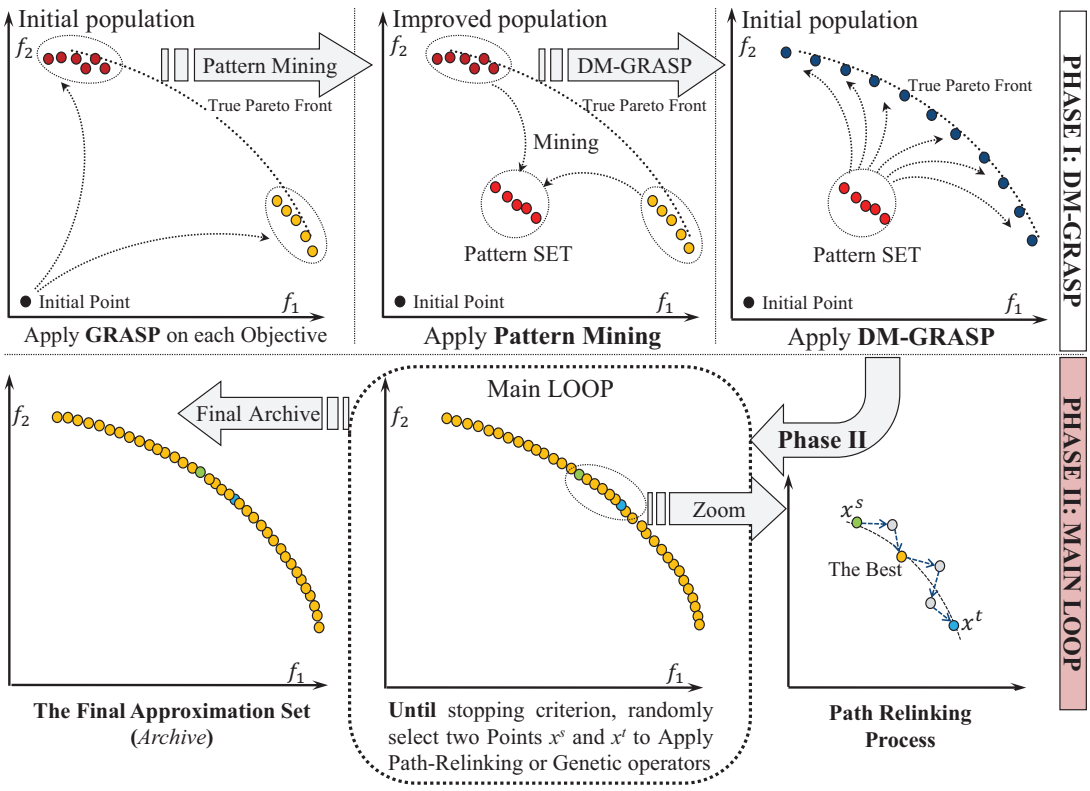


Figure 4.2: the proposed HEMH flow diagram

each of these modules that applied on MOKP will be described in details.

Construction: The construction procedure is presented in Alg. 4.2. It receives as input parameters an initial pattern p copied in the partial solution x to be built, Percentage α used in the selection of the next element to be inserted in x, the preference vector $\Lambda = [\lambda_1, ..., \lambda_m]$, and Archive to store efficient solutions. The algorithm returns the built solution x as an output. In lines (2-6), the pattern p is copied into x and the candidate list CL is initialized to an empty set. Then, CL is constructed by inserting all unselected elements (with $x_j = 0, \forall j \in \{1, ..., n\}$) sorted in decreasing order according to the ratio below:

$$\sum_{i=1}^{m} \lambda_i . c_{ij} / \sum_{i=1}^{m} w_{ij} \tag{4.3}$$

The loop in lines (7-15) is responsible for completing the partial solution x. First, the restricted candidate list (RCL) is composed of $\alpha \times |CL|$ first elements of CL as shown in line 8. Then, an item j is selected at random from RCL to be inserted into x. This process is repeated while the insertion of j does not violate any restriction of the problem. The loop in lines (16-22) completes the solution

x. Finally, Archive is updated by x and x is returned as an output (lines 23-24).

Algorithm 4.2 CONSTRUCTION($p, \alpha, \Lambda, Arch$)

Inputs:
p: *initial partial solution*
α: *parameter for building RCL*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *Search direction*
$Arch$: *List of collected efficient solutions*

1: **Begin:**
2: $x \leftarrow p$; ▷ *Initialize current solution with p*
3: $CL \leftarrow \varnothing$; ▷ *Initialize candidate list CL*
4: **while** $(\exists j : x_j = 0 \wedge max_{j=1}^{n}(\sum_{i=1}^{m} \lambda_i c_{ij} / \sum_{i=1}^{m} w_{ij}))$ **do**: ▷ *Arrange items in CL using Equ. 4.3*
5: $CL \leftarrow$ APPENDITEM(CL, j);
6: **end while**
7: **repeat**: ▷ *Begin construction process*
8: $RCL \leftarrow$ BUILDRCL(CL, α); ▷ *Select the first $\alpha \times |CL|$ elements*
9: $j \leftarrow$SELECTATRANDOM(RCL);
10: $y \leftarrow$ADDITEM(x, j);
11: **if** (y does not violate any constraint of MOKP in Equ. 2.7) **then**:
12: $x \leftarrow y$; ▷ *Update current solution x*
13: $CL \leftarrow$ REMOVEITEM(CL, j); ▷ *Remove j^{th} Element from CL*
14: **end if**
15: **until** (y violates any constraint of MOKP in Equ. 2.7)
16: **for** $i \leftarrow 1$ to $|CL|$ **do**:
17: $e \leftarrow$ Get the jth element in CL
18: $y \leftarrow$ADDITEM(x, e);
19: **if** (y does not violate any constraint of MOKP in Equ. 2.7) **then**:
20: $x \leftarrow y$; ▷ *Update current solution x*
21: **end if**
22: **end for**
23: $Arch \leftarrow$UPDATEARCHIVE(x); ▷ *Update Archive by the constructed solution*
24: **return** x;
25: **End**

Local search: In the local search step, the constructed solution x is improved through investigating its neighbors. Alg. 4.3 presents the local search procedure which uses the first improvement strategy as mentioned in [Vianna & Arroyo 2004]. It receives as input parameters a solution x to be improved, parameter β used at reconstruction process, the preference vector $\Lambda = [\lambda_1, ..., \lambda_m]$, and Archive $Arch$ to store efficient solutions obtained. It simply returns the improved solution x as an output. It begins by initializing all positions of the array $Mark$ by "false" (lines 2-4), which is important to terminate the procedure. An item e can be removed from knapsacks only if $Mark_e = false$. While there exist elements to be removed (i.e. still unmarked), the loop in lines (5-21) is executed. In line 5, a copy of solution x is copied into an auxiliary solution y. The elements that present the smallest ratio defined in Equ. 4.3 are removed from y. This process is repeated until there exist an element ℓ that

is out of knapsacks and may be inserted without violation of any restriction of the problem (lines 7-10). The construction module is invoked to complete the construction of y (line 11). If y is better than x, then x is updated by y and $Mark$ is reinitialized (lines 12-16). Thus, the process is repeated using the new solution x. If y is not better than x, then, in line 18, the first element that is removed from y through the loop in lines(7-10) is marked. Finally, Archive $Arch$ is updated by x and x is returned as an output (lines 22-23).

Algorithm 4.3 LocalSearch($x, \beta, \Lambda, Arch$)

Inputs:
x: *Solution to be refined*
β: *parameter for building RCL at reconstruction of x*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *Search direction*
$Arch$: *List of collected efficient solutions*

1: **Begin:**
2: **for** $j \leftarrow 1$ to n **do**:
3: $\quad Mark_j \leftarrow false.$ ▷ *Initialize Mark array*
4: **end for**
5: **while** $(\exists j \in \{1, ..., n\} : Mark_j = false)$ **do**:
6: $\quad y \leftarrow x;$
7: $\quad$ **repeat**:
8: $\quad\quad$ **find** $j : y_j = 1 \wedge Mark_j = false \wedge min_{j=1}^{n}(\sum_{i=1}^{m} \lambda_i.c_{ij} / \sum_{i=1}^{m} w_{ij})$
9: $\quad\quad y \leftarrow$ RemoveItem(y, j);
10: $\quad$ **until** $\exists \ell : y_\ell = 0 \wedge y \leftarrow$ FlipBit(y, ℓ) does not violate any constraint in (2.7)
11: $\quad y \leftarrow$Construction$(y, \beta, \Lambda, Arch)$; ▷ *Reconstruct the solution y*
12: $\quad$ **if** $(F^{ws}(y, \Lambda) > F^{ws}(x, \Lambda))$ **then**: ▷ *F^{ws} is the weighted sum fun. (2.17) for MOKP in (2.7)*
13: $\quad\quad x \leftarrow y;$
14: $\quad\quad$ **for** $i \leftarrow 1$ to n **do**:
15: $\quad\quad\quad Mark_i \leftarrow false.$ ▷ *Reinitialize Mark array*
16: $\quad\quad$ **end for**
17: $\quad$ **else**:
18: $\quad\quad$ **find** $j : x_j = 1 \wedge Mark_j = false \wedge min_{j=1}^{n}(\sum_{i=1}^{m} \lambda_i.c_{ij} / \sum_{i=1}^{m} w_{ij})$
19: $\quad\quad Mark_j \leftarrow true.$
20: $\quad$ **end if**
21: **end while**
22: $Arch \leftarrow$UpdateArchive(x);
23: **return** x;
24: **End**

Pattern Mining: The pattern mining procedure presented in Alg. 4.4 receives as input parameters the set of minimum supports $\sigma = \{\sigma_1, ..., \sigma_{|\sigma|}\}$ that represent the minimum ratios of repetition of an item to be included in a pattern and $\mathcal{E}$ that represents the set of high quality solutions. It returns the extracted set of patterns $\mathcal{P}$ as an output. The repetition ratio is simply calculated for each item with respect to the $\mathcal{E}$ members.

For each minimum support value $\sigma_k \in \sigma$, $\forall k \in \{1, ..., |\sigma|\}$, each item has repe-

tition ratio greater than or equal σ_k is inserted into $p_k \in \mathcal{P}$. Finally, the set of collected patterns $\mathcal{P}$ is returned as an output.

Algorithm 4.4 PATTERNMINING($\sigma, \mathcal{E}$)

Inputs:
σ: *Set of minimum supports*
$\mathcal{E}$: *Set of Elite solutions*

1: **Begin:**
2: **for** each item $j \in \{1, ..., n\}$ **do**:
3: $val_j \leftarrow 0$;
4: **for** each $y \in \mathcal{E}$ **do**: ▷ *For each elite solution y*
5: $val_j \leftarrow val_j + y_j$;
6: **end for**
7: **end for**
8: $\mathcal{P} \leftarrow \varnothing$; ▷ *Initialize set of patterns* $\mathcal{P}$
9: **for** each $k \in \{1, ..., |\sigma|\}$ **do**:
10: $p_k \leftarrow \varnothing$; ▷ *Initialize new pattern the set of patterns p*
11: **for** each item $j \in \{1, ..., n\}$ **do**:
12: **if** $(val_j \geq \sigma_k)$ **then**:
13: $p_k \leftarrow p_k \cup j^{th}\ item$;
14: **end if**
15: **end for**
16: $\mathcal{P} \leftarrow \mathcal{P} \cup p_k$; ▷ *update the set of patterns* $\mathcal{P}$
17: **end for**
18: **return** $\mathcal{P}$;
19: **End**

Main Loop phase

In this phase, greedy randomize path-relinking or reproduction by crossover and mutation is applied to the solutions previously obtained in the initialization phase to intensify the search process in the regions surrounding the Pareto front. This means, concentrating the search efforts on the promising regions to discover new high quality solutions along the Pareto frontier. In the following, different components and modules used in "*main loop*"' phase are explained in more details.

Greedy Randomized Path-relinking: The greedy randomized path-relinking procedure is presented in Alg. 4.5. It receives as input parameters the starting solution x^s, the guiding solution x^t, the search direction $\Lambda = [\lambda_1, ..., \lambda_m]$ and the parameter α that controls the balance of greediness and randomness of the next move selection. As an output, it simply returns the best intermediate solution found in the path form x^s to x^t. The procedure starts by choosing the best of x^s and x^t to start with (lines 2-4). Then, the best solution x^*, the best fitness z^*, the candidate lists CL and CL_{cmp} are initialized in lines (5-7). In lines (8-13), the candidate lists CL and CL_{cmp} are constructed. Every unmatched j

between x^s and x^t with $x^s_j = 0$ is inserted into CL in descending order according to formula in Equ. 4.3, whereas every unmatched j between x^s and x^t with $x^s_j = 1$ is inserted into CL_{cmp} in increasing order according to formula in Equ. 4.3.

Algorithm 4.5 GreedyRandomPathRelink($x^s, x^t, \alpha, \Lambda, Arch$)

Inputs:
x^s, x^t: *Starting and Guiding Solutions*
$\alpha \in [0,1]$: *Parameter to control greediness/randomness move selection*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *Search direction*
Arch: *List of collected efficient solutions*

1: **Begin:**
2: **if** $(F^{ws}(x^t, \Lambda) > F^{ws}(x^s, \Lambda))$ **then**: ▷ *F^{ws} is the weighted sum fun. (2.17) for MOKP in (2.7)*
3: Swap(x^s, x^t); ▷ *Select the best of x^s and x^t to start with*
4: **end if**
5: $x^* \leftarrow x^s$; ▷ *Initialize the best solution found so far*
6: $z^* \leftarrow F^{ws}(x^s, \Lambda)$; ▷ *$F^{ws}$ is the weighted sum fun. (2.17) for MOKP in (2.7)*
7: $CL \leftarrow \varnothing$; $CL_{cmp} \leftarrow \varnothing$; ▷ *Initialize CL and CL_{cmp}*
8: **while** $(\exists j : x^s_j \neq x^t_j \wedge x^s_j = 0 \wedge max^n_{j=1}(\sum_{i=1}^m \lambda_i c_{ij} / \sum_{i=1}^m w_{ij}))$ **do**: ▷ *Construct CL*
9: $CL \leftarrow$ Append(j);
10: **end while**
11: **while** $(\exists j : x^s_j \neq x^t_j \wedge x^s_j = 1 \wedge min^n_{j=1}(\sum_{i=1}^m \lambda_i c_{ij} / \sum_{i=1}^m w_{ij}))$ **do**: ▷ *Construct CL_{cmp}*
12: $CL_{cmp} \leftarrow$ Append(j);
13: **end while**
14: $x \leftarrow x^s$; ▷ *Start relinking process*
15: **repeat**:
16: $RCL \leftarrow$ Build(CL, α); ▷ *The first $\alpha \times |CL|$ elements of CL*
17: $\Delta(x, x^t) \leftarrow |\{j \in \{1, ..., n\} : x_j \neq x^t_j\}|$; ▷ *Compute Hamming distance between x, x^t*
18: **if** x does not violate constrains in (2.7) **then**:
19: $\ell^* \leftarrow$SelectAtRandom(RCL);
20: $CL \leftarrow$ Remove(CL, ℓ^*);
21: **else**:
22: $\ell^* \leftarrow$ GetTheFirstElement(CL_{cmp});
23: $CL_{cmp} \leftarrow$ Remove(CL_{cmp}, ℓ^*);
24: **end if**
25: $x \leftarrow$ FlipBit(x, ℓ^*);
26: $y \leftarrow$GreedyRepair(x, Λ);
27: **if** $(F^{ws}(y, \Lambda) > z^*)$ **then**: ▷ *F^{ws} is the weighted sum fun. (2.17) for MOKP in (2.7)*
28: $x^* \leftarrow y$;
29: $z^* \leftarrow F^{ws}(y, \Lambda)$
30: **end if**
31: **until** $(\Delta(x, x^t) = 1)$
32: **if** $(x^* \neq x^s)$ **then**:
33: $x^* \leftarrow$LocalSearch($x^*, \beta, \Lambda, Arch$); ▷ *Apply local search*
34: **end if**
35: **return** x^*;
36: **End**

The relinking process started in line 14 in which the intermediate solution x is initialized by the starting solution x^s. Now, the procedure is ready to build the path connects x^s with x^t gradually by creating intermediate solutions through

execution of the loop in lines 15-31. Initially, the RCL is composed of $\alpha \times |CL|$ first elements of CL as shown in line 16. Then, the Hamming distance $\Delta(x, x^t)$ which represents the number of unmatched items between x and x^t is calculated in line 17. The next move is carried out by selecting one of unmatched l^* to be matched, if the intermediate x is feasible, then, l^* is randomly extracted from RCL, otherwise the first element of CL_{cmp} is extracted to be l^* (lines 18-24). The new intermediate solution x is obtained by flipping the item x_{l^*} corresponding to the selected index l^* in the current intermediate solution x (line 25). If the current x is infeasible, the GreedyRepair module is invoked on it to obtain a feasible solution y (line 29). In line 31-34, the best solution x^* and the best fitness z^* are updated by y. This process is repeated until there exist only one unmatched item between the current intermediate solution x and guiding solution x^t. In lines 32-34, Local search is invoked to improve the best solution found in the path x^* only if $x^* \neq x^s$ to guarantee applying local search only on a new solution. Finally, the improved x^* is simply returned as an output.

Reproduction: Reproduction means to generate offspring from parent individuals using crossover and mutation. Crossover is performed by exchanging specified parts of parents with each other, whereas mutation is performed by filliping specified components of the parent individual. There are several strategies to carry out crossover and mutation depending upon the representation as mentioned in section 3.6.1.1. Here, the single point crossover and binary mutation are adopted. Figures 3.7 and 3.8 illustrate the single point crossover and mutation used in binary representation respectively.

Greedy Repair: In this module, the infeasible solution x is repaired to be feasible. Alg. 4.6 implements the GreedyRepair procedure which takes the solution to be repaired x and the weight vector Λ as input parameters. The main idea is to remove the item which has the minimum value of the formula in Equ. 4.3 from the infeasible solution x. This process is repeated until x becomes feasible. Then, x is returned as an output. If the solution x is already feasible, then x is returned without change.

Update solutions: The UpdateSolutions procedure presented in Alg. 4.7 takes three parameters, the solution y, the number of solutions must be updated t and the range of the main population from which solutions are selected to be updated M. The procedure starts with selecting a solution x^j from Pop for update. Then, x^j is compared with y according to the weight sum function in Equ. 2.17 using the weight vector Λ^j, if y is better than x^j, then, x^j is replaced by y and deleted from M. This process is repeated until t iterations or M will be empty.

Algorithm 4.6 GreedyRepair(x, Λ)

Inputs:
x: *Solution to be repaired*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *Search direction*

1: **Begin:**
2: $y \leftarrow x$;
3: **while** y violates any constraint of MOKP in Equ. 2.7 **do**:
4: find $j : y_j = 1 \wedge min_{j=1}^{n}(\sum_{i=1}^{m} \lambda_i.c_{ij} / \sum_{i=1}^{m} w_{ij})$
5: $y \leftarrow$ RemoveItem(y, j);
6: **end while**
7: **return** y;
8: **End**

Algorithm 4.7 UpdateSolutions(y, t, M)

Inputs:
y: *The offspring solution*
t: *Max. replaced solutions*
M: *Updating Range*

1: **Begin:**
2: $c \leftarrow 0$;
3: **repeat**:
4: $j \leftarrow$SelectAtRandom(M);
5: **if** $(F^{ws}(y, \Lambda^j) > F^{ws}(x, \Lambda^j))$ **then**: ▷ *F^{ws} is the weighted sum fun. (2.17) for MOKP in (2.7)*
6: $x^j \leftarrow y$; ▷ *update x^j by y*
7: $c \leftarrow c + 1$; ▷ *update counter c*
8: $M \leftarrow M \backslash \{j\}$; ▷ *Remove j from M*
9: **end if**
10: **until** $(c = t \vee M = \varnothing)$
11: **End**

4.5.2.2 HEMH procedure

As shown in Alg. 4.8, the proposed HEMH procedure is presented. HEMH procedure receives a list of input parameters including the number of subproblems considered (population size) N, the size of neighborhood for each subproblem T, the maximum number of updated solutions t, the probability of selecting parents from neighborhoods δ, the parameter used in the construction step α, the parameter used in the local search process β, the minimum support which used in the pattern mining process σ and the minimum hamming distance allowed to apply path-relinking ε. The procedure starts with initializing a set of N uniformly distributed weight vectors $\{\Lambda^1, ..., \Lambda^N\}$, each one represents the search direction for a single objective subproblem. Then, HEMH constructs the neighborhood structure for each ith subproblem through calculating the Euclidean distance between the weight vector Λ^i and each one of the set of all weight vectors $\{\Lambda^1, ..., \Lambda^N\}$ and choosing the T closest subproblems. In the *initialization* phase, the initial population members are initialized using DM-GRASP. Firstly, pure GRASP

is applied on each objective function separately collecting elite solutions in the archive. Then, pattern mining is applied on the archive to extract the set of patterns $\mathcal{P}$. For each population member, a randomly selected pattern $p \in \mathcal{P}$ is assigned as an input to the CONSTRUCTION procedure to build a complete solution on which the LOCALSEARCH procedure is applied. The resulting solution is the ith member in the initial population. In the *main-loop* phase, the search process is intensified on the promising regions surround the solutions previously obtained in the initialization phase. This can be done through applying the greedy randomized path-relinking or even reproduction (crossover and mutation). For each ith subproblem, The mating/updating range M is determined to be either its neighborhood (Local) with probability equals to δ, or the whole population (Global). To generate a new offspring y, two parent individuals x^j and x^k are randomly selected from M. Then, Hamming distance $\Delta(x^j, x^k)$ is calculated. The greedy randomize path-relinking is applied to generate y only if $\Delta(x^j, x^k) \geq \varepsilon$. Otherwise, reproduction by crossover and mutation is considered. If y is infeasible, the GREEDYREPAIR module is invoked to make it feasible. Now, the offspring y is used to update the current population through updating t solutions from mating/updating range M including f the current solution of the ith subproblem. The UPDATESOLUTIONS module is invoked to perform this task. The Archive is also updated by every generated offspring. The whole process is repeated until stopping criterion is met. Finally, the Archive is returned as an output. The whole process of the proposed HEMH is summarized in the flowchart illustrated in Fig. 4.3.

4.6 Experimental design

In this section, the proposed HEMH is verified and tested to approve its efficiency and effectiveness against some of the other MOEAs in solving MOKP. HEMH was implemented and compiled using Microsoft visual C++ version 6. All of these experiments have been performed on PC workstation with (2 CPUs) Intel X5670 2.93 GHz and 16 GB of RAM. The experimental settings will be discussed in the following.

4.6.1 Tested Algorithms and instances

In order to verify the performance of HEMH, some of the state-of-the-art MOEAs are considered in this study such as NSGAII [Deb *et al.* 2000], SPEA2 [Zitzler *et al.* 2001], GRASPM [Vianna & Arroyo 2004] and MOEA/D [Zhang & Li 2007]. NSGAII and SPEA2 which represent the tow most popular Pareto-based MOEAs are explained in details in section 3.7.4.2. GRASPM is a multiobjective extension of the original GRASP proposed

Algorithm 4.8 HEMH$(N, T, t, \delta, \alpha, \beta, \sigma, \varepsilon)$

Inputs:
N: *Population size or number of subproblems*
T: *Neighborhood size for each weight vector*
$t \leq T$: *Maximum number of replaced solutions*
$\delta \in [0,1]$: *Probability of selecting parents from neighborhood*
$\alpha \in [0,1]$: *Parameter for building RCL used in GRASP*
$\beta \in [0,1]$: *Parameter for local search*
σ: *Set of Minimum support*
ε: *Minimal hamming distance for applying Path-relinking*

Begin: ▷ *Initialization Phase by DMGRASP*
$W_v \leftarrow \{\Lambda^1, ..., \Lambda^N\}$; ▷ *Generate a set of N uniformly distributed weight vectors*
for $i \leftarrow 1$ to N **do**: ▷ *Neighborhood construction for each subproblem*
  $Neighbors_i \leftarrow \{i1, ..., iT\}$; ▷ *where $\Lambda^{i1}, ..., \Lambda^{iT}$ are the T closest weight vectors to Λ^i*
end for
for $i \leftarrow 1$ to m **do**: ▷ *Apply pure GRASP on each objective separately*
  $s \leftarrow \varnothing$;
  $s' \leftarrow$ Construction$(s, \alpha, \Lambda_{f_i}, Arch)$;
  $s'' \leftarrow$ LocalSearch$(s', \beta, \Lambda_{f_i}, Arch)$;
  $Arch \leftarrow$ UpdateArchive(s'');
end for
$\mathcal{P} \leftarrow$ PatternMining$(\sigma, Arch)$; ▷ *Construction of Pattern set*
for $i \leftarrow 1$ to N **do**: ▷ *Apply DM-GRASP to get an initial population*
  $p \leftarrow$ SelectAtRandom$(\mathcal{P})$; ▷ *Select a pattern p*
  $x^i \leftarrow$ Construction$(p, \alpha, \Lambda^i, Arch)$; ▷ *$\Lambda^i$ is the weight vector correspond to i^{th} objective*
  $x^i \leftarrow$ LocalSearch$(x^i, \beta, \Lambda^i, Arch)$;
  $P \leftarrow$ AddSubProblem(x^i, Λ^i); ▷ *Add i^{th} subproblem to the population P*
end for
repeat: ▷ *Main Loop Phase*
  for $i \leftarrow 1$ to N **do**:
    $r \leftarrow$ Random(0,1); ▷ *Generate a random number $\in [0,1]$*
    if $(r < \delta)$ **then**: ▷ *Determine the mating/updating range M for each ith subproblem*
      $M \leftarrow Neighbors_i$;
    else:
      $M \leftarrow \{1, 2, ..., N\}$;
    end if
    $j, k \leftarrow$ SelectAtRandom(M); ▷ *Randomly select two parents*
    if $(\Delta(x^j, x^k) < \varepsilon)$ **then**: ▷ *Hamming distance between x^j and x^k*
      $y \leftarrow$ Reproduction(x^j, x^k); ▷ *Apply 1-point crossover followed by mutation*
      $y \leftarrow$ GreedyRepair(y, Λ^i);
    else:
      $y \leftarrow$ GreedyRandomPathRelink$(x^j, x^k, \alpha, \Lambda^i, Arch)$;
    end if
    $P \leftarrow$ UpdateSolutions(y, t, M);
    $Arch \leftarrow$ UpdateArchive(y);
  end for
until Stopping criterion is satisfied
return $Arch$; ▷ *Return the list of efficient solutions collected*
End

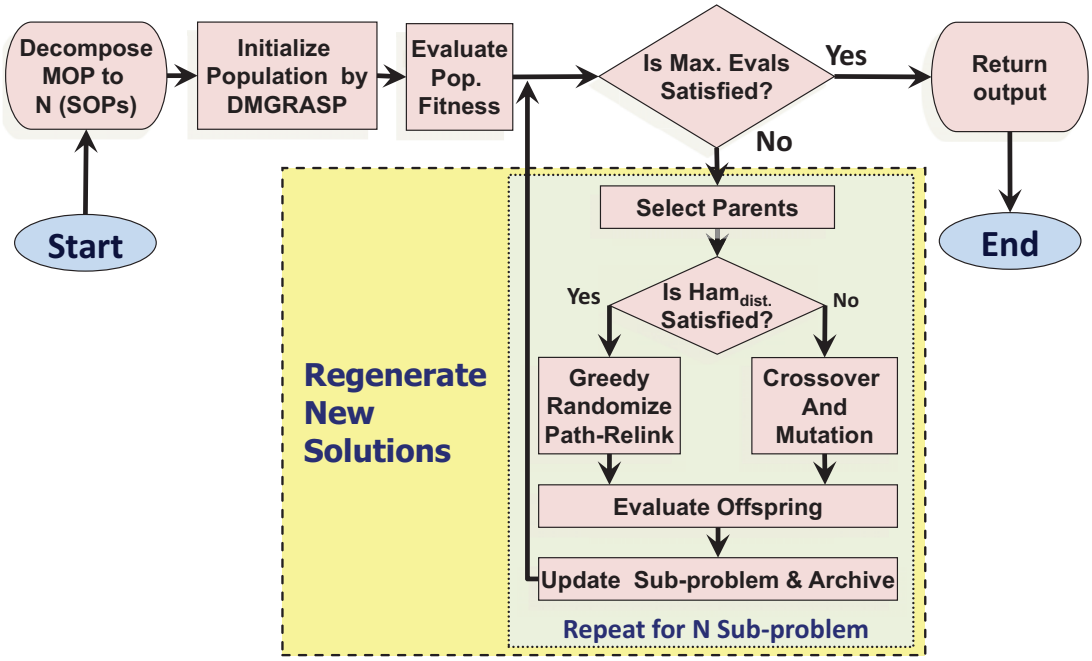


Figure 4.3: the proposed HEMH flowchart

to tackle the MOKP. It adopts the scalarzation techniques to convert the orignal MOKP to a set of single objective knapsack problem. The MOEA/D is the most popular decomposition based MOEAs as discussed in section 4.4. Besides, the set of MOKP test instances listed below in Table 4.1 are commonly used in the comparison of multiobjective metaheuristics. It has been generated by Zitzler & Thiele in [Zitzler & Thiele 1999]. Since that, this set of test instances has become a standard benchmark that has been solved by many other researchers [Bandyopadhyay *et al.* 2008, Zhang & Li 2007, Deb *et al.* 2000] and widely used in testing multiobjective metaheuristics. In this work, this set is adopted to perform the experimental design to verify and test the proposed HEMH.

4.6.2 Parameters settings

In this section, the different parameters used for each MOEA are discussed. For MOEA/D, the parameter (H) which controls the number of weight vectors or also the population size (N), is determined with its corresponding (N) for each problem instance in table 4.1 above according to the complexity. Thus, all of MOEA/D, NSGA-II and SPEA2 use the same population size (N), whereas GRASPM uses (N) as the number of weight vectors. in HEMH, a small population size should be used to encourage path-relinking usage

Table 4.1: The set of used knapsack test instances

Instance			$N(H)$	HEMH $N(H)$	$MaxEvals$
Name	Knaps(m)	Items(n)			
KSP252	2	250	150(149)	75(74)	75000
KSP502	2	500	200(199)	100(99)	100000
KSP752	2	750	250(249)	125(124)	125000
KSP253	3	250	300(23)	153(16)	150000
KSP503	3	500	300(23)	153(16)	150000
KSP753	3	750	300(23)	153(16)	150000
KSP254	4	250	364(11)	165(8)	182000
KSP504	4	500	364(11)	165(8)	182000
KSP754	4	750	364(11)	165(8)	182000

instead of reproduction by crossover and mutation. the values of (H) and the corresponding values of (N) used in HEMH for each problem instance is also listed in table 4.1. The initial populations used in NSGA-II, SPEA2 and MOEA/D are randomly generated from the decision space such that each solution $x = (x_1, ..., x_n)^T \in \{0, 1\}^T$, where $x_i = 1$ with probability equals to 0.5. The maximum number of function evaluations ($MaxEvals$) depicted in table 4.1 is used as a stopping criterion for each MOEA. In both HEMH and GRASPM in which the local search strategy is adopted, each fitness comparison performed inside the local search procedure is considered as a function evaluation for fair comparison. For each compared MOEA, all efficient solutions encountered over the generations are collected in the archive. In these experiments, the same reproduction operator which combines the single-point crossover and the standard mutation is considered. Crossover is preformed with probability equals to 1, whereas mutation is performed for each item independently with probability equals to $\frac{1}{n}$. In both NSGAII and SPEA2, tournament selection is used with tournament size equals to 2. The other control parameters used in these experiments are listed in table 4.2. Finally, the statistical analysis is applied on 30 independent runs for each MOEA on each test instance.

Table 4.2: Set of common parameters used

Parameters	**Algorithms**		
	MOEA/D	GRASPM	HEMH
Neighborhood size: T	10	-	10
Max. no. of replaced solutions: t	-	-	5
RCL definition parameter: α	-	0.1	0.1
Reconstruction parameter: β	-	0.5	0.5
Set of minimum support: σ	-	-	$\{1\}$
Parents selection probability: δ	-	-	0.9
Minimal Hamming distance: ε	-	-	10

4.6.3 Performance Assessment metrics

Due to the nature of MOP, multiple performance indicators should be used in order to investigate the performance of the compared MOEA in terms of both convergence and diversity. In these experiments, two types of indicators are used. The first one is the binary indicators which are used to compare each pair of MOEAs such as set coverage I_C indicator which is discussed in details in section 2.6.2.1. The second type is the unary indicators which are used to assess each MOEA independently of the others such as hypervolume (I_{Hyp}), generational distance (I_{GD}), inverted generational distance (I_{IGD}) and Maximum spread (I_{MS}) indicators. These indicators are presented in sections 2.6.2.2, 2.6.2.3, 2.6.2.3 and 2.6.2.5 respectively.

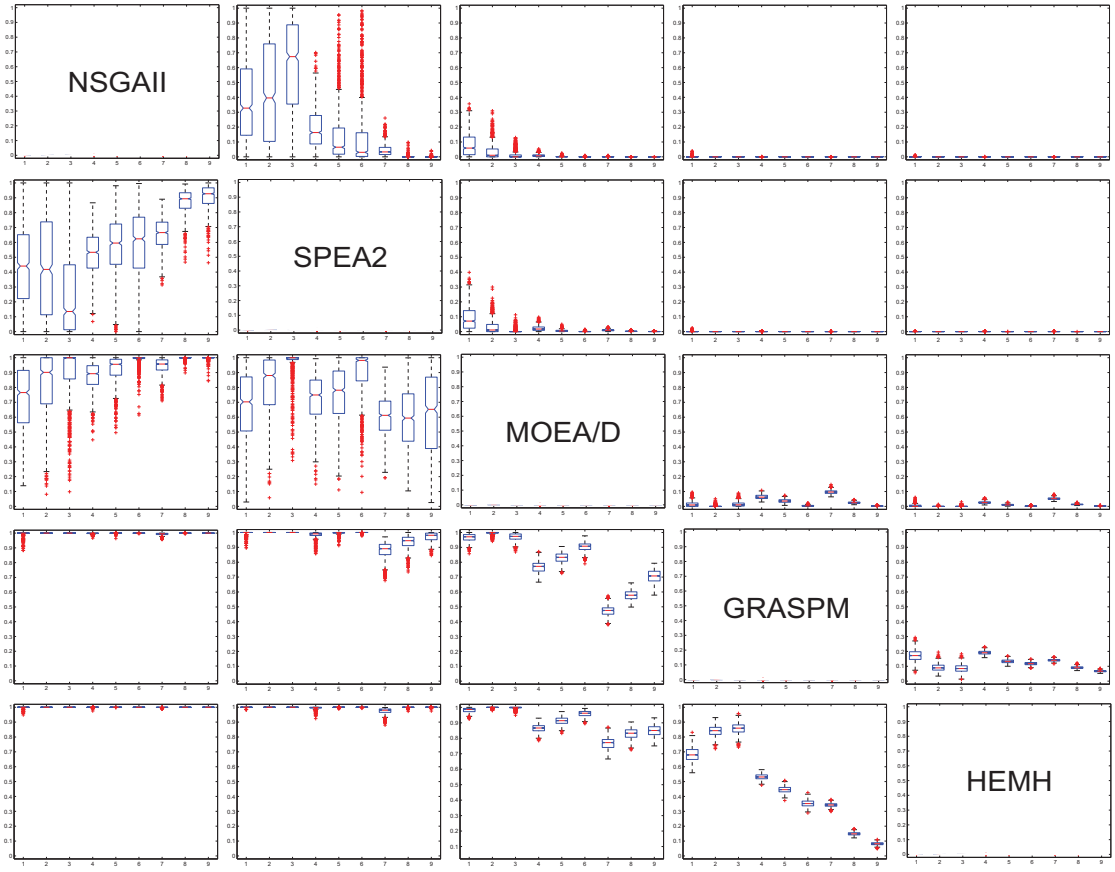


Figure 4.4: Results of the coverage indicator I_C

In these experiments, the reference set $P^*$is alternatively formed for each problem instance by gathering all of non-dominated solutions found by all of the compared MOEAs. The reference point r^* used in hypervolume I_{Hyp} calculations is chosen to be the origin. In all experimental results here, the values of all of these metrics are evaluated after normalizing all the approximation sets in the objective space

in the range [1,2].

4.7 Experimental Results

Here, the different simulation results are shown in details. Firstly, Fig. 4.4 shows the results of the set coverage (I_C) indicator. It contains a chart (with scale 0 at the bottom and 1 at the top) for each ordered pair of the compared MOEAs. Each chart consists of nine box plots representing the distribution of I_C values. Each box plot (from left to right) represents an instance in table 4.1 (from top to down) respectively. A chart located in the raw of algorithm A and the column of algorithm B presents the values of coverage of approximations generated by algorithm B by approximations generated by algorithm A. Additionally, the median values of I_C indicator are listed in table B.1. According to these results, It is clear that HEMH and GRASPM outperform the rest MOEAs. It is also clear that HEMH performs better or even slightly better than GRASPM for all test instances.

Table 4.3: Results of the Hypervolume indicator (I_{Hyp})

Instance	**Algorithm**				
	NSGAII	SPEA2	MOEA/D	GRASPM	HEMH
KSP252	6.680E-01	6.576E-01	7.763E-01	7.948E-01	**7.976E-01**
KSP502	5.889E-01	5.842E-01	7.492E-01	7.710E-01	**7.757E-01**
KSP752	5.516E-01	5.469E-01	7.540E-01	7.702E-01	**7.751E-01**
KSP253	4.129E-01	3.994E-01	5.342E-01	5.538E-01	**5.580E-01**
KSP503	3.175E-01	3.070E-01	4.982E-01	5.247E-01	**5.308E-01**
KSP753	2.665E-01	2.599E-01	4.861E-01	5.211E-01	**5.270E-01**
KSP254	2.122E-01	2.094E-01	3.334E-01	3.502E-01	**3.553E-01**
KSP504	1.325E-01	1.498E-01	2.922E-01	3.235E-01	**3.306E-01**
KSP754	9.766E-02	1.145E-01	2.666E-01	3.124E-01	**3.216E-01**

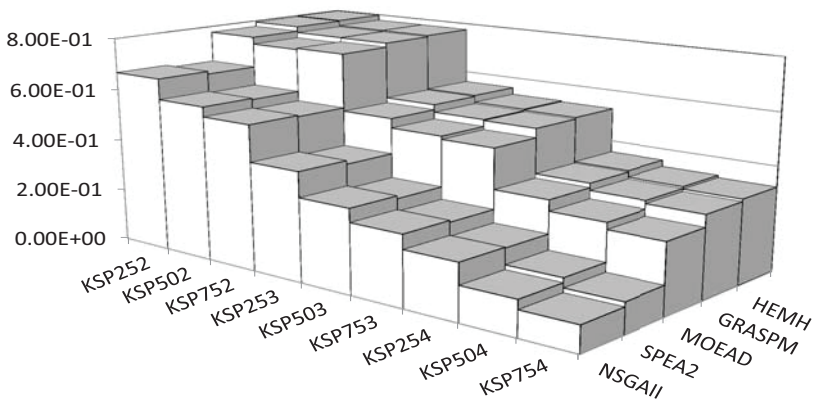


Figure 4.5: Results of the Hypervolume indicator I_{Hyp}

The experimental results of the hypervolume I_{Hyp} indicator are listed in table 4.3 which contains the average values of the indicator achieved over 30 independent runs. These results are also visualized in Fig. 4.5. The detailed statistical analysis of I_{Hyp} are provided in table B.2. According to these results, it is clear that the HEMH outperforms all the compared MOEAs. Since, it has the maximum average I_{Hyp} values. This indicates the ability to improve both convergence and diversity. Whereas, GRASPM and MOEAD have the second and the third rank respectively in all test instances.

In table 4.4, the average values of the generational distance I_{GD} indicator are listed. Additionally, Fig. 4.6 visualizes these results. The detailed statistical analysis for I_{Hyp} are also provided in table B.3. According to the generational distance results, the HEMH outperforms all the compared MOEAs. The GRASPM algorithm achieves the second rank followed by the MOEAD which takes the third rank with respect to all test instances. This means that the HEMH has the capabilities of discovering solutions as near as possible to the Pareto front.

Table 4.4: Results of the Generational distance indicator (I_{GD})

Instance	**Algorithm**				
	NSGAII	SPEA2	MOEA/D	GRASPM	HEMH
KSP252	3.240E-03	3.142E-03	1.457E-03	4.020E-04	**2.307E-04**
KSP502	4.424E-03	4.555E-03	1.458E-03	3.500E-04	**1.747E-04**
KSP752	4.171E-03	4.993E-03	1.009E-03	2.889E-04	**1.462E-04**
KSP253	1.622E-03	1.377E-03	4.457E-04	1.771E-04	**1.261E-04**
KSP503	2.369E-03	1.984E-03	4.468E-04	1.312E-04	**9.126E-05**
KSP753	3.345E-03	2.912E-03	4.760E-04	1.041E-04	**7.739E-05**
KSP254	1.538E-03	1.042E-03	2.849E-04	1.516E-04	**1.140E-04**
KSP504	2.571E-03	1.576E-03	3.203E-04	9.534E-05	**8.983E-05**
KSP754	3.173E-03	2.017E-03	4.009E-04	8.047E-05	**6.864E-05**

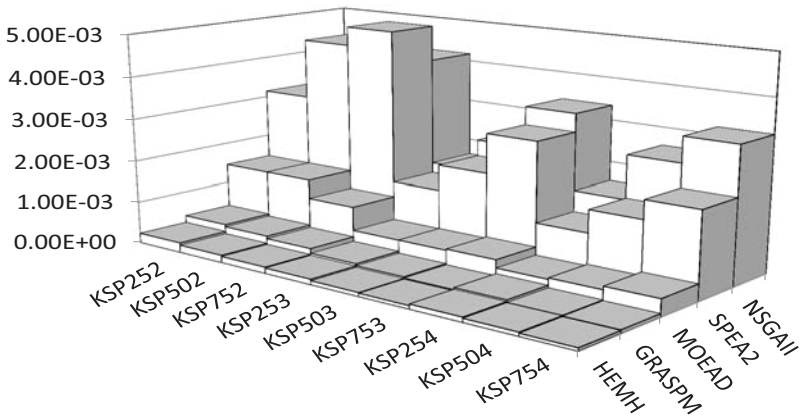


Figure 4.6: Results of the Generational Distance indicator I_{GD}

The results of the inverted generational distance I_{IGD} comparisons are listed in table 4.5 below, which contains the average values of the IGD-indicator over 30 independent runs. Also, Fig. 4.7 visualizes these results. The detailed statistical analysis for I_{IGD} are also provided in table B.4. From these results, It is clear that the HEMH outperforms all of the rest MOEAs, which reflects its ability to obtain solutions with good spread over the Pareto Frontier. The results also indicate that the GRASPM achieves the second rank followed by the MOEAD which take the third rank with respect to all test instances.

Table 4.5: Average results of the Inverted generational distance (I_{IGD})

Instance	**Algorithm**				
	NSGAII	SPEA2	MOEA/D	GRASPM	HEMH
KSP252	7.899E-03	8.608E-03	8.094E-04	3.468E-04	**3.161E-04**
KSP502	8.438E-03	8.595E-03	8.236E-04	2.467E-04	**1.717E-04**
KSP752	8.295E-03	8.126E-03	5.864E-04	2.055E-04	**1.378E-04**
KSP253	1.007E-03	1.153E-03	1.921E-04	9.910E-05	**8.606E-05**
KSP503	1.028E-03	1.143E-03	1.791E-04	9.015E-05	**7.263E-05**
KSP753	1.045E-03	1.124E-03	1.673E-04	8.099E-05	**6.300E-05**
KSP254	3.838E-04	4.127E-04	1.075E-04	8.232E-05	**7.264E-05**
KSP504	3.899E-04	3.968E-04	1.037E-04	7.686E-05	**6.203E-05**
KSP754	4.040E-04	4.081E-04	1.084E-04	7.057E-05	**5.611E-05**

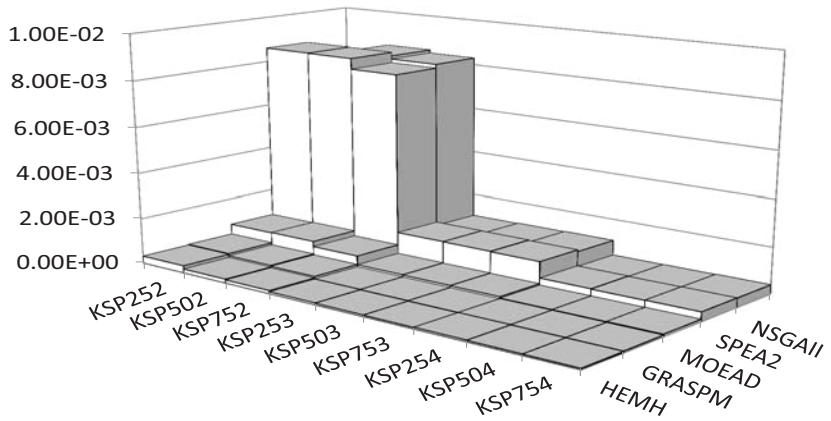


Figure 4.7: Average results of the Inverted Generational Distance I_{IGD}

Table 4.6 and Fig. 4.8 below show the average values of the maximum spread I_{MS} indicator for all test instances. Also, The detailed statistical analysis for I_{MS} are provided in table B.5. Based on these results, the HEMH has the superiority over other MOEAs, followed by GRASPM. This assures their capabilities of exploring the extreme regions in the search space due to the local search used in both of them, which intensify the search on extremes.

Table 4.6: Results of the Maximum Spread indicator (I_{MS})

Instance	Algorithm				
	NSGAII	SPEA2	MOEA/D	GRASPM	HEMH
KSP252	5.168E-01	4.705E-01	1.373E+00	1.360E+00	**1.374E+00**
KSP502	3.788E-01	3.678E-01	1.309E+00	1.371E+00	**1.393E+00**
KSP752	2.598E-01	2.736E-01	1.317E+00	1.354E+00	**1.367E+00**
KSP253	8.916E-01	7.604E-01	1.650E+00	1.677E+00	**1.702E+00**
KSP503	6.653E-01	5.536E-01	1.653E+00	1.703E+00	**1.708E+00**
KSP753	4.758E-01	3.851E-01	1.644E+00	1.713E+00	**1.725E+00**
KSP254	1.234E+00	9.954E-01	1.903E+00	1.944E+00	**1.981E+00**
KSP504	1.066E+00	7.832E-01	1.902E+00	1.975E+00	**1.985E+00**
KSP754	8.273E-01	5.803E-01	1.838E+00	1.958E+00	**1.960E+00**

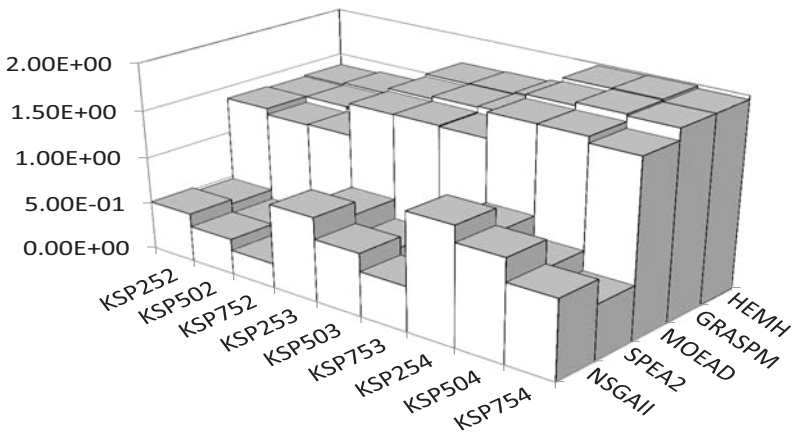


Figure 4.8: Results of the Maximum Spread indicator I_{MS}

In figures 4.9, 4.10 and 4.11, the approximation sets obtained by each MOEA are visualized for bi-objective instances KSP252, KSP502 and KSP752 respectively . Each figure contains 2 scatter graphs. The big one depicts the whole approximation sets whereas the small one in the left bottom corner focused on the part bounded by the small rectangle. In Fig. 4.9, HEMH and GRASPM achieves nearly the same points. From Fig. 4.10 and Fig. 4.11, it is clear that the solutions obtained by HEMH have the best quality. It is also noted from figures that the quality of solutions obtained by HEMH is slightly increased gradually as the size of instance increased. This can be explained as, the larger the size of instance is, the more chance of hamming distance between any two selected solutions to increase. Consequently, path-relinking has more chance to be invoked instead of reproduction. This reflects the role played by path-relinking in improving the search capabilities of HEMH.

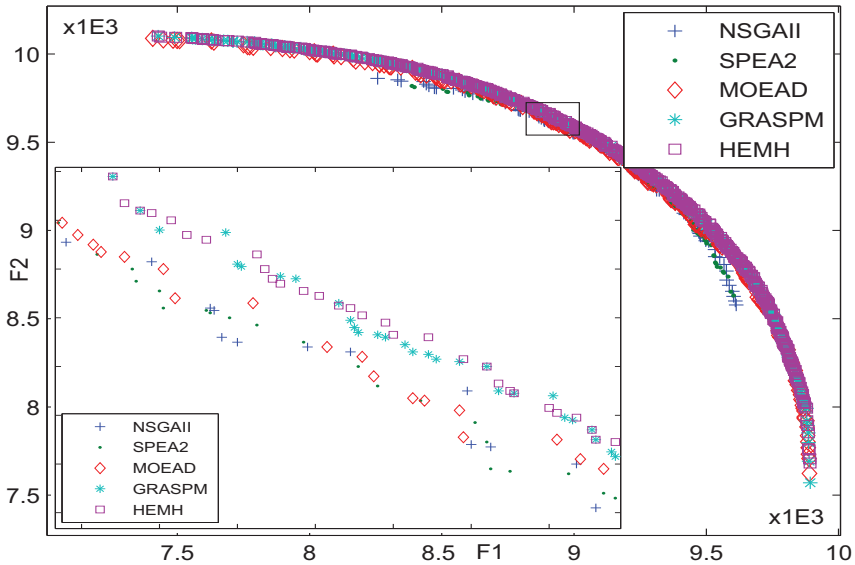


Figure 4.9: The obtained Pareto approximation set for KSP252 Instance

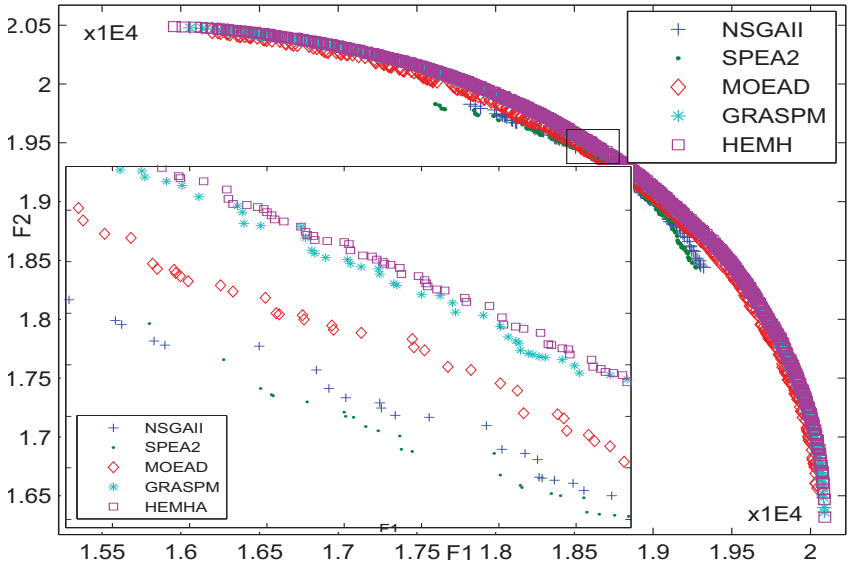


Figure 4.10: The obtained Pareto approximation set for KSP502 Instance

4.8 Summary

In this chapter, a hybrid evolutionary metaheuristics (HEMH) based on DM-GRASP and greedy randomize path-relinking to solve multiobjective knapsack problems is presented. The proposed HEMH is verified using a set of test instances commonly used in the literature. The HEMH is compared with four of the most popular MOEAs that considered as the state-of-the art of the field. A set of quality assessment indicators is also considered to evaluate the performance for all the compared MOEAs. The experimental results indicate the superiority

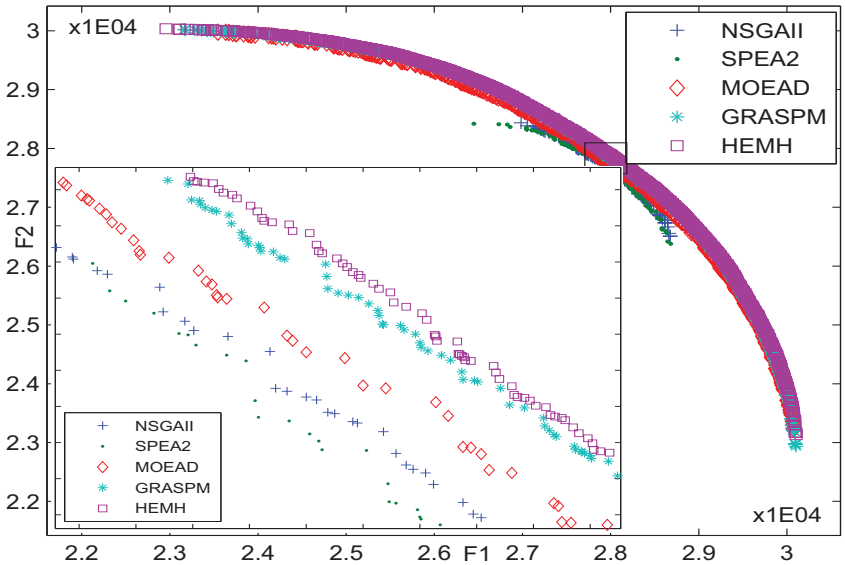


Figure 4.11: The obtained Pareto approximation set for KSP752 Instance

of the decomposition based MOEAs over the Pareto dominance based MOEAs. They also indicate the superiority of local search based MOEAs especially the HEMH. Since, it has an average performance highly competitive with respect to the compared MOEAs based on the assessment indicators used in the study. The main contribution of the proposed HEMH is the combination among different metaheuristics techniques that intensify the search process in discovering the most promising regions in the search space and enhance the ability to explore high quality solutions. The second contribution is the ability to find a good approximation set of high quality solutions using a small set of uniformly distributed search directions due to the use of path-relinking and local search strategies. In the next chapter, we tend to investigate how to enhance the search process proposed in second phase of HEMH through exploring the results of combining different recombination operators within the MOEA/D framework regardless of the quality of the solutions obtained by the first phase.

Chapter 5

Hybrid Metaheuristics based on MOEA/D

Contents

5.1 Introduction **119**

5.2 MOEA/D Framework **121**

5.3 Adaptive Discrete Differential Evolution **121**

5.4 Greedy Path-Relinking **122**

5.5 Proposed Hybridization Variants **125**

5.6 Experimental Design **126**

5.6.1 Parameter settings 126

5.6.2 Assessment Metrics 129

5.7 Experimental Results **129**

5.8 Concluding Remarks **134**

5.9 Summary **135**

5.1 Introduction

Over the last years, a large number of search algorithms were reported that do not purely follow the concepts of one single classical metaheuristics, but they attempt to obtain the best from a set of different metaheuristics (and even other kinds of optimization methods) that perform together, complement each other and augment their exploration capabilities to produce a profitable synergy from their combination. These approaches commonly referred to as hybrid metaheuristics [Raidl 2006]. A promising way to obtain hybrid metaheuristics concerns the combination of several search algorithms with strong specialization in intensification and/or diversification. The flexible architecture of evolutionary algorithms (EA) allows specialized models to be obtained with the aim of providing intensification and/or diversification. In fact, the design of hybrid metaheuristics with

EA is an innovative line of research with prospective future as a way for obtaining search algorithms that may achieve accurate and reliable solutions to hard real-world problems. This can be accomplished through the improvement of EA performance as well as improving the quality of the obtained solutions.

Hybrid Metaheuristics aim to incorporate and combine different metaheuristics with each other to enhance the search capabilities. They can improve both of intensification and diversification toward the preferred solutions and concentrate the search efforts to investigate the promising regions in the search space. As mentioned in chapter 4, a new hybrid evolutionary metaheuristics (HEMH) in which the search process is divided into two phases was developed. The first phase constructs an initial population of high quality solutions using DM-GRASP, whereas in the second phase, the greedy randomized path-relinking and/or reproduction operators are applied within the MOEA/D framework to improve the solutions previously obtained. This leads to intensify the search process in the regions surrounding the Pareto front and therefore, concentrating the search efforts on the promising regions to discover new high quality solutions. In this work, the efforts are concentrated on enhancing the search process proposed in the second phase regardless of the quality of the solutions obtained in the initialization phase. The basic motivation of this work is to obtain the most suitable combinations of search operators that improves the performance of the MOEA/D framework. Hence, extending these combinations to improve the proposed HEMH previously discussed. In this context, The competitive results achieved by the adaptive discrete differential evolution proposed in [Zhang *et al.* 2009] motivated us to combine it within the MOEA/D framework. Moreover, path-relinking could improve the search if it is applied on high quality solutions [Kafafy *et al.* 2011]. So, this chapter tends to study and analyze the effect of hybridization of the adaptive discrete differential evolution operator and/or path-relinking operators with the MOEA/D framework in handling MOCO problems. We study four proposals of hybridization, the first proposal is to combine adaptive discrete differential evolution operator within MOEA/D. The second one is to combine the path-relinking operator within MOEA/D. The third and the fourth proposals combine both of them in MOEA/D. The main goals are to determine the benefits and limitations of those techniques by studying possible combinations and their effects on the search capabilities. The rest this chapter is organized as follows: In section 5.2, the MOEA/D framework was reviewed. In section 5.3, an overview on adaptive discrete differential evolution is highlighted. The path-Relinking strategy is discussed in section 5.4. The proposed hybridization variants are presented in section 5.5. In addition, the experimental design and results are involved in sections 5.6 and 5.7 respectively. Section 5.8 presents some concluding remarks. Finally, the summary and some directions for further research are presented in

section 5.9.

5.2 MOEA/D Framework

Based on many traditional mathematical programming methods for approximating the PF [Miettinen 1999], the approximation of the PF can be decomposed into a number of single objective optimization subproblems. MOEA/D [Zhang & Li 2007] is considered as the most recently developed MOEA in which the decomposition idea is adopted instead of Pareto dominance concept. The MOEA/D framework and its features are explained in details in section 4.4. Here, the MOEA/D procedure is reviewed as it represents an essential part in this work. Alg.5.1 reviews the MOEA/D pseudocode.

Algorithm 5.1 MOEAD(N, T, t, δ)

Inputs:
N: *Population size or number of subproblems used*
T, t: *Neighborhood size & Maximum allowable replaced solutions*
$\delta \in [0, 1]$: *Probability of selecting parents from neighborhood*

1: **Begin:**
2: $\Lambda \leftarrow$ InitializeWeightVectors( ); ▷ *Generate a set of N evenly distributed weight vectors*
3: $B \leftarrow$ InitializeNeighborhood( ); ▷ *Construct the neighborhood structure*
4: $P \leftarrow$ InitializePopulation( ); ▷ *Generate initial solution for each subproblem*
5: $z \leftarrow$ InitializeRefPoint( );
6: **repeat**: ▷ *The main loop*
7: **for all** $(i \in \{1, 2, ..., N\})$ **do** ▷ *For each i_{th} subproblem*
8: $M_i \leftarrow \begin{cases} B(i) & \text{if}(rnd \in [0,1] < \delta) \\ P & \text{otherwise} \end{cases}$ ▷ *Determine the Mating/Updating rang M_i*
9: $x^a, x^b \leftarrow$ Selection(M_i, i): ▷ *Randomly Select two parents solutions x^a and x^b*
10: $u \leftarrow$ Reproduction$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$; ▷ *Apply crossover and mutation*
11: $y \leftarrow$ GreedyRepair(u, Λ^i);
12: EvaluateFitness(y); ▷ *Evaluate the offspring y*
13: $z \leftarrow$ UpdateRefPoint(y);
14: $P \leftarrow$ UpdateSolutions(y, t, M_i); ▷ *Update the population P*
15: **end for**
16: **until** Stopping criterion is satisfied
17: **return** P;

5.3 Adaptive Discrete Differential Evolution

As explained in section 3.6.2.1, the success of the DE algorithm relies on the differential mutation operator. It employs difference vectors built with pairs of candidate solutions in the search domain. The difference vector are scaled and added to a third point, producing the so-called mutant vector. In this chapter, we propose to use the differential mutation operator as an additional operator within the MOEA/D framework. We choose the adaptive discrete differential evolution

strategy proposed in [Zhang *et al.* 2009] to study its effect on the MOEA/D exploration capabilities in the discrete domains. This strategy is described in Alg. 5.2. Assume P is a population of N individuals. The main idea is to select at random three distinct individuals x_{r1}, x_{r2}, x_{r3} from P for each target individual $x_i \in P, \forall i \in \{1, ..., N\}$. The mutant individual v_i is produced by applying the differential uniform mutation on which called the parent base individual x_{r1} with the rate p_m. p_m is calculated based on the parent differential individuals (x_{r2}, x_{r3}) as follows:

$$p_m = F \cdot (\Delta (x_{r2}, x_{r3}) / n) \tag{5.1}$$

where Δ is the Hamming distance, n is the individual length and F denotes the scaling factor. Then, crossover is used to produce the new individual u_i as follows:

$$u_i^j = \begin{cases} v_i^j & \text{if } rnd(j) \leq CR, \text{ or } j = e, \forall j = 1, ..., n. \\ x_i^j & \text{otherwise, } \forall j = 1, ..., n. \end{cases} \tag{5.2}$$

where $rnd(j) \in [0, 1]$ is the jth random number generated by random number generator, e is a component of a random sequence S selected from $\{1, ..., n\}$ to insure that at least one component of u_i is contributed by v_i and $CR \in [0, 1]$ denotes the crossover factor. The mutation scaling factor F and the crossover factor CR are adapted periodically to avoid premature convergence as follows:

$$F = F_0 \cdot e^{(-a_1.(G/G_{max}))} \tag{5.3}$$

$$CR = CR_0 \cdot e^{(-a_2.(G/G_{max}))} \tag{5.4}$$

where G, G_{max} are the current and the maximum evolutionary generation respectively, F_0, CR_0 are the initial values of the scaling factor and the crossover operator respectively. a_1 and a_2 are plus constants. Finally, the new generated individual u_i is returned.

5.4 Greedy Path-Relinking

As described in section 3.6.2.3, Path-relinking generates new solutions by investigating the neighborhood space to explore trajectories that connect high quality solutions. it was suggested in [Glover *et al.* 2000] to integrate intensification and diversification strategies in the context of tabu search and scattered search. In this chapter, greedy path-relinking will be used as an intensification strategy in the MOEA/D framework, integrated with either reproduction by crossover and mutation or adaptive discrete differential evolution. It will be invoked in the higher generations to guarantee applying the relinking process on high quality solutions to improve the performance and enhance the efficiency. The greedy path-relinking operator that proposed here differs from the greedy

Algorithm 5.2 ABDifferentialEvolution$(x, y_1, y_2, y_3, F_0, CR_0, a_1, a_2)$

Inputs:
x: *Current individual*
y_1, y_2, y_3: *Three parent individuals*
$F_0, CR_0 \in [0, 1]$: *Scaling factor and crossover rate*
G, G_{max}: *Current and maximum generations*

1: **Begin:**
2: $F \leftarrow F_0 \cdot e^{(-a_1.(G/G_{max}))}$; ▷ *Adapting scaling factor F*
3: $CR \leftarrow CR_0 \cdot e^{(-a_2.(G/G_{max}))}$; ▷ *Adapting crossover rate CR*
4: $p_m \leftarrow F \cdot (\Delta(y_2, y_3)/n)$; ▷ *Compute Hamming distance according to Equ. 5.1*
5: $v \leftarrow$ Mutation(y_1, p_m); ▷ *Mutation*
6: **for all** $j \in \{1, ..., n\}$ **do**: ▷ *Crossover*
7: **if** $(rnd(j) \leq CR \vee j = e)$ **then**:
8: $u^j \leftarrow v^j$;
9: **else**
10: $u^j \leftarrow x^j$;
11: **end if**
12: **end for**
13: **return** u;

randomized one proposed in section 4.5.2.1 in tow issues. First, it depends on greedy strategy instead of greedy randomized one, which means, the best move is selected directly and thus there is no need to the restricted candidate list (RCL). Second, it adopts changing two bits per move in order to accelerate the time consumed in the relinking process.

The proposed greedy path-relinking procedure receives the inputs listed in Alg. 5.3. Firstly, the best of x^s and x^t is chosen to start with. Then, the best fitness z^* and the best solution x^* are initialized. The candidate lists CL and CL_{cmp} are constructed. Every unmatched j between x^s and x^t with $x^s_j = 0$ is inserted into CL in descending order according to the ratio in formula in Equ. 5.5. whereas, every unmatched j between x^s and x^t with $x^s_j = 1$ is inserted into CL_{cmp} in increasing order according to Equ. 5.5.

$$\sum_{i=1}^{m} \lambda_i \, c_{ij} / \sum_{i=1}^{m} w_{ij} \tag{5.5}$$

The procedure builds the path that connects x^s with x^t gradually by creating intermediate points through execution of the relinking loop. Initially, the intermediate solution x is set to x^s. Then, the number of unmatched items between x and x^t ($\Delta(x, x^t)$) is calculated. The next move is carried out by selecting two of unmatched ℓ^1, ℓ^2 to be matched. If both CL and CL_{cmp} are not empty, then the first elements of CL and CL_{cmp} are extracted to be ℓ^1 and ℓ^2 respectively. Else if one of them is empty, then, the first and second element of the non empty

one will be extracted to be ℓ^1 and ℓ^2 respectively. The new intermediate x is obtained by flipping the two items (x_{ℓ^1}, x_{ℓ^2}) corresponding to the selected indexes ℓ^1 and ℓ^2 in the current intermediate x. If x is infeasible, the Greedy-Repair is invoked to get the feasible solution y. Then, z^* and x^* are updated by y. This process is repeated until there is only one unmatched item between the current intermediate x and the guiding x^t. Finally x^* is returned.

Algorithm 5.3 GreedyPathRelinking(x^s, x^t, Λ)

Inputs:
x^s, x^t: *Starting and Guiding solutions*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *weight vector of the current subproblem*

1: **Begin:**
2: $x^* \leftarrow$GetTheBestOf(x^s, x^t, Λ); ▷ *By computing $F^{ws}(x^s, \Lambda)$, $F^{ws}(x^t, \Lambda)$ based on Equ. 2.17*
3: **if** $x^* \neq x^s$ **then:** Swap(x^s, x^t); ▷ *To begin Relinking from the best*
4: $z^* \leftarrow F^{ws}(x^*, \Lambda)$; ▷ *Compute $F^{ws}(x^*, \Lambda)$ according to Equ. 2.17*
5: $CL\&CL_{cmp} \leftarrow \varnothing$;
6: **while** $\exists j | j \notin CL \wedge x_j^s \neq x_j^t \wedge x_j^s = 0 \wedge Max_{j=1}^{n} \left(\frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}} \right)$ **do**:
7: $CL \leftarrow$Append(j);
8: **end while**
9: **while** $\exists j | j \notin CL_{cmp} \wedge x_j^s \neq x_j^t \wedge x_j^s = 1 \wedge Min_{j=1}^{n} \left(\frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}} \right)$ **do**:
10: $CL_{cmp} \leftarrow$Append(j);
11: **end while**
12: $x \leftarrow x^s$;
13: $\Delta(x, x^t) \leftarrow |\{j \in \{1, ..., n\} : x_j \neq x_j^t\}|$; ▷ *Compute Hamming distance $\Delta(x, x^t)$*
14: **while** $\Delta(x, x^t) \geq 2$ **do**: ▷ *Relinking loop*
15: **if** $|CL| \neq 0 \wedge |CL_{cmp}| \neq 0$ **then**: ▷ *$CL\&CL_{cmp}$ are not empty*
16: $\ell^1 \leftarrow$ ExtractTheFirstElement(CL) ▷ *Extract the 1st element*
17: $\ell^2 \leftarrow$ ExtractTheFirstElement(CL_{cmp})
18: **else if** $|CL| > 1$ **then**:
19: $\ell^1 \leftarrow$ ExtractTheFirstElement(CL)
20: $\ell^2 \leftarrow$ ExtractTheFirstElement(CL)
21: **else**:
22: $\ell^1 \leftarrow$ ExtractTheFirstElement(CL_{cmp})
23: $\ell^2 \leftarrow$ ExtractTheFirstElement(CL_{cmp})
24: **end if**
25: $x \leftarrow$ FlipBits(x, ℓ^1, ℓ^2);
26: $y \leftarrow$Repair(x, Λ)
27: **if** $(F^{ws}(y, \Lambda) > z^*)$ **then**: ▷ *Compute $F^{ws}(y, \Lambda)$ according to Equ. 2.17*
28: $x^* \leftarrow y$; $z^* \leftarrow F^{ws}(y, \Lambda)$;
29: **end if**
30: $\Delta(x, x^t) \leftarrow |\{j \in \{1, ..., n\} : x_j \neq x_j^t\}|$; ▷ *Update the Hamming distance $\Delta(x, x^t)$*
31: **end while**
32: **return** x^*;
33: **End**

5.5 Proposed Hybridization Variants

In this work, we study the effect of using both of adaptive differential evolution operator proposed in [Zhang *et al.* 2009] and/or proposed path-relinking as a reproduction operator instead of standard reproduction (crossover and mutation) in MOEA/D framework. So we have four algorithm variants, the first variant is called MOEAD$_{de}$ in which the adaptive discrete differential evolution completely replaces crossover and mutation operators. The second variant is called MOEAD$_{pr}$ in which the proposed path-relinking operator is applied with the crossover and standard mutation after a certain number of evaluations to guarantee the existence of high quality solutions. Pseudo codes in Alg. 5.4 and Alg. 5.5 describe MOEAD$_{pr}$ and MOEAD$_{de}$ respectively. In the third and the fourth variants, both of differential evolution and path-relinking replaces crossover and mutation, they are called MOEAD$_{dp1}$ and MOEAD$_{dp2}$ respectively. Fig.5.1 depicts these different variants.

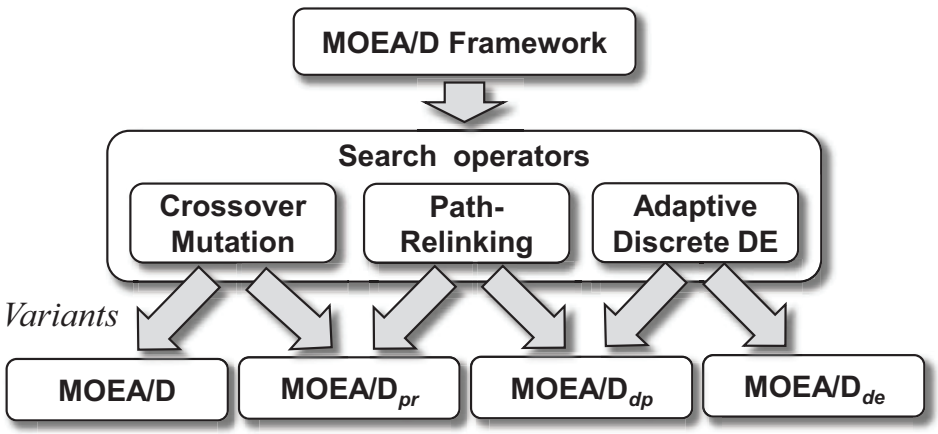


Figure 5.1: The proposed hybridization variants

In Both MOEAD$_{de}$ and MOEAD$_{pr}$, the set of uniform weight vectors Λ is calculated, followed by construction of the neighborhood structure. The initial population is also generated (lines 2-5). Then the main loop is executed until achieving the maximum evaluations (line 6). To generate a new offspring for each subproblem i, the mating/updating range (M) is determined to be either the neighborhood of the ith subproblem (*Local*), or the whole population (*Global*) according to a certain probability (σ). This can give a better chance for selecting distinct parents, which encourages the path-relinking to be invoked in MOEAD$_{pr}$, or allows the differential evolution to operate on distinct individuals in MOEAD$_{de}$. Then, parent selection is performed. In case of MOEAD$_{pr}$ (Alg. 5.4), two parents

x^j and x^k are randomly selected from M. Then, the path-relinking operator is used only if the Hamming distance between the two selected parent is greater than a certain value ε and the number of evaluations $Eval$ exceeds a certain ratio (γ) of the maximum evaluations allowed to guarantee applying path-relinking on high quality solutions. Else, the standard reproduction operator is applied to generate the new offspring. In case of MOEAD$_{de}$ variant (Alg. 5.5), three distinct parent individuals are randomly selected to apply adaptive discrete differential evolution on them. The new generated offspring is evaluated, and used to update the reference point z and also updating the population according to the parameter t, which is used to limit the number of replaced solutions. Finally, the efficient solution set in the final population is returned as an output. In both MOEAD$_{dp1}$ and MOEAD$_{dp2}$, some modifications are applied on MOEAD$_{de}$ (Alg. 5.5) to involve path-relinking after certain number of evaluations that carried out to assure the existence of high quality solutions. These modifications can be briefed as follows: when the number of evaluations $Eval$ exceeds a certain value ($\gamma \times MaxEvals$) previously determined to involve path-relinking, we have three selected parents x^a, x^b and x^c in the selection step (Alg. 5.5:line 9). If we randomly choose two of them which have Hamming distance greater than a certain value (ε) to apply path-relinking on instead of differential evolution, we will get the MOEAD$_{dp1}$ variant. On the other hand, assuming the Hamming distance conditions[1] are satisfied, if we apply path-relinking on the three selected parents(x^a, x^b and x^c) in the following manner: randomly choosing two individuals (x^a,x^c) to apply path-relinking producing a new individual y, then applying path-relinking on y and x^b, we will get MOEAD$_{dp2}$ variant. The pseudo code of both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ can be obtained by replacing lines 10-11 in Alg. 5.5 by the pieces of code shown in Alg. 5.6 and 5.7 respectively.

5.6 Experimental Design

In this work, all experiments have been performed on DELL PC with Intel Core i5-2400 CPU, 3.10 GHz and 4.0 GB of RAM. The comparative study for different algorithm variant was carried out on the set of test instances listed below in table 5.1, which are commonly used in literature. SPEA2 [Zitzler *et al.* 2001] algorithm is also used in this study.

5.6.1 Parameter settings

Here, the different parameters used for each algorithm are discussed. The population size (N) used in SPEA2 is shown in table 5.1. For MOEA/D and its variants MOEAD$_{de}$, MOEAD$_{pr}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$, the parame-

[1]it means that x^j and x^k can be chosen for path-relinking only if: $\Delta(x^j, x^k) \geq \varepsilon$

Algorithm 5.4 MOEAD$_{pr}(N, T, t, \delta, \varepsilon, \gamma)$

Inputs:
N, T, t: *Population size, Neighborhood size & No. of replaced solutions*
$\delta \in [0, 1]$: *Prob. of selecting parents from neighborhood*
ε, γ: *Min. Hamming distance, Min. evaluations allowed for path-relink*

Begin:
$\Lambda \leftarrow$InitializeWeightVectors( );
$B \leftarrow$InitializeNeighborhood( );
$P \leftarrow$InitializePopulation( );
$z \leftarrow$InitializeRefPoint( ); $Eval \leftarrow 0$;
while $(Eval < MaxEvals)$ **do** ▷ *main loop*
  for all $(i \in \{1, 2, ..., N\})$ **do**
    $M \leftarrow \begin{cases} B(i) & \text{if}(rnd \in [0,1] < \delta) \\ P & \text{otherwise} \end{cases}$
    $x^j, x^k \leftarrow$ Selection(M, i) ▷ *select 2 elements*
    if $(\Delta(x^j, x^k) \geq \varepsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
      $y \leftarrow$PathRelinking(x^j, x^k, Λ^i);
    else
      $u \leftarrow$Reproduction(x^j, x^k);
      $y \leftarrow$GreedyRepair(u, Λ^i)
    end if
    EvaluateFitness(y);
    $z \leftarrow$UpdateRefPoint(y);
    $P \leftarrow$UpdateSolutions(y, t, M);
    $Eval \leftarrow$Update( );
  end for
end while
return P;

Table 5.1: Set of knapsack test instances

Instance	**Knaps.**(m)	**Items**(n)	SPEA2(N)	$N(H)$	$MaxEvals$
KSP252	2	250	150	150(149)	75000
KSP502	2	500	200	200(199)	100000
KSP752	2	750	250	250(249)	125000
KSP253	3	250	200	300(23)	100000
KSP503	3	500	250	300(23)	125000
KSP753	3	750	300	300(23)	150000
KSP254	4	250	250	364(11)	125000
KSP504	4	500	300	364(11)	150000
KSP754	4	750	350	364(11)	175000

ter H which controls both the number of weight vectors and the population size (N) is determined for each instance in table 5.1 according to the complexity. The initial population used is randomly generated such that each member $x = (x_1, ..., x_n)^T \in \{0, 1\}^T$, where $x_i = 1$ with probability equal to 0.5. The maximum number of evaluations ($MaxEvals$) is used as a stopping criterion for each algorithm. For each algorithm, all efficient solutions rest in the final iteration is used as the final approximation set. In these experiments, single-point crossover

Algorithm 5.5 MOEAD$_{de}(N, T, t, \delta, F_0, CR_0, a_1, a_2)$

Inputs:
N, T, t: *Population size, Neighborhood size & No. of replaced solutions*
$\delta \in [0, 1]$: *Prob. of selecting parents from neighborhood*
$F_0, CR_0 \in [0, 1]$: *Scaling factor and Crossover rate*

1: **Begin:**
2: $\Lambda \leftarrow$InitializeWeightVectors( );
3: $B \leftarrow$InitializeNeighborhood( );
4: $P \leftarrow$InitializePopulation( );
5: $z \leftarrow$InitializeRefPoint( ); $Eval \leftarrow 0$;
6: **while** $(Eval < MaxEvals)$ **do** ▷ *main loop*
7: **for all** $(i \in \{1, 2, ..., N\})$ **do**
8: $M \leftarrow \begin{cases} B(i) & \text{if}(rnd \in [0,1] < \delta) \\ P & \text{otherwise} \end{cases}$
9: $x^a, x^b, x^c \leftarrow$ Selection(M, i): ▷ *Where:* $x^i \neq x^a \neq x^b \neq x^c$
10: $u \leftarrow$ABDifferentialEvolution$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
11: $y \leftarrow$GreedyRepair(u, Λ^i);
12: EvaluateFitness(y);
13: $z \leftarrow$UpdateRefPoint(y);
14: $P \leftarrow$UpdateSolutions(y, t, M);
15: $Eval \leftarrow$Update( );
16: **end for**
17: **end while**
18: **return** P;

Algorithm 5.6 MOEAD$_{dp1}(N, T, t, \delta, \varepsilon, \gamma, F_0, CR_0, a_1, a_2)$

Replace lines (10-11) in Alg.5.5 by the following:

1: $x^j, x^k \leftarrow$ RandomSelection(x^a, x^b, x^c);
2: **if** $(\Delta(x^j, x^k) \geq \varepsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
3: $y \leftarrow$PathRelinking(x^j, x^k, Λ^i);
4: **else**
5: $u \leftarrow$ABDifferentialEvolution$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
6: $y \leftarrow$GreedyRepair(u, Λ^i);
7: **end if**

Algorithm 5.7 MOEAD$_{dp2}(N, T, t, \delta, \varepsilon, \gamma, F_0, CR_0, a_1, a_2)$

Replace lines(10-11) in Alg.5.5 by the following:

1: $x^a, x^c \leftarrow$ RandomSelection(x^a, x^b, x^c);
2: **if** $(\Delta(x^a, x^c) \geq \varepsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
3: $y \leftarrow$PathRelinking(x^j, x^k, Λ^i);
4: **if** $(\Delta(x^b, y) \geq \varepsilon)$ **then**
5: $y \leftarrow$PathRelinking(y, x^b, Λ^i);
6: **end if**
7: **else**
8: $u \leftarrow$ABDifferentialEvolution$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
9: $y \leftarrow$GreedyRepair(u, Λ^i);
10: **end if**

and standard mutation were considered. Mutation was performed for each item independently with probability $(1/n)$. SPEA2 uses single point crossover with probability=1 and tournament selection with tournament size=2. The other control parameters are listed in table 5.2. Finally, the statistical analysis is applied on 30 independent runs for each test instance.

Table 5.2: Set of common parameter used

Parameters	**MOEA/D variants**			
	MOEA/D	**MOEA/D$_{de}$**	**MOEA/D$_{pr}$**	**MOEA/D$_{dp}$**
Neighborhood size: T	10	10	10	10
Max. replaced solutions: t	2	2	2	2
Parents selection: δ	-	0.9	0.9	0.9
Ratio to apply Path-relink: γ	-	-	0.7	0.7
Min. Hamming Distance: ε	-	-	10	10
Initial crossover rate: CR_0	-	0.4	-	0.4
Initial scaling factor: F_0	-	0.4	-	0.4
Plus constants: a_1, a_2	-	2, 2	-	2, 2

5.6.2 Assessment Metrics

In order to assess the quality of the solutions obtained by each algorithm, some of the quality assessment indicators are used in these experiments including binary and unary indicators. These indicators include the set coverage I_C indicator which is used to compare each pair of MOEAs as discussed in section 2.6.2.1. They also include indicators to assess each MOEA independently such as referenced hypervolume (I_{Rhyp}), generational distance (I_{GD}), inverted generational distance (I_{IGD}) and R_3 ($I_M R_3$) indicators which are presented in sections 2.6.2.2, 2.6.2.3, 2.6.2.3 and 2.6.2.5 respectively.

In these experiments, the reference set P^* is alternatively formed for each problem instance by gathering all nondominated solutions found by all of the compared algorithms in all runs. Also, all approximation sets are normalized in the range [1,2].

5.7 Experimental Results

Here, the different simulation results are shown in details. Firstly, Fig.5.2 depicts the results of I_C metric. It contains a chart (with scale 0 at the bottom and 1 at the top) for each ordered pair of the compared algorithms. Each chart consists of nine box plots representing the distribution of I_C values. Each box plot (from left to right) represents an instance in table 5.1 (from top to down), respectively. A chart located in the raw of algorithm A and the column of algorithm B presents the values of coverage of the approximations generated by algorithm B by approximations generated by algorithm A. It is clear from the

results in Fig.5.2 that all four hybrid variants outperform the original MOEA/D in most test instances. It is also clear that MOEAD$_{pr}$ has the best performance for all bi-objective test instances.

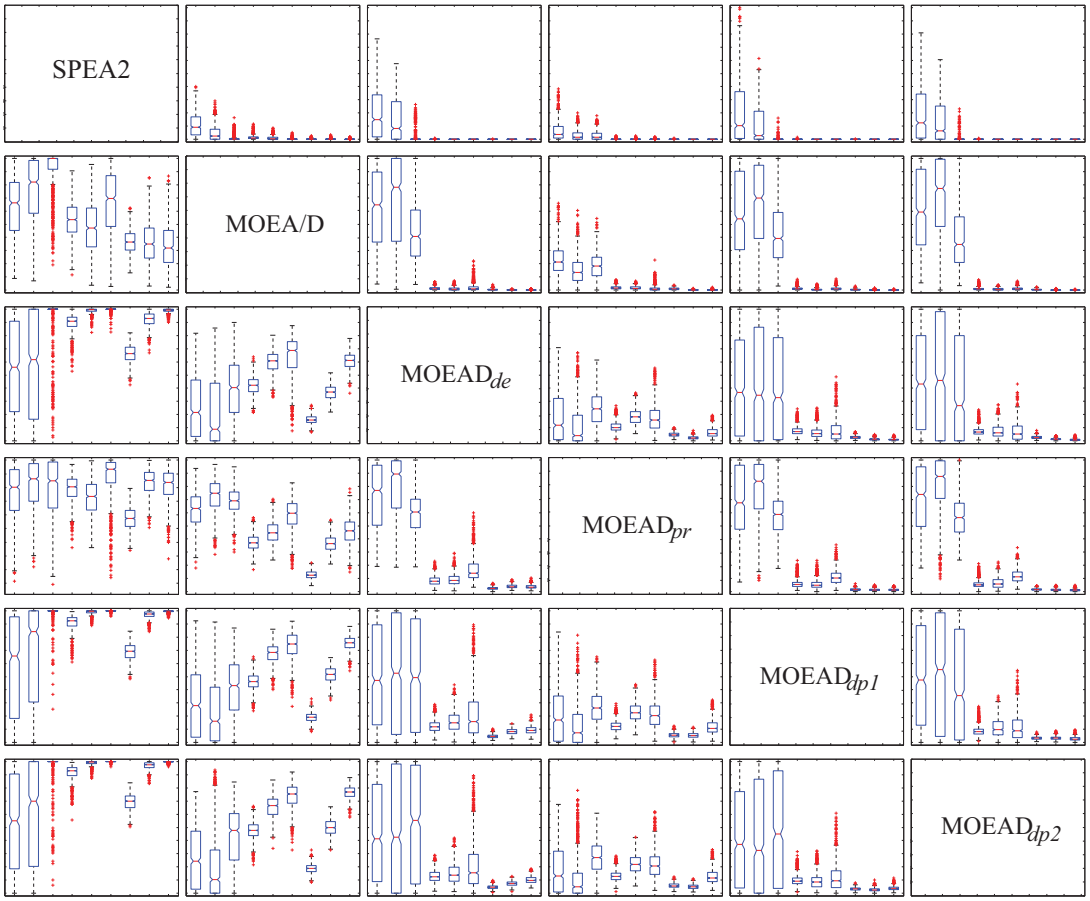


Figure 5.2: Results of I_C indicator

The results of I_{RH} listed in table 5.3 contain the average of I_{RH} values achieved over 30 independent runs for each test instance for each algorithm. Fig.5.3 visualizes the average values. It is clear that all hybrid variants outperform the original MOEA/D for all 3 and 4 objective test instances especially Path-relinking based variants (MOEAD$_{pr}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$), since they have the minimum average values. We find also, MOEAD$_{pr}$ has the best performance with respect to most instances, while the differential evolution based variants (MOEAD$_{de}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$) have poor performance with respect to bi-objective instance compared with the original MOEA/D.

Table 5.3: Average results of the referenced Hypervolume (I_{Rhyp})

Instance	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	4.45E-01	4.07E-02	1.48E-01	**2.96E-02**	1.41E-01	1.43E-01
KSP502	6.51E-01	4.76E-02	1.72E-01	**3.06E-02**	1.68E-01	1.70E-01
KSP752	7.32E-01	3.97E-02	1.19E-01	**2.86E-02**	1.13E-01	1.11E-01
KSP253	1.58E+00	2.09E-01	1.80E-01	**1.40E-01**	1.64E-01	1.67E-01
KSP503	2.11E+00	2.50E-01	2.12E-01	**1.40E-01**	1.86E-01	1.94E-01
KSP753	2.41E+00	2.85E-01	2.80E-01	**1.21E-01**	2.62E-01	2.53E-01
KSP254	4.00E+00	8.34E-01	6.49E-01	6.11E-01	**6.10E-01**	**6.10E-01**
KSP504	5.05E+00	1.06E+00	6.55E-01	**4.92E-01**	5.66E-01	5.66E-01
KSP754	5.76E+00	1.29E+00	7.50E-01	**5.07E-01**	6.09E-01	5.79E-01

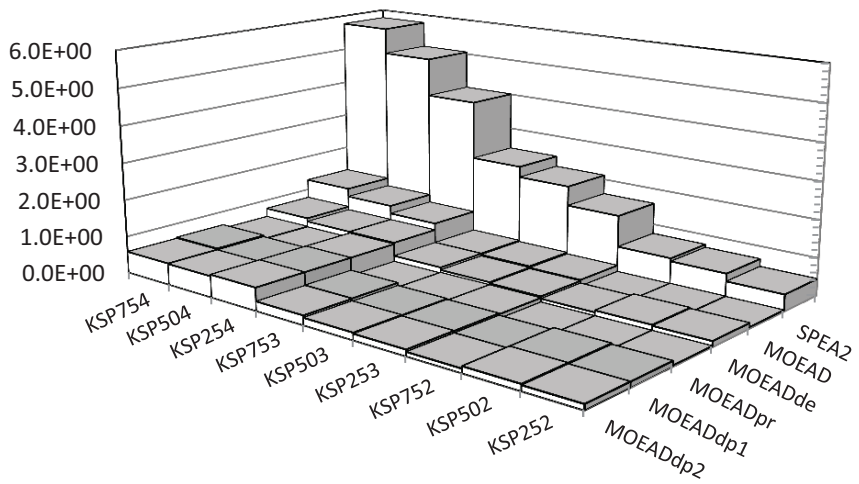


Figure 5.3: Average results of the referenced hypervolume indicator I_{Rhyp}

In table 5.4, the average values of the generational distance I_{GD} are listed. Additionally, Fig. 5.4 visualizes the average values. According to I_{GD} measure, it is clear that the proposed hybrid variants outperform the original MOEA/D for most instances since they have the minimum average values. Also, MOEAD$_{pr}$ outperforms with respect to bi-objective test instances, while both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ have the superiority in the rest. In contrast, we find the differential evolution based variants especially MOEAD$_{de}$ achieve bad results than the original MOEA/D in bi-objective instances. This means that all hybrid variants have the capabilities of discovering solutions as near as possible to the Pareto frontier.

Table 5.4: Average results of the generational distance indicator(I_{GD})

Instance	**Algorithm**					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	2.79E-03	1.39E-03	2.25E-03	**1.12E-03**	2.22E-03	2.33E-03
KSP502	3.98E-03	1.53E-03	2.34E-03	**1.10E-03**	2.18E-03	2.36E-03
KSP752	4.62E-03	1.28E-03	1.38E-03	**1.23E-03**	1.27E-03	1.24E-03
KSP253	3.71E-03	1.88E-03	7.39E-04	9.77E-04	6.55E-04	**6.30E-04**
KSP503	4.06E-03	2.13E-03	7.12E-04	1.24E-03	**5.93E-04**	6.08E-04
KSP753	4.18E-03	2.10E-03	8.06E-04	1.05E-03	6.55E-04	**6.37E-04**
KSP254	5.17E-03	2.37E-03	9.84E-04	1.33E-03	8.76E-04	**8.61E-04**
KSP504	6.34E-03	3.18E-03	9.58E-04	1.02E-03	**5.98E-04**	6.17E-04
KSP754	6.75E-03	3.69E-03	9.73E-04	1.39E-03	6.08E-04	**5.51E-04**

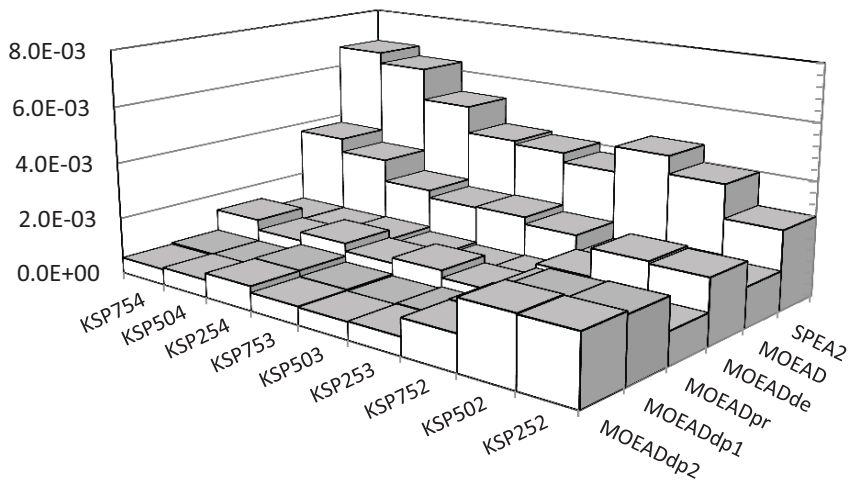


Figure 5.4: Average results of the generational distance indicator I_{GD}

The experimental results of the inverted generational distance (I_{IGD}) are listed in table 5.5. Fig. 5.5 depicts these results. These results are identical with I_{GD}-metric, where MOEAD$_{pr}$ performs better in bi-objective and both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ perform better with respect to many-objective.

Table 5.5: average results of the inverted generational distance(I_{IGD})

Instance	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	1.09E-02	1.00E-03	2.31E-03	**8.37E-04**	2.40E-03	2.25E-03
KSP502	1.16E-02	9.72E-04	1.89E-03	**7.05E-04**	1.88E-03	1.89E-03
KSP752	1.27E-02	7.96E-04	1.29E-03	**7.04E-04**	1.20E-03	1.16E-03
KSP253	2.24E-03	6.11E-04	4.48E-04	4.99E-04	4.38E-04	**4.35E-04**
KSP503	2.83E-03	6.12E-04	4.11E-04	4.82E-04	**3.99E-04**	4.01E-04
KSP753	3.15E-03	5.93E-04	3.89E-04	4.27E-04	3.79E-04	**3.70E-04**
KSP254	1.45E-03	6.80E-04	5.54E-04	6.07E-04	5.53E-04	**5.49E-04**
KSP504	1.69E-03	6.51E-04	4.56E-04	4.82E-04	4.41E-04	**4.37E-04**
KSP754	1.91E-03	6.88E-04	4.40E-04	4.81E-04	4.14E-04	**4.07E-04**

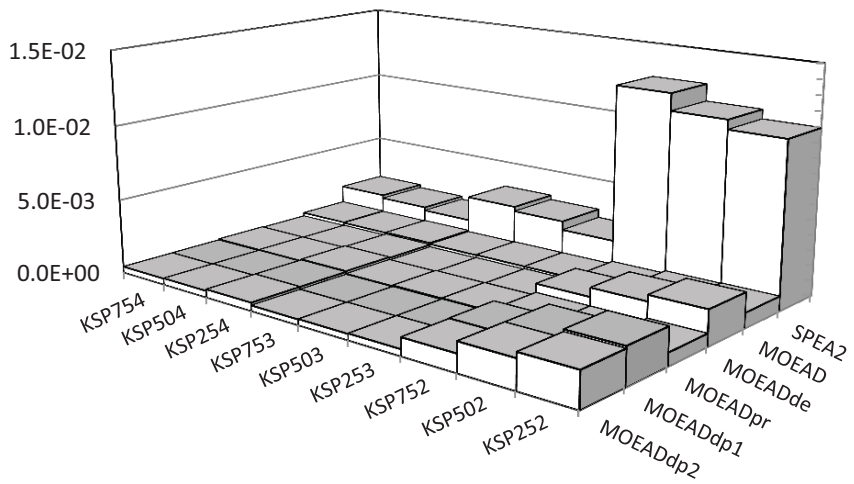


Figure 5.5: Average results of the inverted generational distance I_{IGD}

The R-indicator I_{R_3} illustrated in table 5.6 and depicted by Fig. 5.6 confirms the results of the previous metrics. It indicates that MOEAD$_{pr}$ variant outperforms with respect to bi-objective test instances. Where, MOEAD$_{dp1}$ and MOEAD$_{dp2}$ have the best performance with respect to the other instances.

Table 5.6: Average results of R_3 indicator (I_{R_3})

Instance	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	4.28E-02	5.27E-03	1.08E-02	**3.85E-03**	1.09E-02	1.12E-02
KSP502	7.98E-02	6.66E-03	1.40E-02	**4.64E-03**	1.33E-02	1.40E-02
KSP752	9.61E-02	6.46E-03	8.11E-03	**5.64E-03**	7.23E-03	7.16E-03
KSP253	6.07E-02	1.09E-02	6.29E-03	6.65E-03	5.82E-03	**5.79E-03**
KSP503	9.74E-02	1.32E-02	7.08E-03	7.26E-03	**5.91E-03**	6.27E-03
KSP753	1.21E-01	1.45E-02	8.34E-03	**6.52E-03**	7.25E-03	7.06E-03
KSP254	7.24E-02	1.55E-02	1.03E-02	1.11E-02	9.66E-03	**9.63E-03**
KSP504	1.06E-01	2.05E-02	1.08E-02	9.09E-03	**8.93E-03**	9.00E-03
KSP754	1.38E-01	2.54E-02	1.21E-02	1.02E-02	9.17E-03	**8.81E-03**

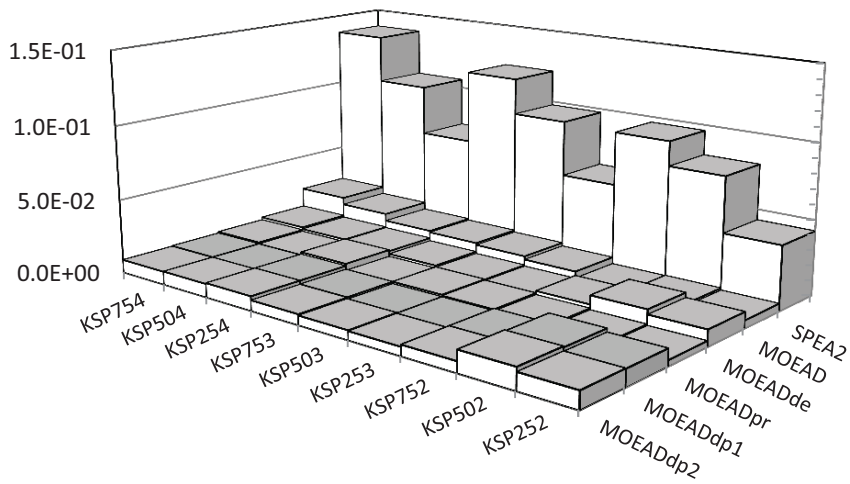


Figure 5.6: Average results of the R_3 indicator I_{R_3}

5.8 Concluding Remarks

Regarding the above results on MOKP test instances, we can deduce the following remarks:

- MOEAD$_{pr}$ variant achieves better results than the original MOEA/D for all test instances in all used metrics.
- MOEAD$_{de}$ variant outperforms the original MOEA/D for test instance with 3 or 4 objective. Conversely, it deteriorates the MOEA/D performance concerning bi-objective instances.
- The performance of both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ is highly affected by MOEAD$_{de}$ performance. Since they depend on differential evolution strategy more than path-relinking.

In general, path-relinking operator has the ability to improve the performance of the MOEA/D for all instances especially with bi-objective instances. Whereas differential evolution improves the MOEA/D performance in three and four objectives instances. Consequently, the performance of their hybrid variants MOEAD_{dp1} and MOEAD_{dp2} is enhanced.

5.9 Summary

In this chapter, four different hybridization variants within MOEA/D framework are presented. The first one is called MOEAD_{de} which involves the adaptive discrete differential evolution as a recombination operator within MOEA/D Framework. The second is called MOEAD_{pr}, which uses the path-relinking operator with the standard reproduction operators. In the third and fourth variants both of differential evolution and path-relinking are used. The four proposals are compared with the original MOEA/D and SPEA2 using a set of MOKP instances commonly used in the literature. A set of quality assessment indicators is also used to assess the performance. The experimental results indicate the superiority of all proposed hybrid variants over the original MOEA/D and SPEA2 for most test instances. In bi-objective test instances, we found that MOEAD_{pr} has the superiority, while MOEAD_{de} has poor performance. On the other hand, in case of instances with three or four objectives, the performance of the differential evolution is improved. Consequently, all proposed variants achieve better performance. They have an average performance highly competitive with respect to the original MOEA/D and SPEA2 based on the assessment indicators used in this study. The general conclusion we have is: for bi-objective MOKP test instances, path-relinking operator has the first rank followed by the standard crossover and mutation then differential evolution. Where in MOKP test instances with three or four objectives, differential evolution and path-relinking perform better than standard crossover and mutation. In the next chapter, these results will be exploited to improve the HEMH [Kafafy *et al.* 2011] presented in chapter 4 . We will also study how to improve the performance of differential evolution on discrete search domains.

Chapter 6

An Improved Hybrid Evolutionary Metaheuristics

Contents

6.1 Introduction **137**

6.2 HEMH, an overview **138**

6.3 Adaptive Binary Differential Evolution **139**

6.4 Path Relinking **140**

6.5 Pareto Adaptive ε-Dominance Archiving **142**

6.6 The Proposed HEMH2 . . . . . **143**

6.7 Experimental Design **146**

6.7.1 Parameter settings 146

6.7.2 Assessment Metrics 148

6.8 Experimental Results **149**

6.9 Summary **153**

6.1 Introduction

Hybrid evolutionary metaheuristics incorporate different cooperative metaheuristics to enhance the search capabilities. As discussed before, the HEMH [Kafafy *et al.* 2011] presented in chapter 4 involves two phases search process. First, a set of initial high quality solutions is constructed by DMGRASP. Then, the greedy randomized path relinking and reproduction (crossover and mutation) operators were applied to improve these initial solutions. The whole process is combined into the MOEA/D framework. There is no doubt that DMGRASP produces high quality initial solutions, but it consumes more time and evaluations which means a little chance to improve these initial solutions by the second phase. According to the comparative study [Kafafy *et al.* 2012b] provided in chapter 5, the second phase can be improved by using different combinations of

search operators rather than using only path relinking and classical crossover and mutation. Due to these reasons, this chapter tends to improve the performance of HEMH through developing a new version called HEMH2 with another two variants called HEMH$_{de}$ and HEMH$_{pr}$. The main motivations of this work are to overcome the limitations from which the performance of HEMH suffers. Unlike HEMH, HEMH2 uses simple inverse greedy algorithm to construct its initial population. Then, the search efforts are directed to improve these solutions by exploring the search space using the adaptive binary differential evolution rather than classical crossover and mutation. After a certain number of evaluations, the greedy path relinking operator is applied on the high quality solutions obtained to investigate the non-visited regions in the search space. During evaluations, a dynamic-sized neighborhood structure is adopted to shrink/extend the mating/updating range. Furthermore, the Pareto adaptive epsilon concept is used to control the archiving process with preserving the extreme solutions. Moreover, all improvement proposals and their effects on the search process will be discussed in details. The reminder of this chapter is organized as follows: HEMH framework is reviewed in section 6.2. Section 6.3 explains the adaptive binary differential evolution. Greedy path relinking strategy is discussed in section 6.4. In section 6.5, the archiving process using Pareto Adaptive ε-dominance is explained. The proposed HEMH2 and its variants are presented in section 6.6. Additionally, the experimental design and results are involved in sections 6.7 and 6.8 respectively. Finally, the summary and future works are involved in section 6.9.

6.2 HEMH, an overview

In HEMH, a combination of different cooperative metaheuristics is provided to handle 0/1 MOKP. The MOEA/D framework [Zhang & Li 2007] is adopted to carry out the combination. The *weighted sum* defined in Equ. (2.17) is considered to decompose the MOKP formulated in Equ. (2.7) into a set of N single objective subproblems, based on a set of N evenly distributed weight vectors $\{\Lambda^1, ..., \Lambda^N\}$. HEMH attempts to simultaneously optimize these subproblems. The HEMH framework consists of the following:

- A population P of N individuals, $P = \{x^1, ..., x^N\}$, where x^i represents the current solution of the ith subproblem.

- A set of N evenly distributed weight vectors $\{\Lambda^1, ..., \Lambda^N\}$, correspond to the N subproblems. Each $\Lambda = [\lambda_1, ..., \lambda_m]$ has m components correspond to m-objectives, such that: $\sum_{i=1}^{m} \lambda_i = 1$, $\forall \lambda_i \in \{0/H, ..., H/H\}$, and $H \in \mathbb{Z}^+$.

- A neighborhood B_i for each subproblem $i \in \{1, ..., N\}$, which includes all subproblems with the T closest weight vectors $\{\Lambda^{i1}, ..., \Lambda^{iT}\}$ to Λ^i.

- An *archive* to collect all efficient solutions explored over the search process.

The HEMH consists of two basic phases, *initialization* and *main loop*. In the initialization phase, an initial population of high quality solutions is constructed by applying DMGRASP on each subproblem. Then, the search efforts are concentrated on the promising regions to explore new efficient solutions. In the main loop phase, for each subproblem i, the mating/updating range M_i is chosen to be either the neighborhood B_i or the whole population P. Then, two individuals are randomly selected from M_i for reproduction. Single point crossover and mutation or greedy randomize path relinking is applied on the selected individuals to generate a new offspring, which is used to update M_i and the archive. This process is repeated until a certain number of evaluations. We refer to [Kafafy *et al.* 2011] for more details. The limitations that affect the HEMH performance are briefed as:

- Despite using DMGRASP achieves high quality initial solutions, it consumes more time and evaluations especially with large populations. Thus, the second phase will not have enough chance to improve the search process.

- Collecting all efficient solutions causes waste in time and storage space especially in many objective cases. So, archiving process should be controlled.

- For each subproblem i, the mating/updating range M_i is either the fixed (static) size neighborhood B_i or the whole population P. This may cause less execution of path relinking, or consume more time.

- Reproduction is made only by single point crossover and mutation or greedy randomized path relinking.

- Path relinking adopts single bit flipping per move and also uses local search to improve the generated solution. This causes more time consumption.

From the above, this chapter presents an improved version of HEMH called HEMH2 with two variants HEMH_{de} and HEMH_{pr}, which have the ability to overcome those limitations and can achieve an enhanced performance.

6.3 Adaptive Binary Differential Evolution

As mentioned in section 3.6.2.1, differential evolution (DE) is a simple and efficient evolutionary algorithm to solve optimization problems mainly in continuous search domains [Chakraborty 2010, Price *et al.* 2005]. DE's success relies on the

differential mutation, that employs difference vectors built with pairs of candidate solutions in the search domain. Each difference vector is scaled and added to another candidate solution, producing the so-called mutant vector. Then, DE recombines the mutant vector with the parent solution to generate a new offspring. The offspring replaces the parent only if it has an equal or better fitness. DE has some control parameters as the mutation factor F, that used to scale the difference vectors, and the crossover rate CR. In this chapter, an adaptive binary DE strategy is introduced to improve the exploration capabilities of HEMH instead of classical crossover and mutation. The procedure of this strategy is described in Alg. 6.1. Given a population P of N individuals, where each individual represented by a n-component 0/1 vector. The main idea is to select at random three distinct individuals x^a, x^b and x^c from P for each target individual $x^i \in P$, $\forall i \in \{1, ..., N\}$. The mutant individual v^i is produced by applying binary differential mutation on the selected individuals according to formula in Equ. 6.1. First, the *difference vector* is calculated by applying logical XOR on the two parent differential individuals x^b and x^c. Then, v^i is determined by applying logical OR on the parent based individual x^a and the difference vector previously obtained. Finally, the new generated offspring u^i is produced by applying crossover according to formula in Equ. 6.2.

$$v^i = x^a + (x^b \oplus x^c) \tag{6.1}$$

$$u^i_j = \begin{cases} v^i_j & \text{if } rnd(j) \leq CR, \text{ or } j \in e, \forall j = 1, ..., n. \\ x^i_j & \text{otherwise}, \forall j = 1, ..., n. \end{cases} \tag{6.2}$$

where $rnd(j) \in [0, 1]$ is a random number generated for the jth component, n is the individual length, e is a random sequence selected from the range $\{1, ..., n\}$ to insure that at least one component of u^i is contributed by v^i and $CR \in [0, 1]$ denotes crossover rate. Here, CR is adapted periodically to avoid the premature convergence based on Equ. 6.3 proposed in [Zhang *et al.* 2009].

$$CR = CR_0 \cdot e^{(-a.(G/G_{max}))} \tag{6.3}$$

where G and G_{max} are the current and the maximum evolutionary generations, CR_0 is the initial crossover rate. a is a constant. Finally, the whole process is depicted in Fig. 6.1.

6.4 Path Relinking

As mentioned in sections 4.3 and 5.4, path relinking operator generates new solutions through exploring the trajectories that connect high quality solutions. Starting from the starting solution x^s, path relinking builds a path in the neigh-

Algorithm 6.1 ADAPTIVEBINARYDIFFEVOL$(x, x^a, x^b, x^c, CR_0, a)$

Inputs:
x: *Current solution*
x^a, x^b, x^c: *Parents individuals*
$CR_0 \in [0,1]$: *Crossover rate*
a: *Plus constant*

1: **Begin:**
2: $CR \leftarrow CR_0 \cdot e^{-a \times (G/G_{max})}$; ▷ *Adapt Crossover rate CR*
3: **for all** $j \in \{1, ..., n\}$ **do**: ▷ *For all items*
4: $v_j = x_j^a + (x_j^b \oplus x_j^c)$; ▷ *Binary Diff. Mutation*
5: $u_j \leftarrow \begin{cases} v_j & \text{if } rnd(j) \leq CR \vee j \in e, \\ x_j & \text{otherwise.} \end{cases}$
6: **end for**
7: **return** u;
8: **End**

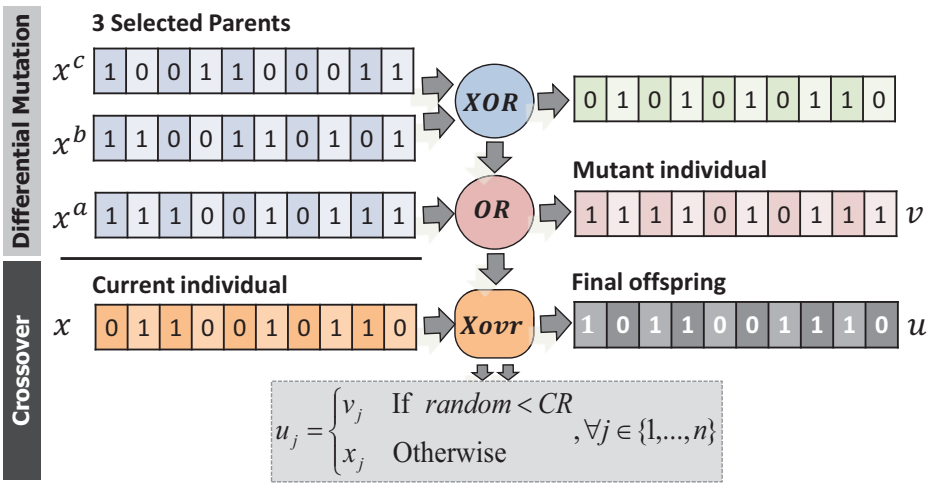


Figure 6.1: The adaptive binary differential evolution

borhood space that leads toward the guiding solution x^t. The relinking procedure has a better chance to investigate in more details the neighborhood of the most promising solutions if relinking starts from the best of x^s and x^t [Ribeiro *et al.* 2002]. In this chapter, the greedy path relinking with two bits per move [Kafafy *et al.* 2012b] is used as an intensification strategy, integrated with the adaptive binary differential evolution. It will be invoked on the higher generations to guarantee applying the relinking process only on high quality solutions. The procedure of the greedy path relinking operator used here is explained in details in section 5.4 as shown in Alg. 5.3.

6.5 Pareto Adaptive ε-Dominance Archiving

It is well known that elitism plays an important role to achieve better convergence for MOEAs. In this work, the elitist schemes are adopted through maintaining an external archive of nondominated solutions discovered during the whole evolutionary process. Also, the maintained solutions inside the external archive is used as global guides to the whole search process. In order to achieve better diversity, the Pareto adaptive $\varepsilon-$dominance, the so-called $pa\varepsilon$-dominance proposed in [Hernández-Díaz *et al.* 2007], is used to update the external archive. At each generation , so as to include a solution into this archive, it is compared with each member already contained in the archive using $pa\varepsilon$-dominance after the grid is generated. The procedure is described in the following.

Every solution in the archive is assigned a box index vector ($B(f) = (B_1, B_2, ..., B_m)^T \in \mathbb{Z}^m$), where m is the number of objectives) which is calculated for the objective vector $f = (f_1, f_2, ..., f_m)$ as follows:

$$B_i(f) = \left\lfloor \frac{\log(\frac{\varepsilon_1^i p^{v_i} - (p^{v_i}-1) f_i}{\varepsilon_1^i})}{\log(\frac{1}{p^{v_i}})} + 1 \right\rfloor \tag{6.4}$$

where p controls the shape of the curve (or surface), T is the number of points desired by the decision maker (desired archive size), ε_1^i is the size of the first box for each dimension in the objective space (Fig. 6.2), and v_i controls the speed of variation. these parameters satisfy the following equations:

$$\begin{cases} \varepsilon_1^i = \frac{(p^{v_i}-1)p^{(T-1)v_i}}{p^{Tv_i}-1} \\ (1-2^{1/p})p^{Tv_i} + 2^{1/p}p^{Tv_i/2} - 1 = 0 \end{cases} \tag{6.5}$$

The box index vector divides the whole objective space into hyper-boxes (see Fig.6.2). With the box index vectors calculated for the offspring y and each archive member x, the offspring y updates the archive as explained in Alg. 6.2, where B_x indicates the box index vector of solution x. More details for this procedure can be found in [Deb *et al.* 2005b].

By using the $pa\varepsilon$-dominance method, the good properties of the original ε-dominance can be maintained, such as ensuring both properties of convergence towards the Pareto optimal set and properties of diversity among the solutions found in a small computation time, while overcoming the main limitation of ε-dominance: the loss of several nondominated solutions from the hyper-grid adopted in the archive.

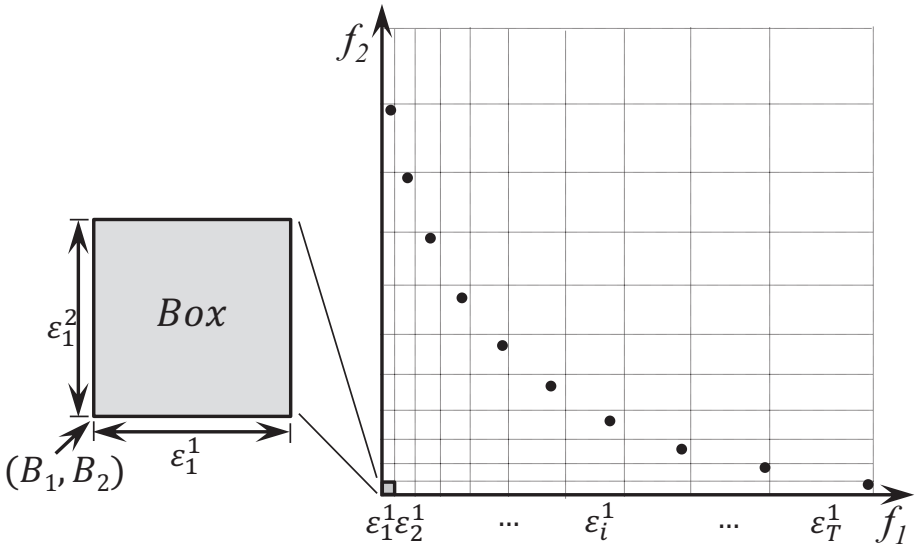


Figure 6.2: The Pareto ε-dominance Archiving

Algorithm 6.2 UpdateArchive$^{pa\varepsilon}(y, Archive)$

Inputs:
y: *Offspring that will update the Archive*
Archive: *The external Archive*

1: **Begin:**
2: **if** B_y of the offspring dominates B_x of any *Archive* member x **then**:
3: Delete all of the dominated *Archive* members
4: Accept the offspring y
5: **else if** B_y is dominated by B_x of any *Archive* member x **then**:
6: Reject the offspring y
7: **else**:
8: **if** y shares the same grid with an *Archive* member x **then**:
9: **if** y dominates x or y is closer to the grid than x **then**:
10: Delete x from the *Archive* and accept offspring y
11: **else**:
12: Reject the offspring y
13: **end if**
14: **else**:
15: Insert offspring y into *Archive*
16: **end if**
17: **end if**
18: **return** *Archive*;
19: **End**

6.6 The Proposed HEMH2

Motivated by the results achieved in [Kafafy *et al.* 2012b], some proposals are adopted to improve HEMH performance and to overcome the limitations dis-

cussed. The main differences between HEMH2 and its predecessor are briefly presented as follows:

- Initial population is created using the simple inverse greedy algorithm (Alg.6.3) for each search direction rather than DMGRASP. The quality of the obtained initial solutions will be affected, but this will give a better chance to the second phase to improve and enhance the search process.
- Instead of collecting all efficient solutions, the Pareto-adaptive epsilon dominance ($pa\varepsilon$-dominance) [Hernández-Díaz *et al.* 2007] explained in section 6.5 is adopted to control the quality and the quantity of the efficient solutions collected in the external archive.
- Dynamic neighborhood size that permit to shrink/extend the neighborhood for each subproblem is considered. Consequently, the parent solutions of a subproblem are always selected from its neighborhood. This can overcome the limitations of the binary differential mutation.
- The adaptive binary DE is used as a reproduction operator instead of crossover and mutation beside the path relinking.
- Path relinking is applied only after a certain number of evaluations as a post optimization strategy. This action guarantees the existence of high quality solutions. Moreover, path relinking flips two bits at each relinking step.
- In HEMH2, local search is avoided either after path relinking or after inverse greedy construction as proposed in HEMH.

Algorithm 6.3 InverseGreedy(x, Λ)

Inputs:
x: *Initial Solution*
$\Lambda = [\lambda_1, ..., \lambda_m]$: *Search direction*

Begin:
$CL \leftarrow \varnothing$;
for all $j \in \{1, ..., n\}$ **do:** $x_j \leftarrow 1$; ▷ *Put all*
while $\exists j | j \notin CL \wedge Min_{j=1}^{n} \left(\frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}} \right)$ **do**:
    $CL \leftarrow$ Append(j); ▷ *set items in ascending all*
end while
while x violates any constraint in formulation in Equ. (2.7) **do**:
    $j \leftarrow$ ExtractTheFirst(CL);
    $x \leftarrow$ RemoveItem(x, j);
end while
return x;
End

Now, the reasons behind the above proposals are explained. Firstly, there is no doubt that generating the initial population using DMGRASP can achieve better quality solutions, but it forces us to use small populations. In some cases, local search highly consumes more time and evaluations to investigate a small specified region in the search space. Consequently, the *main loop* phase has a small chance to improve the search process. To overcome this limitation, the inverse greedy construction is proposed. From the empirical results, the inverse greedy obtains solutions as close as possible to the boundary regions than simple greedy construction. Secondly, using $pa\varepsilon$-dominance will control the size of the archive, especially in many objective cases. Consequently, saving more resources of time and storage space with persevering the quality of the collected solutions. Thirdly, the poor performance of the binary differential mutation occurs when treating differential individuals with large Hamming distance. Selecting parents from the whole population can encourage this scenario. In HEMH2, parents of each subproblem are always selected from its neighborhood which has a dynamic size, this guarantees obtaining individuals with suitable Hamming distances. Fourthly, the adaptive binary DE empirically has the ability to explore the search space better than classical crossover and mutation. Thus, the performance of HEMH will be improved by adopting adaptive binary DE for reproduction rather than crossover. Finally, the proposed path relinking applies two bits flipping/move, that minimizes the whole relinking time. Also, avoiding local search saves both time and evaluations. Fig. 6.3 depicts the whole process in HEMH2.

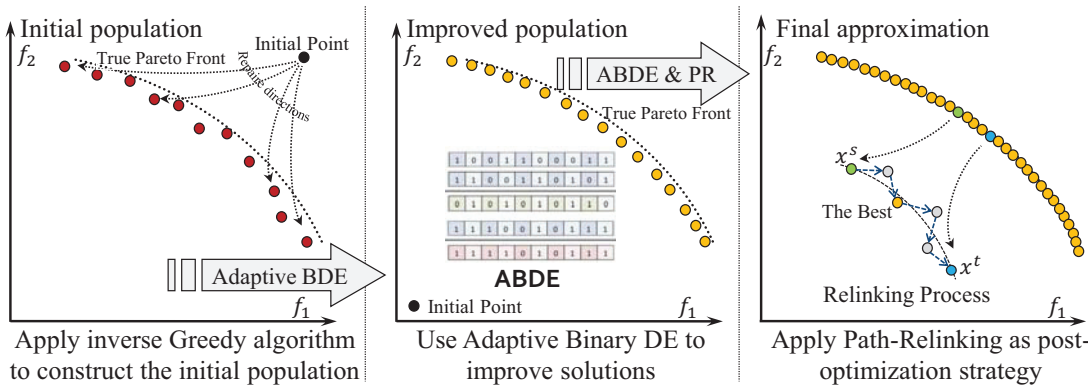


Figure 6.3: The framework of the proposed HEMH2

In Alg. 6.4, the HEMH2 procedure is introduced. Firstly, a set of N evenly distributed weight vectors is created. Then, the neighborhood structure is constructed for each subproblem i by assigning all subproblems sorted increasingly by the Euclidean distance between their weight vectors and the current weight vector Λ_i. After that, an initial population P is created by applying the inverse greedy (Alg.6.3) for each search direction. Now, the main loop is executed until

achieving the maximum evaluations $Mevls$ (line 13). For each subproblem i, SELECTION routine is invoked to determine the *current size* of the neighborhood B_i such that: $|B_i| = T + r$, where T and r represent the number of different and repeated solutions in B_i respectively. This means, the SELECTION routine extends B_i size to guarantee the existence of at least T different solutions and randomly selects three of them x^a, x^b and x^c for reproduction. Two of the three selected parents x^j, x^k are chosen randomly. Then, path relinking is used only if the Hamming distance $\Delta(x^j, x^k)$ is greater than a certain value ε and the number of evaluations $Eval$ exceeds a certain ratio γ of the maximum evaluations $Mevls$ allowed to guarantee applying path relinking on high quality solutions. Else, the adaptive binary DE is applied to generate a new offspring y. The new generated offspring y is evaluated and used to update the neighborhood (B_i) according to the parameter t, which controls the number of replaced solutions. The Archive is also updated by y according to $pa\varepsilon$-dominance [Hernández-Díaz *et al.* 2007]. Finally, the Extreme solutions are added to the archive which is returned as an output.

In order to study the effects of both adaptive binary DE and path relinking operators distinctly, two additional algorithms variants called HEMH_{de} and HEMH_{pr} are considered. Both of them have the same procedure as HEMH2 explained in Alg. 6.4 except that HEMH_{de} only adopts adaptive binary DE for reproduction. Whereas, HEMH_{pr} replaces the adaptive binary DE in HEMH2 procedure by crossover and mutation.

6.7 Experimental Design

In this chapter, both MOEA/D and HEMH are involved to verify our proposals. The comparative study for different algorithms is carried out on a set of test instances from the literature [Zitzler & Thiele 1999] listed in table 6.1. All experiments are performed on a PC with Intel Core i5-2400 CPU, 3.10 GHz and 4.0 GB of RAM.

6.7.1 Parameter settings

Here, the different parameters used for each algorithm are discussed. For MOEA/D and our proposals HEMH_{de}, HEMH_{pr} and HEMH2, the parameter H which controls the population size N by the relation ($N = C_{m-1}^{H+m-1}$) is determined for each instance in table 6.1 according to the complexity. The initial population used in MOEA/D is randomly generated such that each member $x = (x_1, ..., x_n)^T \in \{0, 1\}^T$, where $x_i = 1$ with probability equal to 0.5. For HEMH, the parameter $\acute{H}$ that controls the population size $\acute{N}$ is

Algorithm 6.4 HEMH2$(N, T, t, \varepsilon, \gamma, CR_0, a)$

Inputs:
N: *Population size or no. of subproblems*
T: *Min. neighborhood size*
t: *Max. replaced solutions*
ε: *Min. hamming distance*
γ: *Controls Path-relinking execution*
$CR_0 \in [0,1]$: *Crossover rate*
a: *Plus constant*

1: **Begin:**
2: $W_v \leftarrow \{\Lambda^1, ..., \Lambda^N\}$; ▷ *Initialize set of N evenly distributed weight vectors*
3: **for** $i \leftarrow 1$ to N **do**: ▷ *Construct Neighborhoods*
4: $B_i \leftarrow [i1, ..., iN]$; ▷ *where* $\Lambda^{i1}, ..., \Lambda^{iN}$ *are*
5: **end for** ▷ *increasingly sorted by Euclidean Distance to* Λ^i
6: $Arch \leftarrow \varnothing$; ▷ *Empty archive*
7: $Evals \leftarrow 0$;
8: **for** $i \leftarrow 1$ to N **do**: ▷ *Initialization phase*
9: $x^i \leftarrow$ InverseGreedy(x^i, Λ^i); ▷ *Initialize* x^i *by inverse greedy method*
10: $P \leftarrow$ AddSubProblem(x^i, Λ^i); ▷ *add* i^{th} *subproblem to the population P*
11: $Extremes \leftarrow$ Update(x^i); ▷ *update set of extreme solutions*
12: $Evals \leftarrow Evals + 1$;
13: **end for**
14: **while** $(Evals < MaxEvals)$ **do**: ▷ *The main Loop*
15: **for** $i \leftarrow 1$ to N **do**: ▷ *for each subproblem i*
16: $x^a, x^b, x^c \leftarrow$ Selection(B_i, i); ▷ *Where:* $x^i \neq x^a \neq x^b \neq x^c$
17: $x^j, x^k \leftarrow$ RandSelection(x^a, x^b, x^c);
18: $D \leftarrow \Delta(x^j, x^k)$; ▷ *Compute Hamming distance*
19: $E \leftarrow \gamma \times MaxEvals$; ▷ *min. eval for PR*
20: **if** $(D \geq \varepsilon \wedge Eval \geq E)$ **then**:
21: $y \leftarrow$ PathRelinking(x^j, x^k, Λ^i);
22: **else**:
23: $u \leftarrow$ AdaptiveBinaryDiffEvol$(x^i, x^a, x^b, x^c, CR_0, a)$;
24: $y \leftarrow$ Repair(u, Λ^i)
25: **end if**
26: $P \leftarrow$ UpdateSolutions(y, t, B_i); ▷ *Update the Population P*
27: $Arch \leftarrow$ UpdateArchive$^{pa\varepsilon}(y, Arch)$; ▷ *Update Archive by Pareto Adaptive Epsilon*
28: $Extremes \leftarrow$ Update(y);
29: $Evals \leftarrow Evals + 1$;
30: **end for**
31: **end while**
32: $Arch \leftarrow$ AddExtremes$(Extremes)$; ▷ *Add extremes to the archive*
33: **return** $Arch$;
34: **End**

also considered. HEMH uses the parameters α=0.1 and β=0.5. The maximum number of evaluations ($Mevls$) is used as a stopping criterion for each algorithm. For fair comparison, the same archiving strategy based on $pa\varepsilon$-dominance [Hernández-Díaz *et al.* 2007] are applied to each algorithm to get the final approximation set. The $pa\varepsilon$-dominance uses the archive size (A_s) listed in table 6.1. HEMH applies the path relinking proposed here. Single-point crossover

and standard mutation are considered. Mutation was performed for each item independently with probability $(1/n)$. The other control parameters are listed in table 6.2. Finally, the statistical analysis is applied on 30 independent runs for each instance.

Table 6.1: Set of knapsack instances

Instances	**Knaps.**(m)	**Items**(n)	$N(H)$	$\acute{N}(\acute{H})$	A_s	$Mevls$
KSP252	2	250	150(149)	75(74)	150	75000
KSP502	2	500	200(199)	100(99)	200	100000
KSP752	2	750	250(249)	125(124)	250	125000
KSP253	3	250	300(23)	153(16)	200	100000
KSP503	3	500	300(23)	153(16)	250	125000
KSP753	3	750	300(23)	153(16)	300	150000
KSP254	4	250	364(11)	165(8)	250	125000
KSP504	4	500	364(11)	165(8)	300	150000
KSP754	4	750	364(11)	165(8)	350	175000

Table 6.2: Set of common parameters

Parameters	**Algorithms**			
	HEMH	**HEMH**$_{de}$	**HEMH**$_{pr}$	**HEMH2**
Neighborhood size: T	10	10	10	10
Max. replaced solutions: t	2	2	2	2
Parents selection: δ	0.9	-	-	-
Min. support: σ	$\{1\}$	-	-	-
Ratio controls PR: γ	-	-	0.8	0.8
Min. Hamming Distance: ε	10	-	10	10
initial crossover rate: CR_0	-	0.4	-	0.4
Plus constant: a	-	2	-	2

6.7.2 Assessment Metrics

In order to assess the quality of the solutions obtained by each algorithm, some of the quality assessment indicators are used in these experiments including binary and unary indicators. These indicators include the set coverage I_C indicator which is used to compare each pair of MOEAs as discussed in section 2.6.2.1. They also include indicators to assess each MOEA independently such as referenced hypervolume (I_{Rhyp}), generational distance (I_{GD}), inverted generational distance (I_{IGD}) and R_3 (I_{R_3}) indicators which are presented in sections 2.6.2.2, 2.6.2.3, 2.6.2.3 and 2.6.2.5 respectively.

In these experiments, the reference set P^* for each instance is formed by gathering all efficient solutions found by all algorithms in all runs. Also, all approximation sets are normalized in the range [1,2].

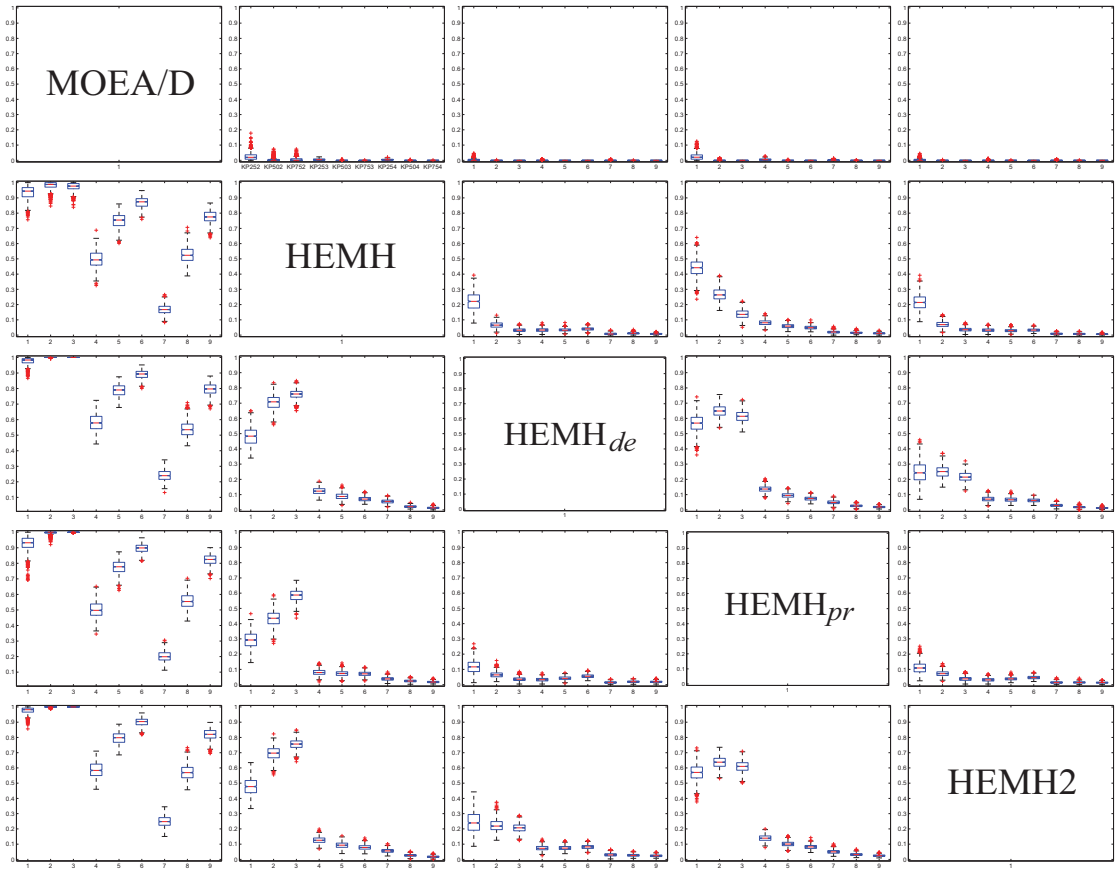


Figure 6.4: Results of the coverage I_C indicator

6.8 Experimental Results

In this section, the different simulation results are shown in details. Firstly, Fig.6.4 depicts the results of I_C metric. It contains a chart (with scale 0 at the bottom and 1 at the top) for the ordered pairs depicted. Each chart consists of nine box plots representing the distribution of I_C values. Each box plot (from left to right) represents an instance in table 6.1 (from top to down) respectively. Besides, the median values of I_C indicator are also presented in table B.11. It is clear from the results in both Fig.6.4 and table B.11 that all the proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D in all test instances. Whereas, Both HEMH2 and HEMH$_{de}$ outperform HEMH for all test instances. Also, HEMH$_{pr}$ slightly performs better than HEMH in most test instances.

Table 6.3: Average results of the referenced hypervolume (I_{Rhyp})

Instances	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	4.66E-02	1.02E-02	**6.07E-03**	9.33E-03	6.08E-03
KSP502	5.67E-02	1.55E-02	5.57E-03	1.16E-02	**5.56E-03**
KSP752	4.73E-02	1.64E-02	**3.84E-03**	7.34E-03	3.94E-03
KSP253	2.24E-01	1.21E-01	8.85E-02	1.09E-01	**8.77E-02**
KSP503	2.76E-01	1.16E-01	**7.39E-02**	9.01E-02	**7.39E-02**
KSP753	2.89E-01	8.85E-02	6.48E-02	7.63E-02	**6.39E-02**
KSP254	8.56E-01	5.55E-01	**4.26E-01**	4.90E-01	4.27E-01
KSP504	1.07E+00	4.53E-01	3.96E-01	4.10E-01	**3.84E-01**
KSP754	1.23E+00	3.93E-01	3.54E-01	3.64E-01	**3.41E-01**

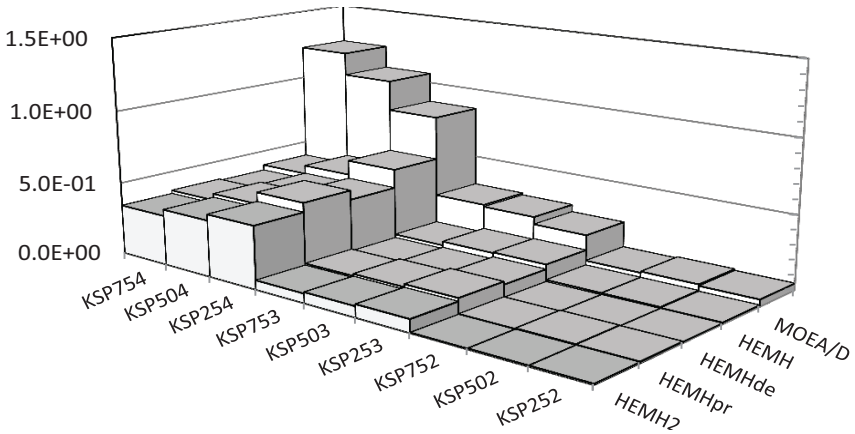


Figure 6.5: Average results of the referenced hypervolume I_{Rhyp}

In table 6.3, the average values of the referenced hypervolume I_{Rhyp} indicator are listed. Additionally, Fig. 6.5 visualizes theses values. The details of the statistical analysis for I_{Rhyp} are also provided in table B.12. According to these results, it is clear that HEMH2 has the best performance in all test instances except KSP252, KSP752 and KSP254 where HEMH$_{de}$ has the best performance. Moreover, the I_{Rhyp} results indicate taht HEMH2 and the two proposed variants HEMH$_{de}$ and HEMH$_{pr}$ outperform both MOEAD and the original HEMH in all test instances used.

Table 6.4 contains the average values for the generational distance I_{GD} indicator. Also, Fig. 6.6 visualizes theses values. The details of the statistical analysis for I_{GD} are shown in table B.13. These results indicate that HEMH2 has the superiority followed by HEMH$_{de}$ in all test instances except KSP502, KSP752 and KSP254 where HEMH$_{de}$ has the best performance. According to the generational distance I_{GD} results, the three proposed variants HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform both the original MOEA/D and HEMH.

Table 6.4: Average results of the generational distance (I_{GD})

Instances	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	1.50E-03	5.17E-04	3.40E-04	5.74E-04	**3.37E-04**
KSP502	1.53E-03	4.41E-04	**1.44E-04**	3.12E-04	1.54E-04
KSP752	1.33E-03	4.77E-04	**7.52E-05**	1.91E-04	7.81E-05
KSP253	1.98E-03	6.83E-04	3.92E-04	6.82E-04	**3.88E-04**
KSP503	2.25E-03	4.95E-04	2.76E-04	4.25E-04	**2.70E-04**
KSP753	2.16E-03	3.63E-04	2.17E-04	2.77E-04	**2.07E-04**
KSP254	2.52E-03	1.14E-03	**6.07E-04**	9.13E-04	6.14E-04
KSP504	3.45E-03	6.63E-04	4.08E-04	5.14E-04	**3.62E-04**
KSP754	3.79E-03	4.91E-04	2.88E-04	3.56E-04	**2.53E-04**

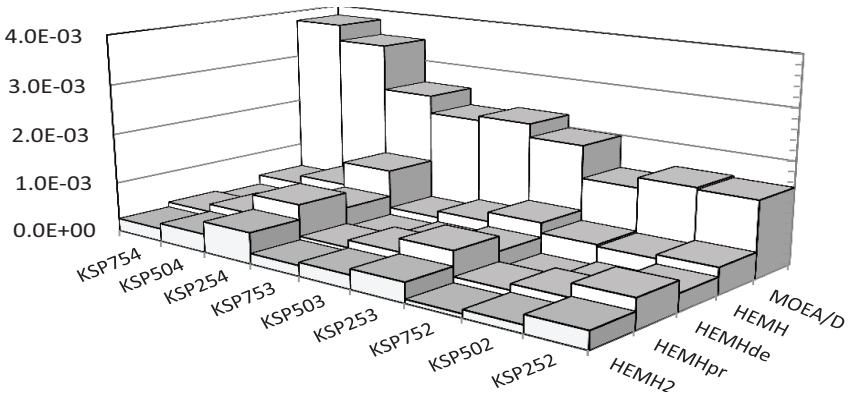


Figure 6.6: Average results of the generational distance (I_{GD})

In tables 6.5, the average values of the inverted generational distance I_{IGD} indicators are listed. Additionally, Fig. 6.7 visualizes theses values. The details of the statistical analysis for I_{IGD} indicator are also provided in table B.14 . According to these results, it is clear that HEMH$_{de}$ has the best performance in all test instances except KSP754 in which HEMH2 has the best performance. Moreover, the I_{IGD} results indicate that all the proposed variants HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform both MOEA/D and the original HEMH in all test instances used in this study.

The average results for the R_3 I_{R_3} indicator are listed in table 6.6. The visualization for these results are illustrated in Fig. 6.8. Moreover, the details of the statistical analysis for I_{R_3} indicator are also listed in table B.15. These results indicate the superiority of HEMH$_{de}$ in handling bi-objectives test instances, whereas HEMH2 has the superiority in solving test instances with three and four objective functions. The I_{R_3} indicator results confirm the results of the other indicators where the proposed HEMH2 and its variants HEMH$_{de}$ and HEMH$_{pr}$ outperform both MOEA/D and the original HEMH in all test instances used in this study.

Table 6.5: Average results of the Inverted. Generational Distances (I_{IGD})

Instances	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	9.32E-04	4.83E-04	**3.49E-04**	4.51E-04	3.50E-04
KSP502	7.31E-04	2.72E-04	**1.31E-04**	2.10E-04	1.32E-04
KSP752	5.41E-04	2.43E-04	**7.89E-05**	1.14E-04	7.95E-05
KSP253	6.31E-04	4.48E-04	**3.83E-04**	4.41E-04	3.84E-04
KSP503	5.28E-04	3.33E-04	**2.58E-04**	2.99E-04	2.60E-04
KSP753	4.55E-04	2.46E-04	**1.93E-04**	2.24E-04	1.95E-04
KSP254	6.75E-04	5.69E-04	**4.88E-04**	5.33E-04	4.91E-04
KSP504	5.95E-04	4.04E-04	**3.46E-04**	3.67E-04	3.46E-04
KSP754	5.46E-04	3.28E-04	2.71E-04	2.85E-04	**2.70E-04**

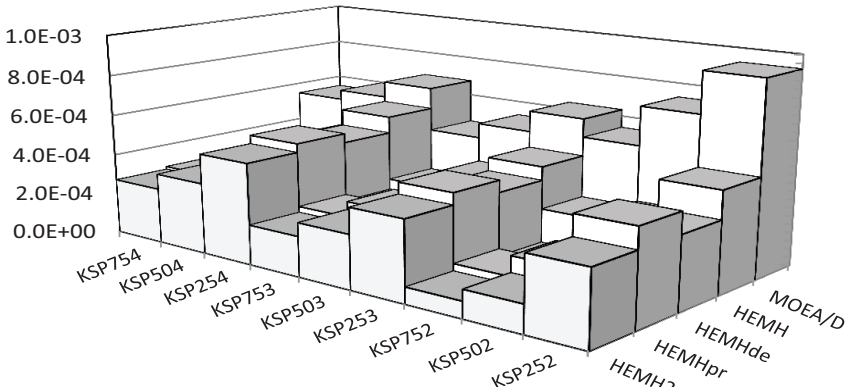


Figure 6.7: Average results of the Inverted. Generational Distances (I_{IGD})

In general, based on those results, it is clear that the proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D and HEMH. Since they achieve the minimum average values for all assessment indicator adopted in the study. Moreover, the proposed HEMH2 and HEMH$_{de}$ have the superiority for all test instances, followed by HEMH$_{pr}$. In some cases, it is observed that HEMH$_{de}$ achieves better results than HEMH2 and HEMH$_{pr}$, depending upon the adaptive binary differential evolution. This reflects that the adaptive binary differential evolution capabilities in exploring the search space are more effective than path relinking and classical crossover and mutation.

Table 6.6: Average results of the R_3 indicator (I_{R_3})

Instances	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	6.05E-03	1.88E-03	**1.29E-03**	2.32E-03	1.30E-03
KSP502	7.30E-03	2.06E-03	**6.89E-04**	1.62E-03	7.12E-04
KSP752	6.88E-03	2.47E-03	**5.46E-04**	1.21E-03	5.54E-04
KSP253	1.11E-02	5.62E-03	4.49E-03	5.24E-03	**4.48E-03**
KSP503	1.34E-02	5.16E-03	3.83E-03	4.60E-03	**3.78E-03**
KSP753	1.42E-02	4.25E-03	3.38E-03	3.92E-03	**3.32E-03**
KSP254	1.61E-02	9.70E-03	7.86E-03	8.84E-03	**7.84E-03**
KSP504	2.02E-02	7.91E-03	7.18E-03	7.40E-03	**7.06E-03**
KSP754	2.39E-02	7.02E-03	6.02E-03	6.26E-03	**5.82E-03**

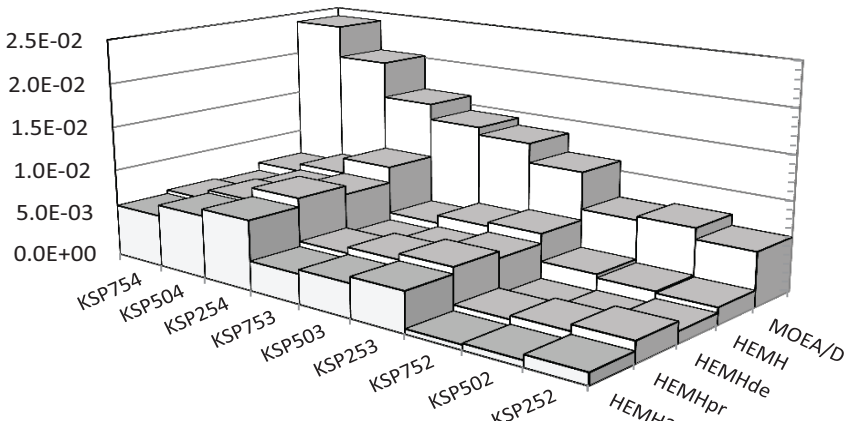


Figure 6.8: Average results of the R_3 indicator (I_{R_3})

6.9 Summary

In this chapter, an improved hybrid evolutionary metaheuristics HEMH2 and two other variants called HEMH$_{de}$ and HEMH$_{pr}$ are proposed to enhance HEMH performance. The HEMH2 adopts the inverse greedy procedure in its initialization phase. Both adaptive binary DE and greedy path relinking operators are used. The HEMH$_{de}$ only uses adaptive binary DE to generate the new offspring whereas, HEMH$_{pr}$ uses crossover and mutation beside path-relinking. The proposals are compared with the original MOEA/D and their predecessor HEMH using a set of MOKP instances from the literature. A set of quality assessment indicators is also used to assess the performance. The experimental results indicate the superiority of all proposals over the original MOEA/D and the predecessor HEMH based on the assessment indicators used in this study. According to these results, we can deduce that the adaptive binary differential evolution included in both HEMH2 and HEMH$_{de}$ has better exploration capabilities which overcome the local search capabilities contained in the original HEMH. Therefore

both of HEMH2 and HEMH$_{de}$ outperform HEMH. In some cases, HEMH$_{de}$ can achieve highly competitive results compared with HEMH2 based on the adaptive binary differential evolution which can achieve better performance than greedy path-relinking. In the next chapter, we will extend our proposals to handle multiobjective optimization problems in the continuous search domains.

Chapter 7

A Search Strategy Adaptation based Approach

Contents

7.1 Introduction 155

7.2 A Review of Search Operators 156

7.2.1 Genetic operators 156

7.2.2 Differential Evolution operator 158

7.2.3 Particle swarm optimization 158

7.2.4 Guided Mutation operator 159

7.3 The proposed HESSA 159

7.3.1 Multiple Search Strategies Adaptation 160

7.3.2 HESSA framework 161

7.3.3 HESSA procedure 162

7.4 Experimental Design 164

7.4.1 Parameter settings 164

7.4.2 Assessment Metrics 167

7.5 Experimental Results 169

7.6 Summary 175

7.1 Introduction

Hybrid evolutionary algorithms have been successfully applied to solve numerous multiobjective optimization problems (MOP). In our previous research work in [Kafafy *et al.* 2011] and [Kafafy *et al.* 2012a] which are presented in the previous chapters 4 and 6 respectively, the new hybrid evolutionary metaheuristics HEMH and its improved version HEMH2 are proposed for discrete search domains. According to the experimentation, they have the ability to enhance the

search capabilities through adopting different combinations of search operators combined with the MOEA/D framework. Motivated by these results, this chapter tries to extend this work to the continuous search domains. It studies the cooperation of different search operators and analyzes its effects on handling continuous MOPs. In this chapter, a new hybrid evolutionary approach with search strategy adaptation (HESSA) [Kafafy *et al.* 2013] which incorporates a pool of adaptive search strategies within the MOEA/D framework is presented. The main goals are to capture the benefits of those strategies with providing cooperation and integration among them for improving search capabilities. Also, to make the approach capable of selecting the most suitable search strategy according to the problem on hand. In HESSA, the search process is carried out through adopting a pool of different search strategies, each of which has a specified success ratio. A new offspring is generated using a randomly selected strategy. Then, according to the success of the generated offspring to update the population and/or the archive, the success ratio of the selected strategy is adapted. This provides the ability for HESSA to adopt the appropriate search strategy according to the problem on hand. Furthermore, the cooperation among different strategies leads to improve the exploration and the exploitation of the search space. The proposed pool is combined to a suitable evolutionary framework for supporting the integration and cooperation. Moreover, the efficient solutions explored over the search are collected in an external repository to be used as global guides for the whole search process. The remainder of this chapter is organized as follows. In section 7.2, some of the different search operators are overviewed. The proposed HESSA is presented in section 7.3. In additions, the experimental design and results are involved in sections 7.4 and 7.5 respectively. Finally, section 7.6 presents the conclusions and some further directions.

7.2 A Review of Search Operators

In chapter 3, global search techniques especially metaheuristic based algorithms are reviewed in details. Here, the concentration is directed to the different search operators which used in this work. In this section, the components of the search strategies used in this research will be briefly reviewed as follows:

7.2.1 Genetic operators

As discussed in section 3.6.1.1, crossover and mutation are the two most popular genetic operators. Crossover is the process of exchanging the genetic material of the parents to create new offspring. Whereas, the mutation operator is used to preserve the diversity of the population during generations. In the literature, various types of crossover and mutation were proposed. These

types are successfully used to handle different types of optimization problems in the continuous search domains. In this context, the SBX crossover [Deb & Agrawal 1995], Multiple parents crossover [Elsayed *et al.* 2011] and polynomial mutation [Deb & Agrawal 1995] will be focused.

7.2.1.1 SBX crossover

The simulated binary crossover (SBX) is widely used in practice. It has been found to work well in many test problems that have a continuous search space. From a pair of parents x^a and x^b, the SBX crossover produces an offspring y as follows:

$$y = \begin{cases} \frac{1}{2}\left((1+\beta)x^a + (1-\beta)x^b\right) & \text{if: } p \leq 0.5 \\ \frac{1}{2}\left((1-\beta)x^a + (1+\beta)x^b\right) & \text{otherwise} \end{cases} \tag{7.1}$$

$$\beta = \begin{cases} (2u)^{1/(1+\eta_c)}, & u \leq 0.5 \\ (1/(2-2u))^{1/(1+\eta_c)}, & \text{otherwise} \end{cases} \tag{7.2}$$

where $p, u \in [0, 1]$ are two uniform random numbers and η_c is the distribution index.

7.2.1.2 Multi-parents crossover

various multi-parents crossover were proposed in the literature for continuous search domains such as simplex (SPX) [Tsutsui *et al.* 1999b], parents centric (PCX) [Deb *et al.* 2002], etc. However, the new multiple parents crossover (MPC) proposed in [Elsayed *et al.* 2011] will be used here. According to Equ. 7.3, the MPC crossover constructs a new offspring y, from three different randomly selected parents x^a, x^b and x^c as follows:

$$y = \begin{cases} x^a + \beta \times (x^b - x^c) & \text{if: } p \leq \frac{1}{3} \\ x^b + \beta \times (x^a - x^c) & \text{if: } \frac{1}{3} < p \leq \frac{2}{3} \\ x^c + \beta \times (x^a - x^b) & \text{otherwise} \end{cases} \tag{7.3}$$

where $\beta \sim N(\mu, \sigma)$ is a Gaussian random number and $p \in [0, 1]$ is a uniform random number.

7.2.1.3 Polynomial mutation

In polynomial mutation, the probability to produce a child near to the parent is greater than the probability to produce one distant it. The mutant offspring $\acute{x}$ can be produced as:

$$\acute{x}_j = \begin{cases} x_j + \delta_j \times (b_j - a_j) & \text{with probability } p_m \\ x_j & \text{with probability } 1 - p_m \end{cases} \tag{7.4}$$

$$\forall \delta_j = \begin{cases} (2u_j)^{1/(1+\eta_m)} - 1, & u_j \leq 0.5 \\ 1 - (2 - 2u_j)^{1/(1+\eta_m)} & \text{otherwise} \end{cases} \tag{7.5}$$

where $u_j \in [0, 1]$ is a random number. The distribution index η_m and the mutation rate p_m are two control parameters. a_j and b_j are the lower and the upper limits of x_j.

7.2.2 Differential Evolution operator

As reviewed in section 3.6.2.1, differential evolution (DE) uses differential mutation to employ the difference vectors built with pairs of candidate solutions in the search domain. Each difference vector is scaled and added to another candidate solution, producing the so-called mutant vector. Then, DE recombines the mutant vector with the parent solution to generate a new offspring. The offspring replaces the parent only if it has an equal or better fitness. There are different strategies to carry out this process such as "DE/rand/n/bin", "DE/best/n/exp", "DE/rand-to-best/n/bin", etc, where n is the number of difference vectors used [Price *et al.* 2005]. In this work, the "DE/rand/1/bin" strategy is considered. DE has some control parameters as scaling factor F, that used to scale the difference vectors, and crossover rate CR. Given a population P of N individuals, the idea is to randomly select three distinct individuals x^a, x^b and x^c from P for each target individual $x^i \in P$, $\forall i \in \{1, ..., N\}$. The mutant individual v^i is produced according to Equ. 7.6. Then, the *binomial crossover* is applied on v^i and x^i to produce a new offspring u^i as:

$$v^i = x^a + F \times (x^b - x^c) \tag{7.6}$$

$$u_j^i = \begin{cases} v_j^i & \text{if } rnd \leq CR, \text{ or } j = j_{rnd}, \\ x_j^i & \text{otherwise}, \forall j = 1, ..., n. \end{cases} \tag{7.7}$$

where $rnd \in [0, 1]$ and $j_{rnd} \in \{1, ..., n\}$ is a random chosen index to insure that at least one component of u^i is contributed by v^i, n is the individual length and $CR \in [0, 1]$.

7.2.3 Particle swarm optimization

PSO is a population-based stochastic optimization technique that simulates the social behavior of bird flocking and fish schooling as mentioned in section 3.6.3.1. It was originally proposed in [Kennedy & Eberhart 1995]. PSO consists of a population of particles (solutions). Each particle i has its own position x^i and moves through the search space with an adaptable velocity v^i towards the best position that it has achieved x_{pb}^i and the overall best solution x_{gb}^i. For each ith particle at generation t, the velocity and the new position for the next generation

can be evaluated as follows:

$$v^i(t{+}1) = w \cdot v^i(t) + c_1 r_1 (x^i_{pb} - x^i(t)) + c_2 r_2 (x^i_{gb} - x^i(t)) \tag{7.8}$$

$$x^i(t{+}1) = x^i(t) + v^i(t{+}1) \tag{7.9}$$

where $w \geq 0$ represents the inertia weight, c_1 and c_2 are the acceleration coefficients and r_1, $r_2 \sim U(0,1)^n$. For each j, if $x^i_j(t+1)$ violates its domain $[a_j, b_j]$, it will be repaired and also its velocity $v^i(t+1)$ will be reset as follows:

$$\begin{aligned} x^i_j(t+1) &= \begin{cases} a_j & \text{if } x^i_j(t+1) < a_j \\ b_j & \text{if } x^i_j(t+1) > b_j \end{cases} \\ v^i_j(t+1) &= -\gamma v^i_j(t+1) \end{aligned} \tag{7.10}$$

where a_j and b_j are lower and upper bounds of the jth component respectively. Here, the parameter γ is set to 1.

7.2.4 Guided Mutation operator

Guided mutation proposed in [Hsieh *et al.* 2007] provides an integration between global and local search capabilities, through guiding the rotation of the evolutionary strategy (ES) mutation ellipses, for global search, and using the regular ES operation to conduct local search to find the promising solution. In guided mutation, the new solution y is generated from its parent x using the guided target solution t as follows:

$$y_j = \begin{cases} x_j + 0.5(t_j - x_j) \times r + R \times N(0,1) & \text{with } p_m \\ x_j + 0.5(t_j - x_j) \times r & \text{with } 1 - p_m \end{cases} \tag{7.11}$$

$$\forall R = \begin{cases} 0.1 \times |t - x| & \text{if } 0.1 \times |t - x| > \mu \\ \mu & \text{otherwise} \end{cases} \tag{7.12}$$

where $r \sim N(0,1)$ is a Gaussian number and p_m is the mutation rate. The new offspring y consists of the current position of its parent x, the guided vector derived from its target t and the mutation step R which specified by the distance $|t - x|$ and bounded by the control parameter μ. Finally, An illustration of the guided mutation operator is depicted in Fig. 7.1.

7.3 The proposed HESSA

In this context, the proposed HESSA is presented in more details. In the research work in [Kafafy *et al.* 2011, Kafafy *et al.* 2012b, Kafafy *et al.* 2012a], the influence of incorporating different cooperative metaheuristics in MOEA/D framework was examined for discrete search domains. The achieved results motivate

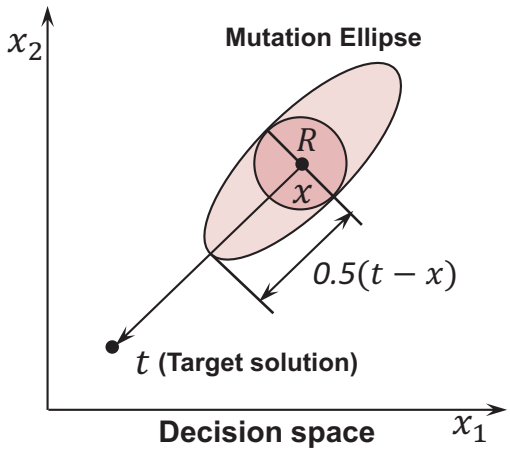


Figure 7.1: Illustration of Guided Mutation

us to extend the idea to the continuous case. However, an adaptive multiple search strategies are adopted for tackling continuous search domains. In the following subsections, the components of the proposed HESSA are discussed.

7.3.1 Multiple Search Strategies Adaptation

In this work, a pool of multiple search strategies is adopted to generate the new offspring solutions instead of using a single strategy as depicted by Fig. 7.2. To generate a new offspring, the candidate pool is accessed for selecting one search strategy for each target individual in the current population. During evolution, each certain number of consecutive pool's invokes is considered as a learning period (LP). The more successfully one strategy behaved in the previous learning period to generate promising solutions, the more probability it will be chosen in the current learning period to be used for generating the new offspring solutions. At each learning period, the probability of selecting each strategy from the candidate pool are summed to 1. These probabilities are adapted gradually during the evolution process. In the initial learning period, all strategies have the same chance to be selected, i.e., each strategy k has a probability $p_k = \frac{1}{K}$, where K is the total number of strategies in the candidate pool. During each learning period, each strategy k can be chosen to generate the new solution according to its probability p_k using the stochastic universal selection [Baker 1987]. The number of selecting each strategy k is represented by $calls_k$. Each strategy is considered to achieve a success if it has the ability to generate an offspring capable of updating the current population. The number of successful calls for each strategy k is registered by suc_k. The number of invokes for the candidate pool is expressed as: $calls_{tot} = \sum_{l=1}^{L} \sum_{k=1}^{K} calls_{k,l}$, where, L is the total number of learning periods in the whole evolution. However, after each learning period l

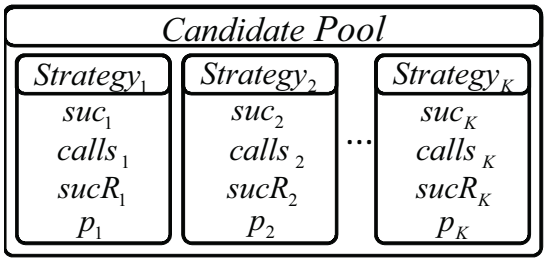


Figure 7.2: The structure of the candidate pool.

Table 7.1: Set of reproduction strategies used

Strategy	Description
SBXPM	The SBX crossover is applied on two parents followed by polynomial mutation to generate an offspring.
DEXPM	The differential evolution is applied on three selected parents followed by polynomial mutation.
MPCPM	The multiple parent crossover is applied on three selected parents followed by polynomial mutation.
GM	Guided mutation is used to produce an offspring from its parent and the global guide solution.
PSO	Particle swarm computes a new position from the current parent, its personal best and global guide.

(when $calls_{tot}\%LP = 0$), the probability of selecting each strategy k for the next learning period $p_{k,l+1}$ will be adapted according to the following formulas:

$$p_{k,l+1} = \frac{sucR_{k,l}}{\sum_{k=1}^{K} sucR_{k,l}} \tag{7.13}$$

$$sucR_{k,l} = \begin{cases} \frac{suc_{k,l}}{calls_{k,l}} + \varepsilon & \text{if } calls_{k,l} > 0, \forall k, l \\ \varepsilon & \text{otherwise} \end{cases} \tag{7.14}$$

where $sucR_{k,l}$ is the success rate of the kth strategy at the learning period l. The small constant value $\varepsilon = 0.01$ is used to avoid the possible null success rates. Consequently, the strategies with null success rate have a chance to be chosen for generating offspring. Both $suc_{k,l}$ and $calls_{k,l}$ represent the number of successful invokes and the total number of invokes of the kth strategy at the learning period l.

7.3.2 HESSA framework

Like MOEA/D [Zhang & Li 2007], the proposed approach uses a decomposition technique to convert the MOP formulated in Equ. 2.1 into a set of single objective subproblems. The weighted Tchebycheff approach described in formula in Equ. 2.19 is used in this study. However, if we have a set of N evenly distributed

weight vectors $\{\Lambda^1, ..., \Lambda^N\}$, correspondingly after decomposition, we have N single objective subproblems. The proposed algorithm attempts to simultaneously optimize these subproblems. Each subproblem i has its own set of neighbors called B_i, which includes all the subproblems with the T closest weight vectors $\{\Lambda^{i1}, ..., \Lambda^{iT}\}$ to Λ^i in terms of Euclidean distance. The structure of the proposed framework is briefed as follows:

- A population P of N individuals, $P = \{x^1, ..., x^N\}$, where x^i represents the current solution of the ith subproblem. Each individual x^i has its own velocity v^i, its personal best position x^i_{pb} and its age a_i.
- A set of N evenly distributed weight vectors $\{\Lambda^1,...,\Lambda^N\}$, correspond to the N subproblems. Each $\Lambda = [\lambda_1,..., \lambda_m]$ has m components correspond to m-objectives, such that: $\sum_{i=1}^{m} \lambda_i = 1$, $\forall \lambda_i \in \{0/H, 1/H, ..., H/H\}$ and $N = C_{m-1}^{H+m-1}$, $\forall H \in \mathbb{Z}^+$.
- A neighborhood B_i for each subproblem $i \in \{1, ..., N\}$, which includes all subproblems with the T closest weight vectors $\{\Lambda^{i1}, ..., \Lambda^{iT}\}$ to Λ^i.
- A set of adaptive reproduction strategies contained in a *Pool* for generating new solutions. Each strategy is selected according to its probability as mentioned above. Table 7.1 summarizes the set of adopted strategies.
- An external *archive* to collect efficient solutions explored over the search process. The archive also plays the role of global leaders repository.

Finally, the structure of HESSA is depicted in Fig. 7.3.

7.3.3 HESSA procedure

After constructing the proposed framework, the proposed approach implements two basic phases. The first one is the *initialization* phase in which an initial population is randomly generated, whereas the second is the *main-loop* in which the search efforts are conducted to improve the initial population. The whole process is summarized in Alg. 7.1. Firstly, in lines (2-5), a set of N evenly distributed weight vectors is initialized. Then, the neighborhood structure B_i is constructed for each subproblem i by assigning all subproblems with the T closest weight vectors to Λ_i. The candidate *Pool* is also built using the adopted reproduction strategies. The archive and the evaluation counter are initialized. Secondly, the initial population is constructed in lines (6-18). For each subproblem i, the current solution x^i is randomly initialized. Then, x^i is evaluated and used to update the reference point r^* [Zhang & Li 2007], the personal best x^i_{pb} and the *archive*. The velocity v^i and the age a_i are also initialized by 0. The ith subproblem is

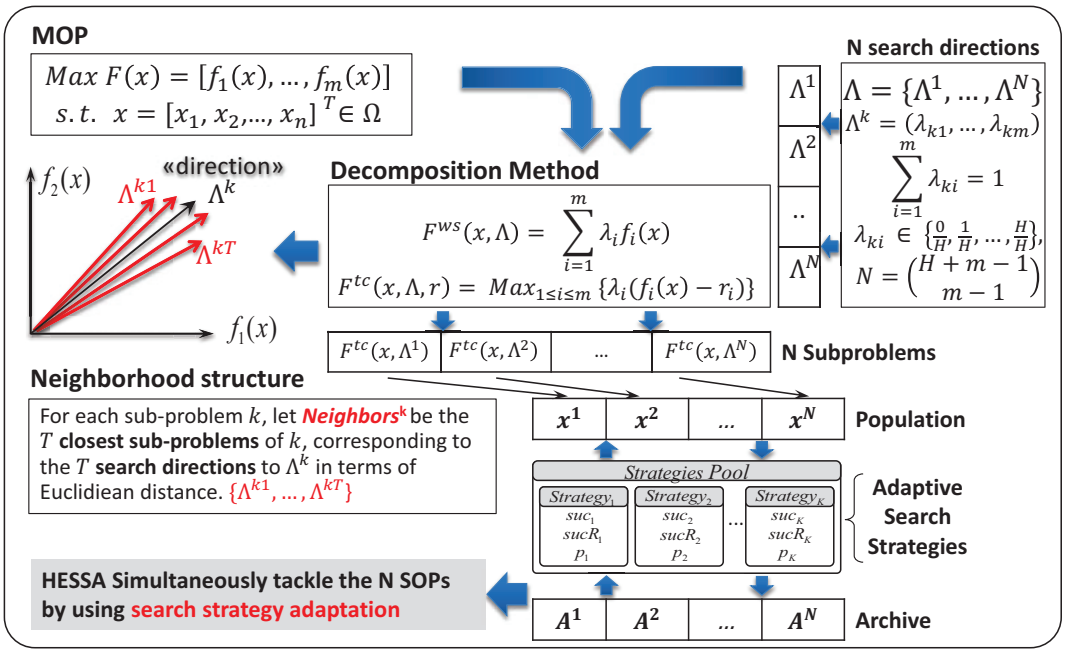


Figure 7.3: HESSA Algorithm structure

appended to the population P. Now, the *main-loop* is executed until achieving the maximum evaluations $Mevals$ (lines 19-37). For each subproblem i, the mating/updating range M_i is chosen to be either the neighborhood B_i or the whole population. Then, three different parent solutions are randomly selected from M_i for reproduction. The global leader x^i_{gb} is randomly selected from the archive. A reproduction strategy S_k is also selected from the *Pool* for generating the new offspring y through invoking the REPRODUCTION module. According to the selected strategy S_k, the offspring y is generated. In case of using the guided mutation or the particle swarm, the age parameter a_i controls the generation process. In this case, if a_i exceeds the maximum allowable age T_a, a Gaussian value as: $N(\frac{1}{2}[x^i_{gb} - x^i_{pb}], |x^i_{gb} - x^i_{pb}|)$ is assigned to y. After that, the offspring y is evaluated and used to update the reference point r^*. The current population P is updated by invoking the UPDATESOLUTIONS module. The Archive is also updated by y according to the crowding distance. The evaluation counter is updated and checked. At the end of each learning period, the *Pool* is adapted by calculating the probability p_k for each strategy k according to Equ. 7.13. At the end of the evolution, the archive is returned.

In the UPDATESOLUTIONS module explained in Alg. 7.3, the offspring y updates the population P as follows: a random index j is selected from the updating range M_i. Then, the current solution of the jth subproblem x^j is updated only if y achieves better scalar fitness according to Equ. 2.19. In this case, the success of the selected strategy Suc_k is increased. And the age a_j is reset. Also, the

personal best x_{pb}^{j} is updated by the same manner. Finally the selected index j is eliminated from M_i. If the current solution x_j is not updated, its age a_j is increased. This process continues until updating t solution or M_i becomes empty.

7.4 Experimental Design

In this chapter, HESSA is verified using some of the state of the art MOEAs as: MOEA/D_1 [Zhang & Li 2007], MOEA/D_2 [Li & Zhang 2009] and dMOPSO [Martínez & Coello 2011]. As well known, MOEA/D_1 applies SBX-crossover and polynomial mutation (SBXPM) as a search strategy, whereas MOEA/D_2 adopts a search strategy consists of differential evolution followed by polynomial mutation (DEXPM). In addition, dMOPSO is really based on PSO strategy combined with the MOEA/D framework. Also, a set of standard test problems which cover MOPs with different PFs' characteristics as convexity, concavity, disconnections and multifrontality is adopted. The test problems contain bi-objectives test MOPs including Fonseca, Kursawe [Coello *et al.* 2006], ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 proposed in [Zitzler *et al.* 2000]. They also contain three-objectives MOPs such as DTLZ2, DTLZ4, DTLZ6 and DTLZ7 proposed in [Deb *et al.* 2005a]. The mathematical formulation and the characteristics of both bi-objective and three-objective test problems used in this study are shown in tables 7.2 and 7.3 respectively. According to these tables, 30 decision variables are used for ZDT1, ZDT2 and ZDT3, whereas ZDT4 and ZDT6 are tested by 10 decision variables. In DTLZ2, DTLZ4 and DTLZ6, 12 decision variables are used, whereas DTLZ7 is tested by 22 decision variables. All experiments are performed on a PC with Intel Core i5-2400 CPU, 3.1 GHz and 4 GB of RAM.

7.4.1 Parameter settings

For each algorithm, the population size N and the maximum evaluations $MaxEvals$ are set to 100, 10000 for bi-objective problems and 300, 30000 for three-objective test problems respectively. The archive size and the learning period LP are set to N, 1000 respectively. In dMOPSO and HESSA, the inertia weight w and coefficients c_1, c_2 used in PSO are uniformly generated as $U(0.1, 0.5)$ for w and $U(1.2, 2)$ for c_1 and c_2 as used in [Martínez & Coello 2011]. For HESSA, the guided mutation parameter μ is set to 0.03 and the mutation rate P_m is set to $1/n$. In MPC crossover, β is set to $N(0.7, 0.1)$ as used in [Elsayed *et al.* 2011]. The other common parameters are depicted in table 7.4. Finally, the statistical analysis is applied on 30 independent runs for each test MOP.

Algorithm 7.1 HESSA$(N, T, t, LP, \delta, p_c, p_m, \eta_c, \eta_m, CR, F, T_a)$

Inputs:
- N: *Population size or no. of sub-problems*
- T: *Minimum neighborhood size*
- t: *Maximum replaced solutions*
- LP: *Learning period*
- δ: *probability of selecting parents from neighborhood*
- p_c, p_m: *Crossover and mutation probabilities*
- η_c, η_m: *Distribution indexes for crossover and mutation*
- CR, F: *Crossover rate & scaling factor for differential evolution*
- T_a: *Maximum allowable age*

Begin:
$W_v \leftarrow \{\Lambda^1, ..., \Lambda^N\}$; ▷ *initialize a set of N evenly distributed weight vectors*
for $i \leftarrow 1$ to N **do**: ▷ *Construct Neighborhood structures*
  $B_i \leftarrow [i1, ..., iT]$; ▷ *where $\Lambda^{i1}, ..., \Lambda^{iT}$ are T closest to Λ^i in terms of Euclidean distance*
end for
$Pool \leftarrow$ ConstructPool(sbxpm,dexpm,mpcpm,gm,pso); ▷ *5 strategies*
$Eval \leftarrow 0$; ▷ *initialize Evaluation counter*
$Arch \leftarrow \varnothing$; ▷ *initialize with empty archive*
for $i \leftarrow 1$ to N **do**: ▷ *Initialization phase*
  $x^i_j \leftarrow U(a_j, b_j), \forall j = 1, ..., n.$ ▷ *get a uniform random $x_j \in [a_j, b_j]$*
  $x^i \leftarrow$ Evaluate(x^i); ▷ *Evaluate x^i*
  $r^* \leftarrow$ UpdateReferencePoint(x^i); ▷ *Update reference point r^**
  $x^i_{pb} \leftarrow x^i$; ▷ *Initialize personal best*
  $v^i \leftarrow 0$; $a_i \leftarrow 0$ ▷ *Initialize velocities & ages*
  $P \leftarrow$ AddSubProblem$(x^i, \Lambda^i, v^i, x^i_{pb}, a_i)$; ▷ *add the ith subproblem*
  $Arch \leftarrow$ UpdateArchive(x^i); ▷ *update Arch*
  $Eval \leftarrow Eval + 1$; ▷ *update Eval*
end for
while $(Eval < MaxEvals)$ **do**: ▷ *Main Loop*
  for $i \leftarrow 1$ to N **do**: ▷ *determine the mating/updating rang M_i*
    $M_i \leftarrow \begin{cases} B_i & \text{if}(rnd \in [0,1] < \delta) \\ 1, ..., N & \text{otherwise} \end{cases}$
    $x^a, x^b, x^c \leftarrow$ SelectParents(M_i, i); ▷ *Where: $x^i \neq x^a \neq x^b \neq x^c$*
    $x^i_{gb} \leftarrow$ SelectGlobalBest$(Arch)$; ▷ *randomly select the global guide*
    $S_k \leftarrow$ SelectStrategy$(Pool)$; ▷ *select a strategy S_k from Pool*
    $calls_k \leftarrow calls_k + 1$; ▷ *update # of calls for strategy S_k*
    $y \leftarrow$ Reproduction$(S_k, x^i, x^a, x^b, x^c, v^i, x^i_{pb}, x^i_{gb}, a_i)$; ▷ *Apply search strategy*
    $y \leftarrow$ Evaluate(y); ▷ *Evaluate y*
    $r^* \leftarrow$ UpdateReferencePoint(y); ▷ *Update reference point r^**
    $P \leftarrow$ UpdateSolutions(y, t, M_i, P, S_k, r^*);
    $Arch \leftarrow$ UpdateArchive(y, S_k); ▷ *crowding distance*
    $calls_{tot} \leftarrow calls_{tot} + 1$; ▷ *update total calls*
    $Eval \leftarrow Eval + 1$; ▷ *update Eval*
    if $(calls_{tot} \% LP = 0)$ **then**: ▷ *The End of learning period*
      $Pool \leftarrow$ AdaptPool$(Pool)$; ▷ *recalculate p_k for each strategy k*
    end if ▷ *The End of Pool Adaptation*
  end for ▷ *The End of Generation*
end while
return $Arch$;
End

Algorithm 7.2 REPRODUCTION$(S_k, x^i, x^a, x^b, x^c, v^i, x^i_{pb}, x^i_{gb}, a_i)$

Inputs:
S_k: *the selected strategy*
x^i: *the current solution*
x^a, x^b, x^c: *three selected parents*
v^i, a_i: *velocity and age*
x^i_{pb}, x^i_{gb}: *the personal best and the guide*

Begin:
if ($S_k =$ "SBXPM") **then**: ▷ *SBX crossover then Polynomial mutation*
  $y \leftarrow$CROSSOVER(x^a, x^b);
  $y \leftarrow$POLYMUTATION(y);
else if ($S_k =$ "DEXPM") **then**: ▷ *Differential Evolution & Polynomial mutation*
  $y \leftarrow$DIFFEVOLUTION$(x^i, x^a, x^b, x^c, CR, F)$;
  $y \leftarrow$POLYMUTATION(y);
else if ($S_k =$ "MPCPM") **then**: ▷ *Multi-Parent crossover & Polynomial mutation*
  $y \leftarrow$MPCROSSOVER$(x^a, x^b, x^c, x^i_{gb})$;
  $y \leftarrow$POLYMUTATION(y);
else if ($S_k =$ "GM") **then**: ▷ *Apply Guided Mutation or reset y*
  $y \leftarrow \begin{cases} \text{GUIDEDMUTATION}(x^i, x^i_{gb}); & \text{if: } a_i < T_a \\ N(\frac{1}{2}[x^i_{gb} - x^i_{pb}], |x^i_{gb} - x^i_{pb}|) & \text{otherwise} \end{cases}$
else if ($S_k =$ "PSO") **then**: ▷ *Apply Particle Swarm or reset y*
  $y \leftarrow \begin{cases} \text{PSO}(x^i, x^i_{pb}, x^i_{gb}, v^i, a_i); & \text{if: } a_i < T_a \\ N(\frac{1}{2}[x^i_{gb} - x^i_{pb}], |x^i_{gb} - x^i_{pb}|) & \text{otherwise} \end{cases}$
end if ▷ *The End of Reproduction*
return y; ▷ *return the new generated offspring*
End

Algorithm 7.3 UPDATESOLUTIONS(y, t, M_i, P, S_k, r^*)

Inputs:
y: *the new solution*
t: *the Maximum replaced solutions*
P: *the current population*
M_i: *the updating rang of the i_{th} sub-problem*
S_k: *the selected search strategy*
r^*: *the reference point*

Begin:
$c \leftarrow 0$; ▷ *initialize counter*
while ($c < t \wedge M_i \neq \varnothing$) **do**: ▷ *update Loop*
  $j \leftarrow$SELECTRANDOMINDEX(M_i); ▷ *randomly select index j*
  if ($F^{Tc}(y, \Lambda^j, r^*) \leq F^{Tc}(x^j, \Lambda^j, r^*)$) **then**: ▷ *in case of success*
    $x^j \leftarrow y$; ▷ *update j^{th} subproblem*
    $a_j \leftarrow 0$; $c \leftarrow c + 1$; ▷ *reset age, update counter*
    $suc_k \leftarrow suc_k + 1$; ▷ *update # of success for strategy S_k*
    if ($F^{Tc}(x^j, \Lambda^j, r^*) \leq F^{Tc}(x^j_{pb}, \Lambda^j, r^*)$) **then**:
      $x^j_{pb} \leftarrow x^j$; ▷ *update the personal best*
    end if
    $M_i \leftarrow$REMOVEINDEX(M_i, j); ▷ *exclude j from M_i*
  else: $a_j \leftarrow a_j + 1$; ▷ *in case of failure, update age a_j*
  end if
end while
return P; ▷ *return the updated population P*
End

Table 7.2: List of the bi-objective test problems

Problem	Mathematical formulation	Comments
Fonseca	$f_1(x) = 1 - exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2)$ $f_2(x) = 1 - exp(-\sum_{i+1}^{n}(x_i + \frac{1}{\sqrt{n}})^2)$ $-4 \leq x_i \leq 4, i = 1, ..., n., n = 3$	Non-Convex
Kursawe	$f_1(x) = \sum_{i=1}^{n-1}(-10e^{(0.2)\times\sqrt{x_i^2+x_{i+1}^2}})$ $f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5sin(x_i)^3)$ $-5 \leq x_i \leq 5, i = 1, ..., n., n = 3$	Disconnected
ZDT1	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$ $0 \leq x_i \leq 1, i = 1, ..., n. n = 30$	Convex
ZDT2	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$ $0 \leq x_i \leq 1, i = 1, ..., n. n = 30$	Non-Convex
ZDT3	$f_1(x) = x_1$ $f_2(x) = g(x)h(x)$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$ $h(x) = [1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}sin(10\pi x_1)]$ $0 \leq x_i \leq 1, i = 1, ..., n. n = 30$	Convex, Disconnected
ZDT4	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + 10(n-1) + h(x)$ $h(x) = \sum_{i=2}^{n}[x_i^2 - 10cos(4\pi x_i)]$ $x_1 \in [0, 1], -5 \leq x_i \leq 5, i = 2, ..., n. n = 10$	Non-Convex, Multifrontal
ZDT6	$f_1(x) = 1 - e^{-4x_1}sin^6(6\pi x_1)$ $f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$ $g(x) = 1 + 9[(\sum_{i=2}^{n} x_i^2)/(n-1)]^{0.25}$ $0 \leq x_i \leq 1, i = 1, ..., n. n = 30$	Non-Convex, Many-to-one, Non-uniform

7.4.2 Assessment Metrics

In order to assess the quality of the solutions obtained by each algorithm, some of the quality assessment indicators are used in these experiments including binary and unary indicators. These indicators include the set coverage I_C indicator which is used to compare each pair of MOEAs as discussed in section 2.6.2.1. They also include indicators to assess each MOEA independently such as referenced hypervolume (I_{Rhyp}), generational distance (I_{GD}), inverted generational distance (I_{IGD}) and unary additive epsilon (I_{ε_+}) indicators which are presented

Table 7.3: List of three-Objective Test Problems

Problem	Mathematical formulation	Comments
DTLZ2	$f_1(x) = cos(x_1\pi/2)cos(x_2\pi/2)(1+g(x))$ $f_2(x) = cos(x_1\pi/2)sin(x_2\pi/2)(1+g(x))$ $f_3(x) = sin(x_1\pi/2)(1+g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0.5)^2$ $0 \leq x_i \leq 1, \forall i = 1, ..., n, n = 12$	Non-Convex, Scalable, Multimodal
DTLZ4	$f_1(x) = cos(x_1^\alpha\pi/2)cos(x_2^\alpha\pi/2)(1+g(x))$ $f_2(x) = cos(x_1^\alpha\pi/2)sin(x_2^\alpha\pi/2)(1+g(x))$ $f_3(x) = sin(x_1^\alpha\pi/2)(1+g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0.5)^2$ $0 \leq x_i \leq 1, \forall i = 1, ..., n, n = 12$	Non-Convex, Separable, Unimodal
DTLZ6	$f_1(x) = x_1$ $f_2(x) = x_2$ $f_3(x) = (1+g(x))h(f_1, f_2, f_3, g)$ $g(x) = \frac{9}{n-2}\sum_{i=3}^{n} x_i$ $h(f_1, f_2, f_3, g) = 3 - \sum_{i=1}^{3}\left(\frac{f_i}{1+g}(1+sin(3\pi f_i))\right)$ $0 \leq x_i \leq 1, \forall i = 1, ..., n, n = 22$	Unimodal, Bias, Many-to-one-mapping
DTLZ7	$f_1(x) = \frac{1}{10}\sum_{i=1}^{10} x_i$ $f_2(x) = \frac{1}{10}\sum_{i=11}^{20} x_i$ $f_3(x) = \frac{1}{10}\sum_{i=21}^{30} x_i$ $g_1(x) = f_3(x) + 4f_1(x) - 1 \geq 0$ $g_2(x) = f_3(x) + 4f_2(x) - 1 \geq 0$ $g_3(x) = 2f_3(x) + f_1(x) + f_2(x) - 1 \geq 0$ $0 \leq x_i \leq 1, \forall i = 1, ..., n, n = 30$	Disconnected

Table 7.4: Set of common parameters

Parameters	Algorithms			
	MOEAD$_1$	**MOEAD$_2$**	**dMOPSO**	**HESSA**
Neighborhood size: T	30	30	-	30
Max. Replaced Solutions: t	-	2	-	2
Parents selection Prob.: δ	-	0.9	-	0.9
Crossover rate: p_c	1	-	-	1
Crossover index: η_c	20	-	-	20
Mutation rate: p_m	$1/n$	$1/n$	-	$1/n$
Mutation index: η_m	20	20	-	20
DE parameters: CR, F	-	1,0.5	-	1,0.5
Age threshold: T_a	-	-	2	2
PBI penalty value: θ	-	-	5	-

in sections 2.6.2.2, 2.6.2.3, 2.6.2.3 and 2.6.2.6 respectively. In these experiments, the True Pareto front for each test problem is used as the reference set P^*. Also, the reference point r^* used in the referenced hypervolume (I_{Rhyp}) calculations is set as the nadir vectors taken from the reference set P^* of each test problems.

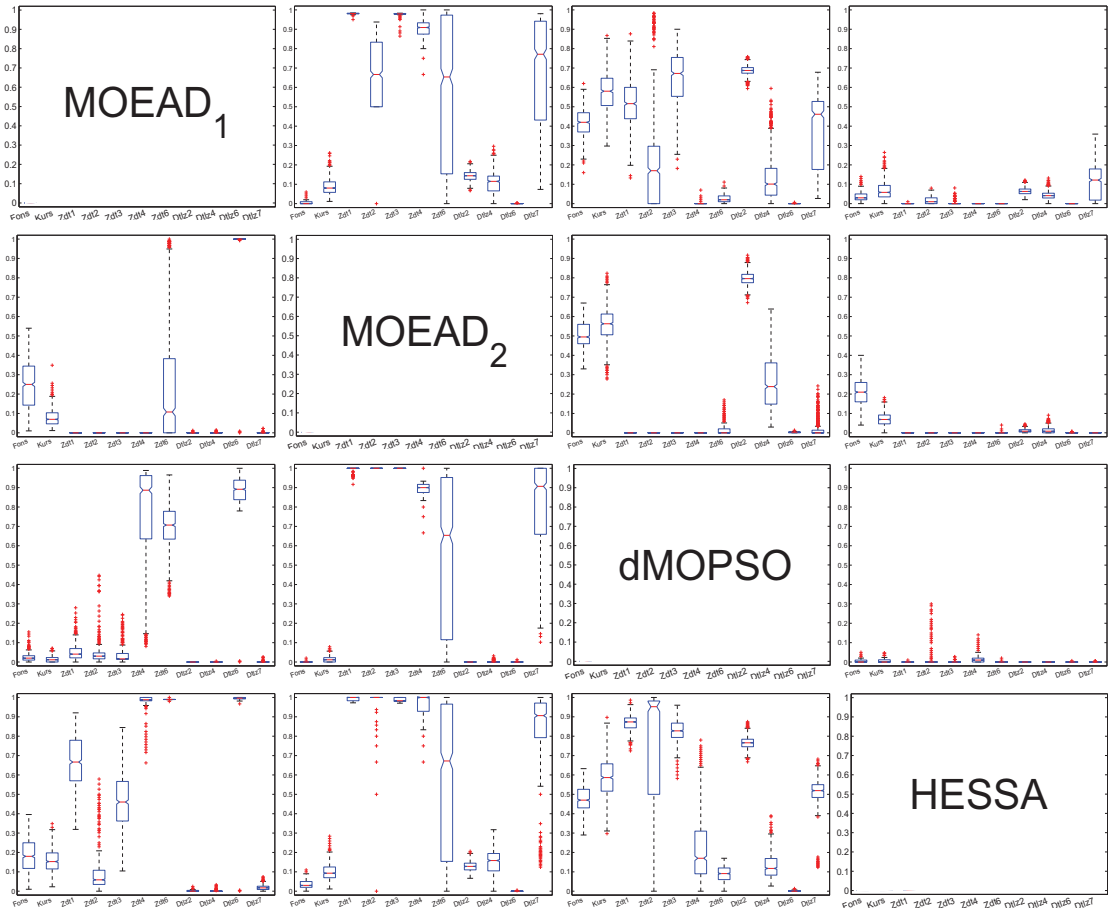


Figure 7.4: Results of the coverage I_C indicator

7.5 Experimental Results

In this section, the different simulation results are shown in details. Firstly, the results of the set coverage I_C indicator are shown in Fig. 7.4 which contains a chart (with scale 0 at the bottom and 1 at the top) for each ordered pair of the compared MOEAs. Each chart consists of Eleven box plots representing the distribution of I_C values. The box plots (from left to right) correspond to test problems Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, DTLZ2, DTLZ4, DTLZ6 and DTLZ7 respectively. A chart located in the raw of algorithm A and the column of algorithm B presents the values of coverage of approximations generated by algorithm B by approximations generated by algorithm A.

Additionally, the median values of I_C indicator are listed in table B.16. According to these results, it is clear that HESSA generally performs better than the other algorithms in all test MOPs except DTLZ2 and DTLZ4 with respect to MOEAD_1, and slightly have the same performance with dMOPSO in DTLZ6.

The results listed in table 7.5 express the average and the standard deviation of the referenced hypervolume I_{RH} indicator. The detailed statistical results of I_{RH} are also presented in table B.17. These results indicate that HESSA is able to achieve better performance in all bi-objective test MOPs except Fonseca, and have the second best performance in three-objective test problems except DTLZ4 in which it achieves the best performance.

In table 7.6, the average and the standard deviation of the I_{GD} indicator are shown. Additionally, table B.18 shows the detailed statistical analysis for these values. It is clear that these results confirm the previous results of the I_{RH} indicator in most test problems. For DTLZ4, HESSA achieves the second best performance after MOEAD_1, whereas in DTLZ7, HESSA achieves the best performance followed by MOEAD_1.

For the I_{IGD} indicator, the average and the standard deviation are shown in table 7.7. Also, table B.19 lists more details for the I_{IGD} results. According to these results, HESSA outperforms the other algorithms in all bi-objective test problems except Fonseca. It has also the second best performance in most three-objective test problems except DTLZ4 in which HESSA achieves the best performance. These results typically confirm the results of the I_{RH} indicator.

Finally, table 7.8 shows the average and the standard deviation of the epsilon $I_{\varepsilon+}$ indicator. The detailed statistical analysis for $I_{\varepsilon+}$ is shown in table B.20. These results are nearly the same as those obtained by the previous indicators. HESSA achieves the best performance in all test problems except Fonseca, DTLZ2 and DTLZ4, in which MOEAD_2 has the best performance. Generally from the above results, HESSA achieves the best performance in most cases or at least the second best performance.

Concerning the search strategy adaptation in HESSA, Fig. 7.5 and Fig. 7.6 depict the adaptation of different search strategies for bi-objective and three-objective test problems. Here, The runs with the minimum I_{IGD} values are considered. For each test problem, it is clear that all search strategies begin with the same probability to be selected to launch the search process. The learning process for each search strategy proceeds during the evolutions. At the end of each learning period, the performance of each search strategy is evaluated to adapt its probability of selection. This reflects the ability of HESSA to control the search process by launching the suitable search strategy at the appropriate time for each test instances.

Table 7.5: Results of I_{Rhyp} indicator (Average,σ)

MOPs	MOEAD$_1$	MOEAD$_2$	dMOPSO	HESSA
Fonse	$5.03e-3_{6.5e-4}$	$3.64e-3_{4.3e-5}$	$1.19e-2_{1.4e-3}$	$4.08e-3_{1.1e-4}$
Kursa	$2.94e-1_{1.8e-2}$	$3.83e-1_{3.7e-2}$	$1.49e+0_{2.1e-1}$	$2.77e-1_{1.3e-2}$
ZDT1	$2.92e-2_{1.5e-2}$	$4.50e-1_{8.9e-2}$	$2.36e-2_{2.5e-3}$	$5.50e-3_{2.0e-4}$
ZDT2	$1.87e-1_{8.1e-2}$	$3.33e-1_{0.0e+0}$	$1.23e-1_{1.4e-1}$	$5.48e-3_{4.7e-4}$
ZDT3	$2.28e-2_{2.3e-2}$	$6.22e-1_{6.3e-2}$	$3.09e-2_{6.2e-3}$	$5.65e-3_{2.9e-4}$
ZDT4	$1.05e-1_{7.5e-2}$	$6.66e-1_{1.1e-8}$	$1.01e-1_{1.2e-1}$	$5.53e-3_{4.3e-4}$
ZDT6	$1.54e-2_{2.6e-3}$	$1.32e-1_{8.8e-2}$	$8.76e-3_{3.6e-3}$	$2.16e-4_{1.2e-5}$
DTLZ2	$4.61e-2_{1.3e-3}$	$4.84e-2_{1.1e-3}$	$1.21e-1_{6.7e-3}$	$4.64e-2_{1.1e-3}$
DTLZ4	$1.40e-1_{1.3e-1}$	$2.05e-2_{3.0e-2}$	$5.81e-2_{1.2e-2}$	$4.01e-4_{1.1e-3}$
DTLZ6	$1.11e-2_{6.9e-3}$	$2.67e-4_{7.6e-6}$	$7.71e-4_{1.6e-4}$	$3.11e-4_{3.4e-6}$
DTLZ7	$1.71e-1_{2.1e-2}$	$5.36e-1_{1.2e-1}$	$1.67e-1_{2.0e-2}$	$1.69e-1_{5.2e-3}$

Table 7.6: Results of I_{GD} indicator (Average,σ)

MOPs	MOEAD$_1$	MOEAD$_2$	dMOPSO	HESSA
Fonse	$1.56e-3_{2.4e-4}$	$9.96e-4_{3.3e-5}$	$4.49e-3_{5.1e-4}$	$1.14e-3_{5.8e-5}$
Kursa	$8.56e-3_{1.2e-3}$	$1.09e-2_{1.6e-3}$	$4.89e-2_{8.7e-3}$	$7.46e-3_{6.6e-4}$
ZDT1	$5.15e-3_{1.4e-3}$	$3.67e-1_{9.7e-2}$	$1.29e-2_{1.8e-3}$	$8.30e-4_{1.2e-4}$
ZDT2	$1.06e-3_{1.4e-3}$	$4.58e-1_{1.6e-1}$	$1.35e-2_{1.1e-2}$	$9.12e-4_{2.8e-4}$
ZDT3	$5.83e-3_{6.5e-3}$	$5.06e-1_{1.0e-1}$	$8.87e-3_{1.6e-3}$	$2.70e-3_{1.6e-4}$
ZDT4	$7.15e-2_{8.1e-2}$	$1.75e+1_{6.4e+0}$	$1.87e-3_{1.3e-3}$	$8.38e-4_{2.4e-4}$
ZDT6	$1.35e-2_{2.1e-3}$	$2.15e-1_{1.9e-1}$	$5.45e-3_{9.0e-3}$	$2.62e-3_{3.7e-5}$
DTLZ2	$5.85e-3_{1.1e-4}$	$7.85e-3_{2.4e-4}$	$6.85e-2_{5.0e-3}$	$6.33e-3_{1.7e-4}$
DTLZ4	$2.51e-2_{1.1e-2}$	$3.60e-2_{3.0e-3}$	$4.58e-2_{5.2e-3}$	$3.51e-2_{1.4e-3}$
DTLZ6	$4.19e-2_{2.8e-2}$	$3.72e-3_{8.4e-5}$	$4.42e-3_{2.3e-4}$	$3.85e-3_{6.8e-5}$
DTLZ7	$2.25e-2_{1.7e-3}$	$1.69e-1_{7.4e-2}$	$5.20e-2_{6.2e-3}$	$2.19e-2_{4.0e-4}$

Table 7.7: Results of I_{IGD} indicator (Average,σ)

MOPs	MOEAD$_1$	MOEAD$_2$	dMOPSO	HESSA
Fonse	$4.21e-3_{4.5e-4}$	$3.57e-3_{2.2e-5}$	$1.63e-2_{5.3e-3}$	$3.70e-3_{5.9e-5}$
Kursa	$4.24e-2_{1.2e-3}$	$4.42e-2_{1.2e-3}$	$1.06e-1_{2.1e-2}$	$4.20e-2_{5.2e-4}$
ZDT1	$3.87e-2_{3.2e-2}$	$3.90e-1_{9.6e-2}$	$1.48e-2_{1.5e-3}$	$4.05e-3_{7.1e-5}$
ZDT2	$2.46e-1_{1.2e-1}$	$8.98e-1_{1.5e-1}$	$1.95e-1_{2.7e-1}$	$4.00e-3_{1.2e-4}$
ZDT3	$2.67e-2_{2.7e-2}$	$4.66e-1_{7.6e-2}$	$1.75e-2_{2.4e-3}$	$1.06e-2_{1.1e-4}$
ZDT4	$1.05e-1_{6.8e-2}$	$6.48e+0_{2.8e+0}$	$1.60e-1_{1.8e-1}$	$4.11e-3_{1.3e-4}$
ZDT6	$1.93e-2_{3.4e-3}$	$1.63e-1_{2.2e-1}$	$3.63e-3_{1.1e-3}$	$1.89e-3_{1.6e-5}$
DTLZ2	$3.72e-2_{2.0e-4}$	$3.81e-2_{3.4e-4}$	$7.33e-2_{4.6e-3}$	$3.73e-2_{2.3e-4}$
DTLZ4	$1.69e-1_{1.4e-1}$	$2.99e-2_{5.2e-3}$	$4.41e-2_{6.5e-3}$	$2.98e-2_{2.0e-3}$
DTLZ6	$3.82e-2_{2.6e-2}$	$4.39e-3_{3.2e-5}$	$7.72e-3_{7.2e-4}$	$4.52e-3_{1.5e-5}$
DTLZ7	$2.24e-1_{1.5e-1}$	$3.76e-1_{1.8e-1}$	$8.94e-2_{5.1e-2}$	$1.15e-1_{3.5e-3}$

Table 7.8: Results of I_{ε_+} indicator (Average,σ)

MOPs	MOEAD$_1$	MOEAD$_2$	dMOPSO	HESSA
Fonse	$9.09e-0_{2.0e-0}$	$6.47e-3_{1.4e-4}$	$7.45e-2_{3.3e-2}$	$7.07e-3_{3.7e-4}$
Kursa	$8.79e-2_{1.6e-2}$	$1.09e-1_{1.4e-2}$	$4.57e-1_{1.9e-1}$	$8.24e-2_{9.0e-3}$
ZDT1	$1.10e-1_{7.0e-2}$	$4.88e-1_{1.2e-1}$	$2.88e-2_{3.7e-3}$	$8.25e-3_{4.1e-4}$
ZDT2	$7.21e-1_{2.6e-1}$	$1.42e+0_{1.7e-1}$	$3.26e-1_{4.4e-1}$	$7.44e-3_{4.5e-4}$
ZDT3	$1.27e-1_{1.8e-1}$	$8.66e-1_{1.7e-1}$	$4.64e-2_{1.2e-2}$	$1.51e-2_{4.2e-4}$
ZDT4	$2.13e-1_{1.0e-1}$	$6.80e+0_{2.8e+0}$	$2.53e-1_{2.3e-1}$	$8.91e-3_{8.4e-4}$
ZDT6	$2.85e-2_{4.9e-3}$	$3.09e-1_{2.4e-1}$	$2.99e-2_{8.6e-3}$	$5.03e-3_{2.3e-4}$
DTLZ2	$8.76e-2_{4.2e-3}$	$7.83e-2_{8.1e-3}$	$1.07e-1_{6.1e-3}$	$8.44e-2_{4.8e-3}$
DTLZ4	$4.33e-1_{3.3e-1}$	$8.24e-2_{5.7e-2}$	$1.38e-1_{1.8e-2}$	$6.39e-2_{1.0e-2}$
DTLZ6	$4.30e-2_{2.1e-2}$	$1.02e-2_{2.6e-4}$	$1.47e-2_{1.9e-3}$	$1.04e-2_{1.5e-4}$
DTLZ7	$6.16e-1_{6.3e-1}$	$9.65e-1_{6.0e-1}$	$2.49e-1_{2.1e-1}$	$1.68e-1_{3.9e-3}$

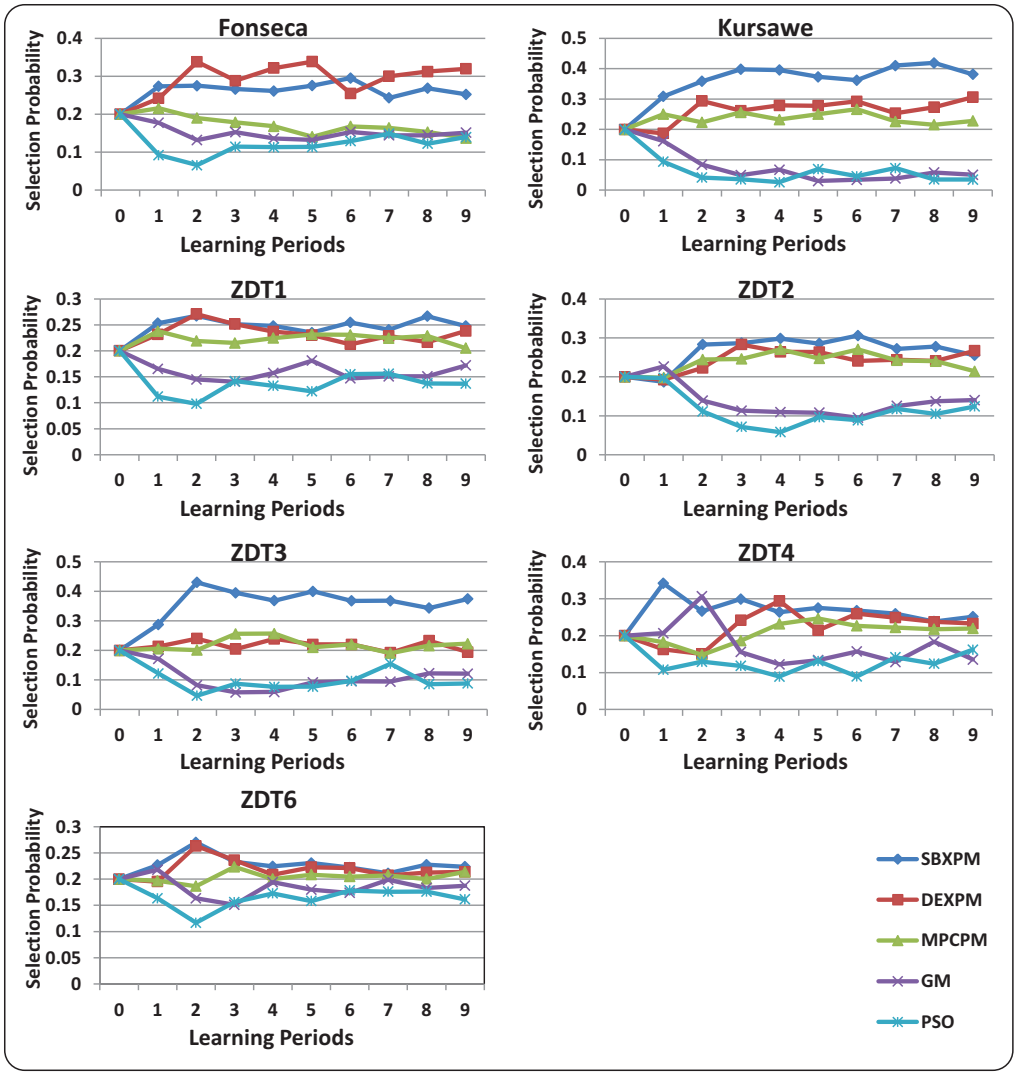


Figure 7.5: Search Strategy Adaptation for bi-objectives test problems

In Fig. 7.5, concerning Fonseca test problem, it is shown that DEXPM search strategy achieves the highest probability of selection followed by SBXPM, whereas MPCPM, GM and PSO achieve lower values for probability of selection during the learning periods. For Kursawe, SBXPM has the highest probability of selection followed by DEXPM then MPCPM, whereas GM and PSO has lower values for probability of selection during the different learning periods. For ZDT1, ZDT2 and ZDT4 test problems, the search strategies with the highest probability of selection are SBXPM, DEXPM and MPCPM interchangeably, where GM and PSO has lower values for selection probabilities. In ZDT3, DEXPM gains the highest probability of selection during whole the search process, whereas both SBXPM and MPCPM has the medium values of selection probabilities, then GM and PSO achieves the lower values of selection probabilities. In ZDT6, despite

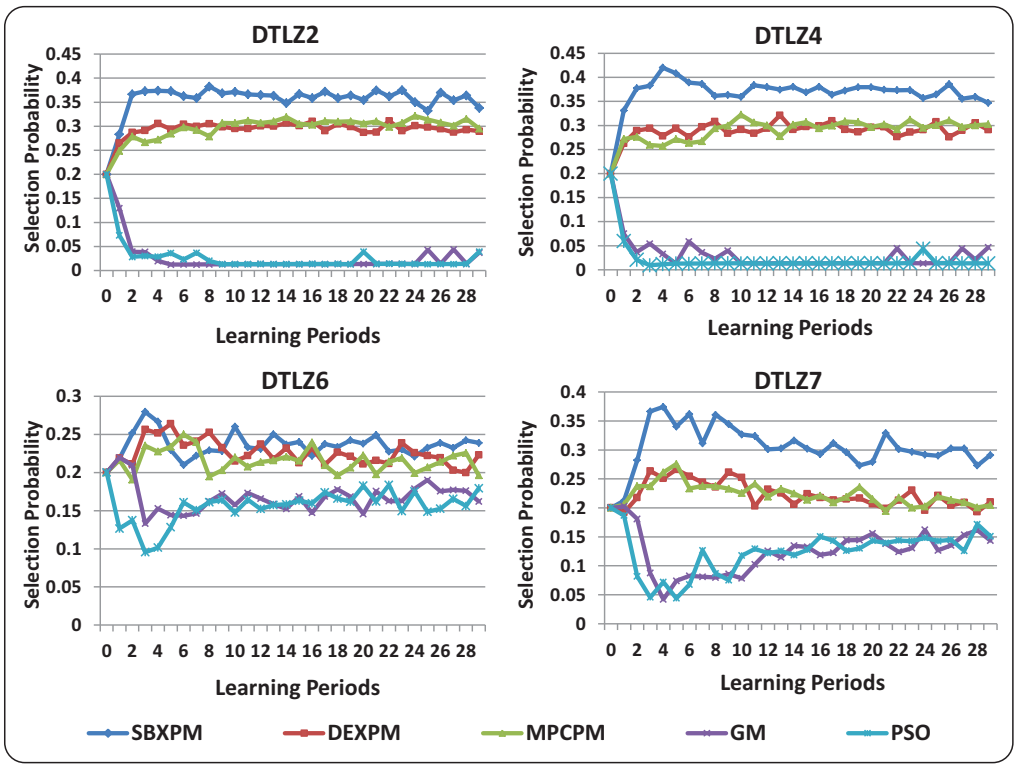


Figure 7.6: Search Strategy Adaptation for three-objectives test problems

all strategies has close values for probability of selection, it is clear that DEXPM an SBXPM performs better followed by MPCPM.

In Fig. 7.6, concerning DTLZ2, DTLZ4 and DTLZ7, SBXPM performs better since it has the highest probability of selection followed by DEXPM and MPCPM search strategies, whereas both GM and PSO achieve weak performance in these test instances. For DTLZ6, the strategies with the highest probability of selection are SBXPM,DEXPM and MPCPM interchangeably, whereas both GM and PSO has the lowest values of selection probabilities.

In general, it is obvious form both Fig. 7.5 and Fig. 7.6 that, SBXPM have the best performance followed by DEXPM and MPCPM for all test problems except Fonseca in which DEXPM has the best performance, whereas GM and PSO search strategies achieve weak performance.

Here, the scatter plots of the results achieved by the compared algorithms are considered. For bi-objective test problems, the scatter plots presented in Fig. 7.7, Fig. 7.8, Fig. 7.9, Fig. 7.10, Fig. 7.11, Fig. 7.12 and Fig. 7.13 contain the final Pareto fronts achieved by each algorithm for Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 test problems. Also, for tri-objective test problems,

Fig. 7.14, Fig. 7.15, Fig. 7.16 and Fig. 7.17 present the scatter plots which contain the final Pareto fronts achieved by each algorithm for DTLZ2, DTLZ4, DTLZ6 and DTLZ7 test problems respectively. In these plots, the runs that achieve the minimum I_{IGD} values are considered.

In both Fig. 7.7 and Fig . 7.8, the approximation sets achieved by the compared algorithms for Fonseca and Kursawe test problems are plotted respectively . It is clear that, HESSA, MOEAD_1 and MOEAD_2 achieves approximation sets of high quality solutions uniformly distributed along the True Pareto Front.

According to the scatter plots illustrated in Fig. 7.9, Fig. 7.10 and Fig. 7.11 in which the approximation sets achieved by the compared algorithms for ZDT1, ZDT2 and ZDT3 are plotted respectively. It is clear that HESSA achieves high quality approximation sets with high convergence and diversity to the true Pareto front fallowed by MOEAD_1 and dMOPSO.

Figure 7.12 visualizes the approximation sets achieved by each algorithm for ZDT4. These results indicate that HESSA achieves good approximation set in terms of convergence and diversity with respect to the other algorithms followed by dMOPSO and MOEAD_1.

Figure 7.13 plots the approximation sets obtained by each algorithm for ZDT6. These results indicate that HESSA has the best performance followed by MOEAD_2 then MOEAD_1 followed by dMOPSO.

Fig. 7.14 depict the approximation sets obtained by each algorithm for DTLZ2. The results indicate that each of HESSA, MOEAD_1 and MOEAD_2 perform better as they achieve high quality approximation sets in terms if convergence and diversity with respect to the true Pareto front. This reflects the search strategy adaptation plots for DTLZ2 in Fig. 7.6, where SPXPM and DEXPM have the highest probability for selection as the most suitable search strategies for this test problem. Regarding that MOEAD_1 and MOEAD_2 adopts SPXPM and DEXPM search strategies respectively. Moreover, we can conclude that PSO is not suitable for this problem. So, dMOPSO achieves weak performance with DTLZ2. Also, PSO has low values for probability of selection in HESSA.

Fig. 7.15 plots the approximation sets obtained by each algorithm for DTLZ4. The results declare that HESSA and MOEAD_1 has better performance with DTLZ4, whereas MOEAD_2 and dMOPSO achieve weak results. It is clear that both HESSA and MOEAD_1 obtained more solutions dispersed along the PF more than MOEAD_2 and dMOPSO. So, in Fig.7.6 with DTLZ4 it is obvious that SPXPM has the highest probability of selection which is a basic component in MOEAD_1 framework.

Fig. 7.16 depicts the approximation sets obtained by each algorithm for DTLZ6. It is clear that each of HESSA, MOEAD_1 and MOEAD_2 nearly achieve the

same results with DTLZ6 which is slight better than dMOPSO. We can see the reflection for those results in Fig.7.6 with problem DTLZ6 where each SPXPM and DEXPM have the higher and close probability of selection and PSO has slightly lower values for the probability of selection.

Fig. 7.17 depicts the approximation set obtained by each algorithm for DTLZ7. As can be seen, DTLZ7 is a multi-frontlty problem, it has a four front. $MOEAD_1$ can reach only one of them. $MOEAD_2$ is able to reach two of the four fronts. dMOPSO can reach the four parts of the front but it is obvious that the solutions obtained is far from the the true ones. In contrast, HESSA is able to reach the four front parts, whereas the solutions obtained is more close to the true ones.

In general, we can conclude that the most suitable strategy for the problem on hand usually has higher selection probability during the learning periods. This proves the ability of HESSA to launch the most suitable strategy at the appropriate time for the problem on hand.

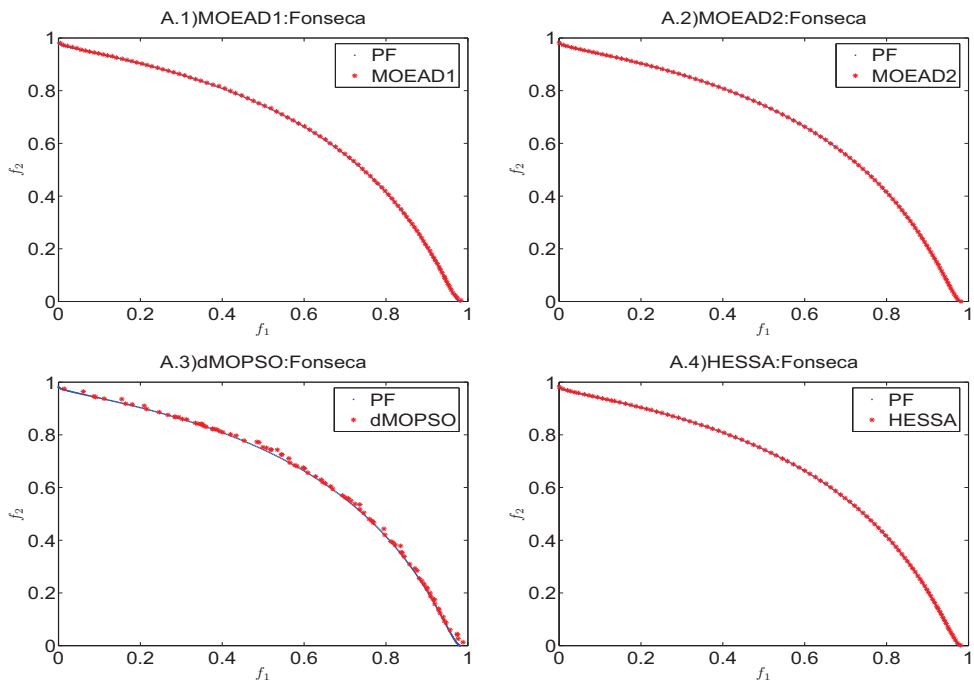


Figure 7.7: The Pareto fronts achieved for Fonseca test problems

7.6 Summary

In this chapter, a hybrid evolutionary approach with search strategy adaptation (HESSA) for handling multiobjective continuous problems is presented. In HESSA, the search process is controlled by the adaption of the search strategies used during the evolution process. HESSA is verified using a set of test

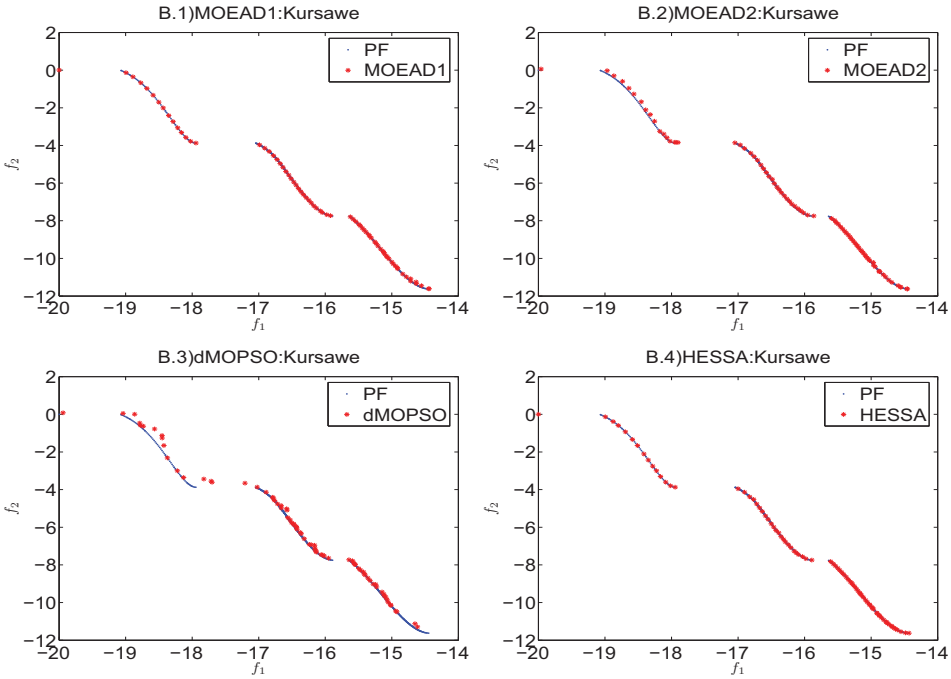


Figure 7.8: The Pareto fronts achieved for Kursawe test problems

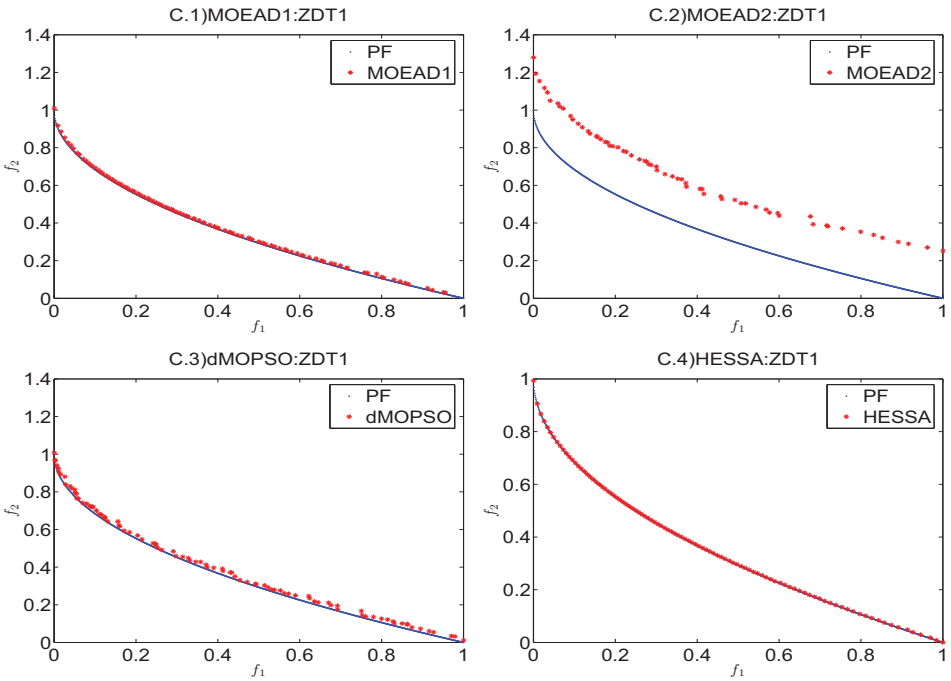


Figure 7.9: The Pareto fronts achieved for ZDT1 test problems

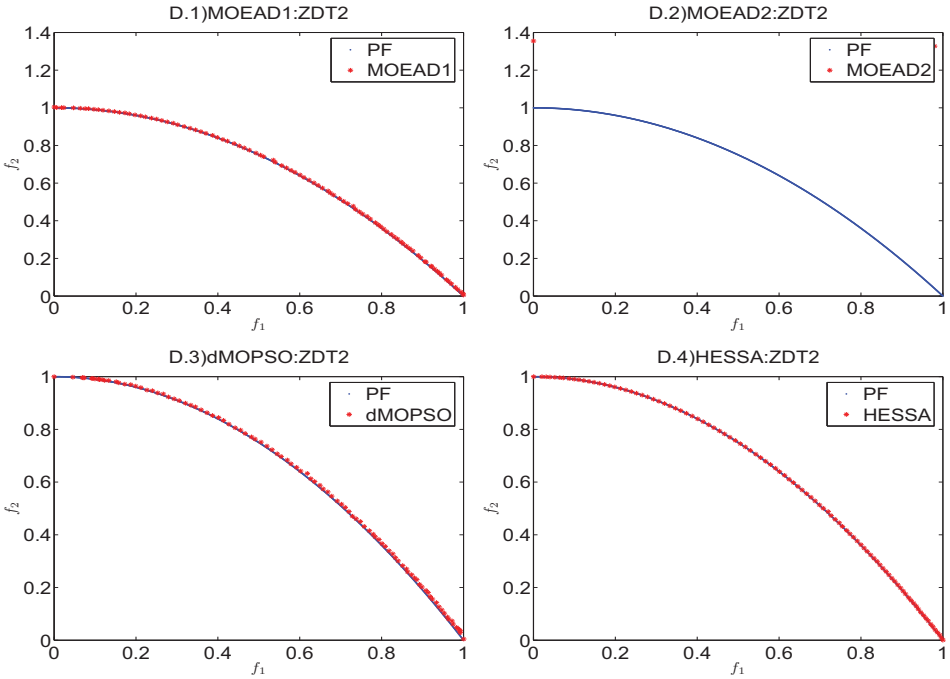


Figure 7.10: The Pareto fronts achieved for ZDT2 test problems

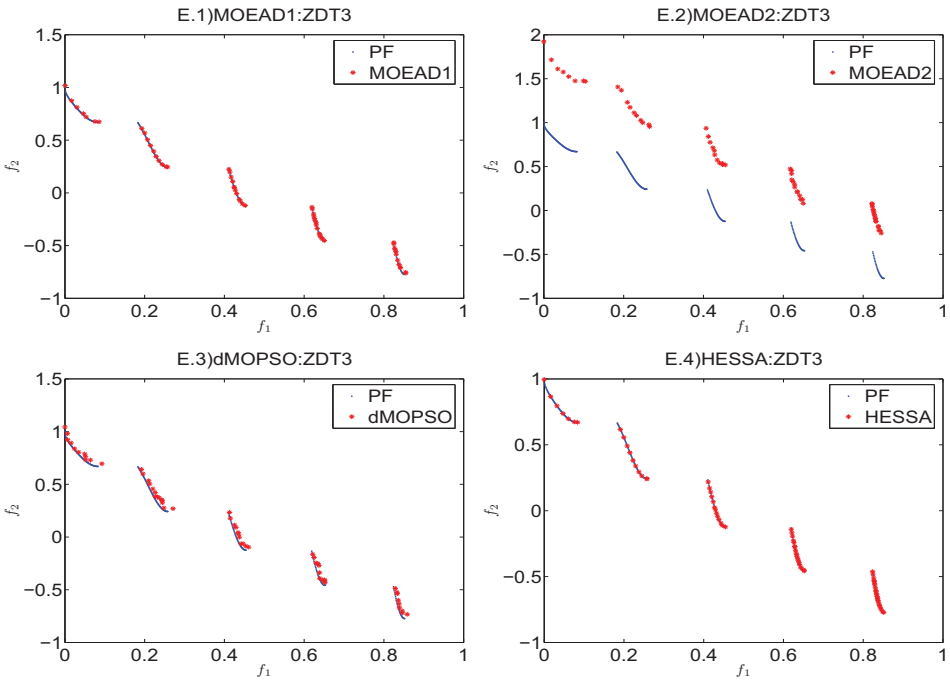


Figure 7.11: The Pareto fronts achieved for ZDT3 test problems

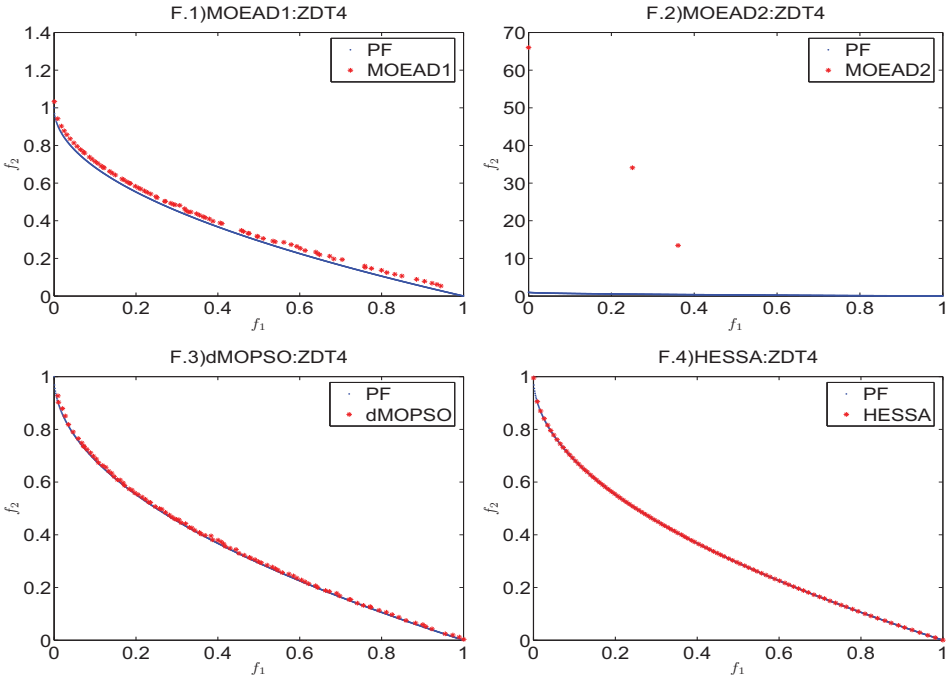


Figure 7.12: The Pareto fronts achieved for ZDT4 test problems

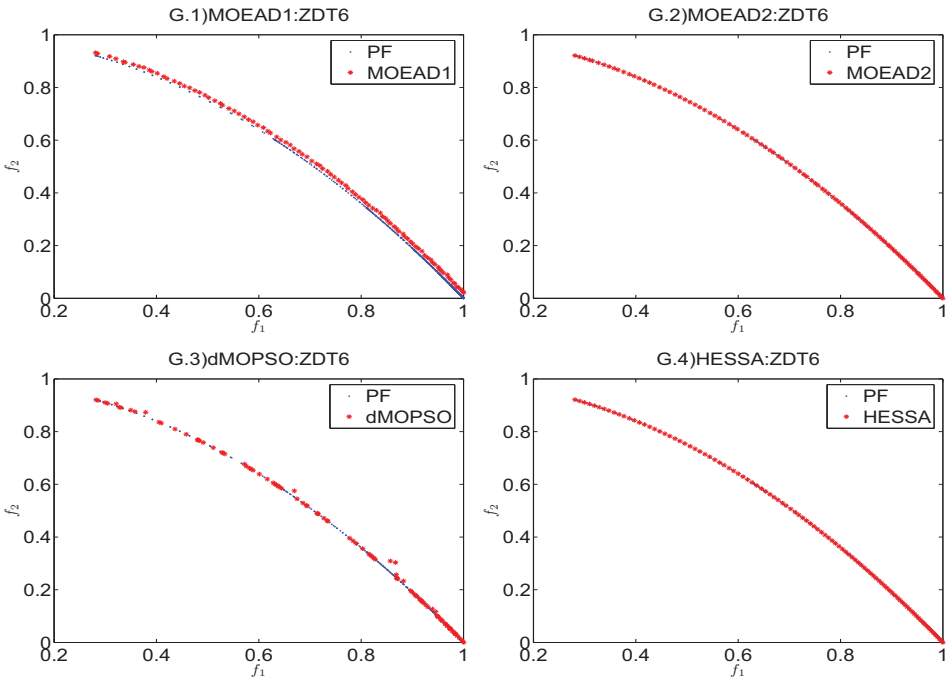


Figure 7.13: The Pareto fronts achieved for ZDT6 test problems

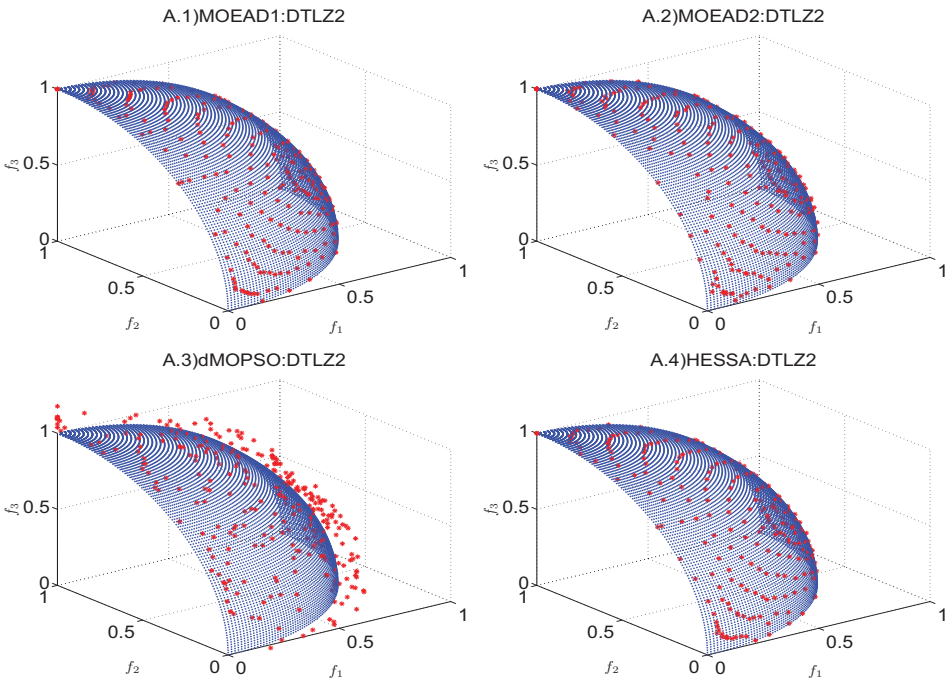


Figure 7.14: The Pareto fronts achieved for DTLZ2 test problems

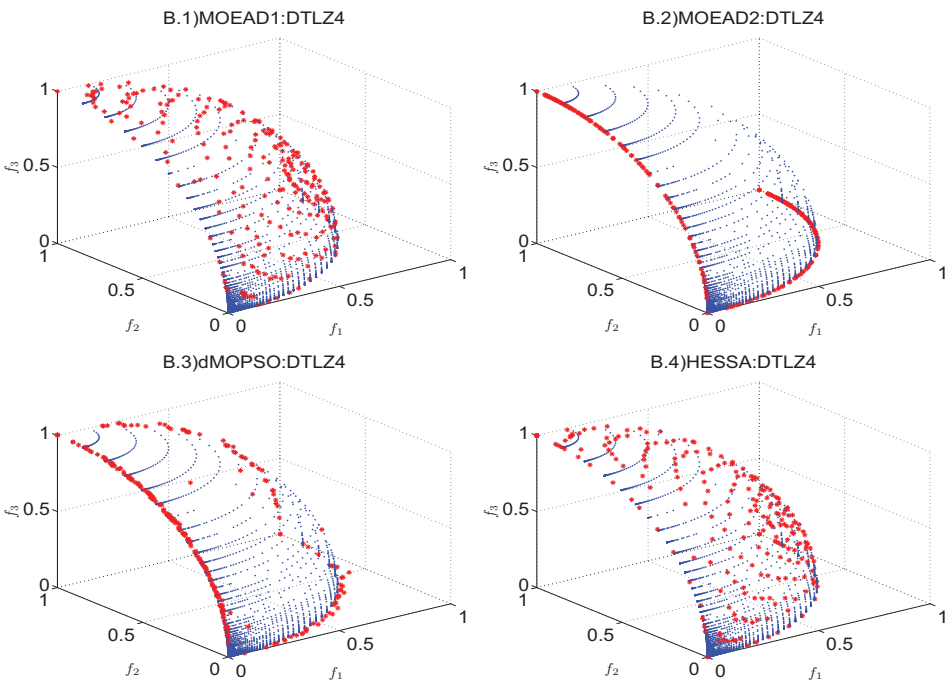


Figure 7.15: The Pareto fronts achieved for DTLZ4 test problems

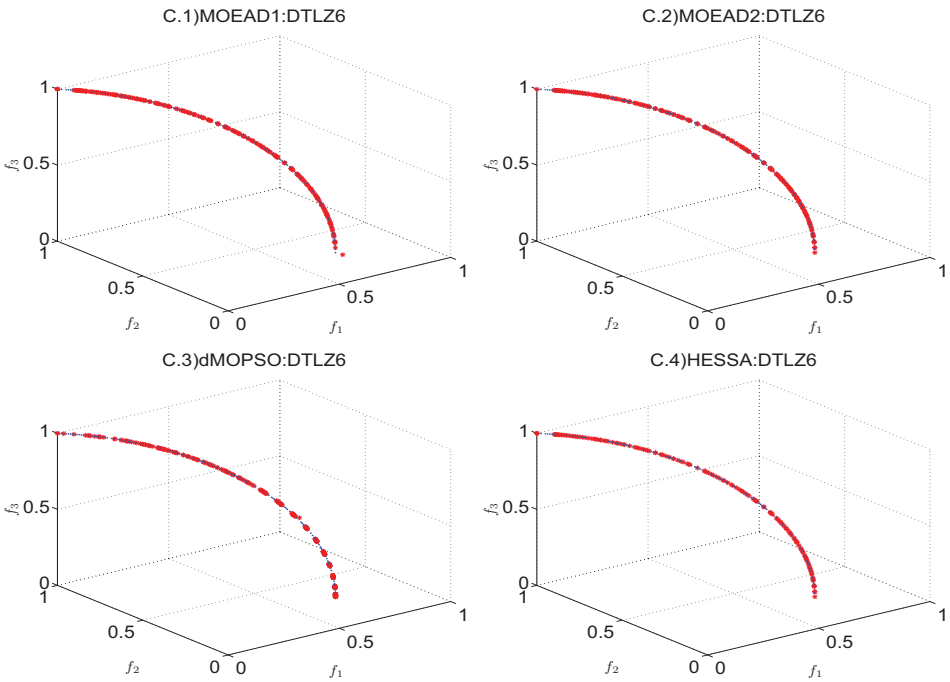


Figure 7.16: The Pareto fronts achieved for DTLZ6 test problems

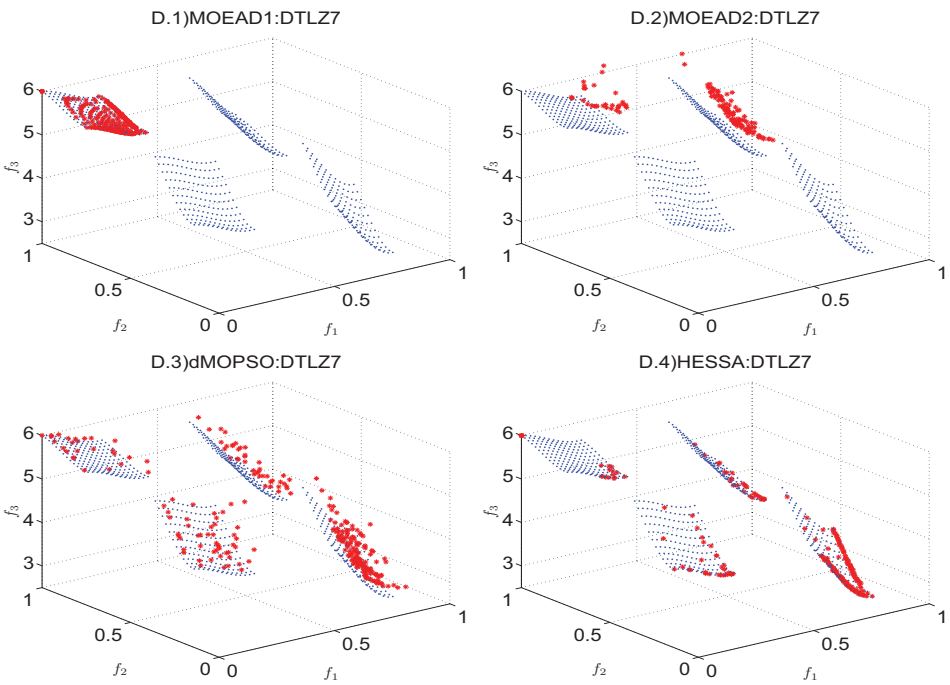


Figure 7.17: The Pareto fronts achieved for DTLZ7 test problems

MOPs commonly used in the literature. HESSA is also compared with three state of the art MOEAs. A set of quality indicators is also considered to evaluate the performance for all the compared MOEAs. The experimental results indicate the superiority of HESSA over both MOEA/D and dMOPSO on the most of test problems used. They also indicate that HESSA has an average performance highly competitive with respect to the compared MOEAs based on the assessment indicators used in this study. The contribution of HESSA is the combination among different cooperative search operators that intensify the search process to discover the promising regions in the search space and enhance the ability to explore high quality solutions. The second contribution is the ability to adapt the search process by using the suitable search operator to the problem on hand.

Chapter 8

Conclusions and Perspectives

Contents

8.1 Conclusions 183
8.2 Perspectives 185

8.1 Conclusions

In this chapter, the general results obtained from the previous chapters will be summarized. Moreover, the contributions of this thesis will be mentioned concerning every proposed algorithm.

In chapter 4, a hybrid evolutionary metaheuristics (HEMH) based on DM-GRASP and greedy randomized path-relinking to solve multiobjective knapsack problems is presented. The proposed HEMH is verified using a set of test MOKP instances commonly used in the literature. The HEMH is compared with four of the most popular MOEAs that considered as the state-of-the art. A set of quality assessment indicators is also considered to evaluate the performance for all the compared MOEAs. The experimental results indicate the superiority of the decomposition based MOEAs over the Pareto dominance based MOEAs. They also indicate the superiority of local search based MOEAs especially the proposed HEMH. Since, it has an average performance highly competitive with respect to the compared MOEAs based on the assessment indicators used in the study. The main contribution of the proposed HEMH is the combination among different metaheuristics techniques that intensify the search process in discovering the most promising regions in the search space and enhance the ability to explore high quality solutions. The second contribution is the ability to find a good approximation set of high quality solutions using a small set of uniformly distributed search directions due to the use of path-relinking and local search strategies.

In chapter 5, four different hybridization variants within the MOEA/D framework are presented. The first one is called MOEAD$_{de}$ which involves the adaptive discrete differential evolution as a recombination operator within the MOEA/D

Framework. The second is called MOEAD$_{pr}$, which uses the greedy path relinking operator with the standard reproduction operators. In the third and the fourth variants, both of adaptive discrete differential evolution and greedy path-relinking are adopted. The four proposals are compared with the original MOEA/D and SPEA2 using a set of MOKP test instances commonly used in the literature. A set of quality assessment indicators was also used to assess the performance of the compared algorithms. The experimental results indicate the superiority of all proposed hybrid variants over the original MOEA/D and SPEA2 for most test instances. In bi-objective test instances, we found that variants that contained greedy path-relinking have the superiority, especially MOEAD$_{dp2}$ variant. On the other hand, in case of instances with three or four objectives, the MOEAD$_{de}$ variant has the best performance followed by MOEAD$_{dp2}$. They have an average performance highly competitive with respect the original MOEA/D and SPEA2 based on the assessment indicators used in this study. The general conclusions can be drawn are: for bi-objective MOKP test instances, greedy path-relinking operator has the first rank followed by the adaptive discrete differential evolution and the standard reproduction by crossover and mutation. Where in MOKP test instances with three or four objectives, the adaptive discrete differential evolution has the superiority, followed by path-relinking and then the standard reproduction by crossover and mutation.

In chapter 6, an improved hybrid evolutionary metaheuristics HEMH2 and two other variants called HEMH$_{de}$ and HEMH$_{pr}$ are proposed to enhance HEMH performance. The HEMH2 adopts the inverse greedy procedure in its initialization phase. Both adaptive binary differential evolution and greedy path relinking operators are used. The HEMH$_{de}$ only adopts the adaptive binary differential evolution whereas, HEMH$_{pr}$ uses crossover and mutation beside greedy path relinking. The proposals are compared with the original MOEA/D and their predecessor HEMH using a set of MOKP test instances from the literature. A set of quality assessment indicators is also used to assess the performance. The experimental results indicate the superiority of all proposals over the original MOEA/D and their predecessor HEMH based on the assessment indicators used in this study. According to these results, we can deduce that the adaptive binary differential evolution included in both HEMH2 and HEMH$_{de}$ has better exploration capabilities which overcome the local search capabilities that contained in the original HEMH. Therefore, both of HEMH2 and HEMH$_{de}$ outperform HEMH. In some cases, HEMH$_{de}$ can achieve highly competitive results compared with HEMH2 based on the adaptive binary differential evolution which can achieve better performance than greedy path relinking operator.

In chapter 7, a hybrid evolutionary approach with search strategy adaptation (HESSA) is presented for handling multiobjective continuous problems. In

HESSA, the search process is controlled by adapting the search strategies used during the evolution process. HESSA is verified using a set of test MOPs commonly used in the literature. HESSA is also compared with three state of the art MOEAs. A set of quality assessment indicators is also considered to evaluate the performance for all the compared MOEAs. The experimental results indicate the superiority of HESSA over both MOEA/D and dMOPSO on the most of test problems used. They also indicate that HESSA has an average performance highly competitive with respect to the compared MOEAs based on the assessment indicators used in this study. The contribution of HESSA is the combination among different cooperative search strategies that intensify the search process to discover the promising regions in the search space and enhance the ability to explore high quality solutions. The second contribution is the ability to adapt the search process by using the suitable search strategy to the problem on hand.

Finally, this thesis provides several hybrid evolutionary metaheuristics algorithms for handling multiobjective decision making problems. The general conclusion can be drawn is that all of the proposed algorithms have the ability to achieve highly competitive results with respect to the state of the art algorithms taken from the literature.

8.2 Perspectives

Concerning the proposed algorithms, there are several possible aspects for future work have arisen form this research. These aspects can be summarized as follows:

1. General perspectives
    - The tuning parameters of the proposed approaches especially HEMH2, HEMH_{de} and HESSA will be investigated as well as their convergence analysis.
    - Investigate and study how to exploit decision makers' preferences to guide the search process for the proposed algorithms.
    - Fuzzy AHP can be combined in the proposed algorithms to help in selecting the most suitable efficient solution according to the decision makers' preferences.
    - Hybridization with other techniques such as scatter search technique, ant-colony optimization and DNA computing,...etc should be studied.
    - Fuzzy linguistic variables may be used to extract the preferences of the decision makers which can be used to direct the search efforts to the preferred regions instead of the whole search space. This could lead to minimize the search time.

- Enhancing the performance of the archiving process used in the proposed algorithms for controlling the quality of the obtained solutions.
- Parallel implementations should be investigated for improving the proposed approaches.
- Large scale problems should be considered to get benefit of the proposed approaches.

2. Perspectives concerning HEMH2 and its variants

   - Studying how to exploit other metaheuristics to improve the performance of HEMH2 and its variants and to extend their capabilities to solve other types of multiobjective combinatorial optimization problems and real world problems.
   - Fuzzy logic controller based on quality assessment metrics could be adopted for adapting the neighborhood size for each subproblem in HEMH2.

3. Perspectives concerning HESSA

   - Improving the performance of HESSA by extending the idea of the dynamic sized-neighborhoods and by using other search strategies.
   - Studying the use of quality assessment metrics to adapt the search strategies in HESSA as will as the archiving strategy.

Appendix A

Résumé Général de la Thèse

Contents

A.1 Introduction 187

A.2 Exposé du problème 189

A.3 Historique des Métaheuristiques 190

A.4 Contribution de la thèse 194

A.5 HEMH: Métaheuristiques évolutionnaire hybrides . . . 196

A.5.1 Contexte du HEMH 196

A.5.2 L'algorithme HEMH 197

A.5.3 Résultats du HEMH 198

A.6 Métaheuristiques hybride basées sur le MOEA/D 198

A.6.1 Algorithmes pour les variantes hybrides 199

A.6.2 Les résultats des variantes hybrides 200

A.7 HEMH2: Un HEMH amélioré 201

A.7.1 L'algorithme HEMH2 202

A.7.2 Résultats du HEMH2 203

A.8 HESSA: Une stratégie de recherche basée sur l'adaptation 203

A.8.1 Adaptation de la stratégie de recherche 203

A.8.2 La méthode HESSA 204

A.8.3 L'algorithme HESSA 206

A.8.4 Résultats de la méthode HESSA 207

A.9 Conclusions et perspectives 207

A.1 Introduction

La prise de décision est une partie intégrante de notre vie quotidienne. Elle concerne les individus, les sociétés, les grands groupes internationaux, les nations

et va jusqu'aux organisations au niveau global. Elle considère des situations s'étendant dans la complexité, des problèmes simples aux plus complexes à objectifs multiples. Ainsi, avec des objectifs multiples, l'incompatibilité apparaîtra. La prise de décision multi-objectifs (MODM) [Zeleny 1982, Gál *et al.* 1999, Triantaphyllou 2000] traite des situations de décision où le décideur a plusieurs objectifs habituellement contradictoires. Dans les problèmes typiques de la vie réelle, il n'y a aucune alternative idéale pouvant être considérée comme solution optimale pour chaque objectif. Par Conséquent, la tâche la plus importante dans des problèmes de décision multicritères ou multiobjectifs est de trouver une bonne solution de compromis qui remplit la meilleure alternative du point de vue du décideur, prenant en considération tous les objectifs simultanément. En conséquence, la qualité du compromis dépend des préférences du décideur.

Plusieurs situations réelles sont représentées comme de problèmes d'optimisation multiobjectifs (MOP). Ces problèmes sont souvent caractérisés par leur grande taille et la présence de multiples objectifs contradictoires. En général, la tâche principale dans le processus d'optimisation multiobjectifs consiste à identifier l'ensemble des solutions Pareto optimales, ou obtenir une bonne approximation vis-à-vis du front de Pareto (PF). Plusieurs métaheuristiques ont été introduites durant les trente dernières années [Coello *et al.* 2006], comme les algorithmes évolutionnaires (EA), les stratégies d'évolution (ES), le recuit simulé (SA), la recherche *Tabu* (TS), la Scatter Search (SS), l'optimisation par essaim particulaires (PSO), l'évolution différentielle (DE) [Blum & Roli 2003].

Les algorithmes évolutionnistes multi-objectifs (MOEAs) sont des domaines de recherche très prometteurs aujourd'hui. En fait, ces algorithmes permettent de fournir plusieurs avantages pour résoudre les problèmes d'optimisation difficiles. Plusieurs travaux sur la résolution des problèmes MOPs et leurs applications utilisant les algorithmes évolutionnistes sont proposés dans la littérature [Deb *et al.* 2000, Farina *et al.* 2004, Jaszkiewicz 2003, Zitzler *et al.* 2001, Zitzler & Thiele 1999]. Les approches de NSGAII [Deb *et al.* 2000] et de SEPA2 [Zitzler *et al.* 2001] sont les approches Pareto les plus populaires à base de MOEAs. Basée sur plusieurs méthodes traditionnelles de programmation mathématiques [Miettinen 1999], l'approximation du PF peut être décomposée en plusieurs sous-problèmes mono-objectif.

Plusieurs approches de MOEAs adoptent ce principe d'ailleurs, comme dans les approches de MOGLS [Jaszkiewicz 2002b] et de MOEA/D [Zhang & Li 2007]. Plusieurs algorithmes de recherche visent à obtenir le meilleur de l'ensemble des différentes métaheuristiques qui s'exécutent ensemble, qui sont complémentaires et qui augmentent leurs capacités d'exploration. Ces méthodes sont généralement appelés des *méthaheuristiques hybrides*. La diversification et l'intensification [Blum & Roli 2003] sont deux éléments majeurs lors de la con-

ception d'une méthode de recherche globale. La *diversification* signifie la capacité de visiter plusieurs régions de l'espace de recherche, tandis que l'*intensification* signifie la capacité d'obtenir de bonnes solutions pour ces régions. Un algorithme de recherche permet de satisfaire ces deux buts, pour faire face aux conflits qui peuvent exister. Les métaheuristiques hybrides permettent de contrôler cet équilibre [Lozano & García-Martínez 2010].

Dans cette thèse, nous proposons de nouvelles metaheuristiques hybrides pour faire face aux problèmes combinatoires multiobjectifs et aux problèmes d'optimisation continue multiobjectifs. Les expériences que nous avons menées démontrent que ces nouvelles approches permettent d'obtenir de meilleurs résultats vis-à-vis des approches de l'état de l'art. Ce résumé est organisé comme suit : après l'introduction, la section A.2 présente brièvement l'énoncé du problème. Dans la section A.3 un bref aperçu sur les métaheuristiques est présenté. Les contributions de la thèse sont mises en évidence dans la section A.4. Les travaux proposés qui sont présentés dans cette thèse, également publiés dans [Kafafy *et al.* 2011], [Kafafy *et al.* 2012b], [Kafafy *et al.* 2012a] et [Kafafy *et al.* 2013] sont brièvement présentés et discutés dans, respectivement, les sections A.5, A.6, A.7 et A.8. Enfin, une brève conclusion achève ce résumé dans la section A.9.

A.2 Exposé du problème

Dans l'optimisation multiobjectif, il n'y a généralement pas de solution optimale qui satisfasse tous les objectifs à la fois. Il faut donc trouver des solutions de compromis, non dominées par d'autres solutions, appelées solutions Pareto optimales [Chankong & Haimes 1983]. Sans perte de généralité, le problème d'optimisation multi-objectifs (MOP) est formulé comme suit:

$$\begin{array}{ll} Min: & F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \\ s.t.: & g_j(x) \leq 0, \forall j = 1, 2, ..., k \\ & x = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n \end{array} \tag{A.1}$$

Où $F(x)$ est un vecteur des objectifs à m-dimensions, $f_i(x)$ est la $i^{ème}$ fonction objectif à minimiser, $\forall i = 1, ..., m$, $x = (x_1, x_2, \cdots, x_n)^T$ est le vecteur décision à n-dimensions, $g_j(x)$ est la $j^{ème}$ contrainte, $\forall j = 1, 2, ..., k$. Une solution $x \in \mathbb{R}^n$ réalisable doit respecter les k contraintes.

Il est important dans l'optimisation multicritère d'évaluer et de comparer la qualité relative des sorties respectives des méthodes de résolution et des algorithmes, nous renvoyons à [Zitzler *et al.* 2008] pour plus de détails. Dans cette thèse, deux types d'indicateurs d'évaluation sont utilisées. Le premier

concerne les indicateurs binaires qui sont utilisés pour comparer chaque couple de techniques telles que la couverture d'ensemble (I_C) [Zitzler & Thiele 1999]. Le deuxième type concerne les indicateurs unaires qui sont utilisés pour évaluer chaque technique indépendamment des autres, comme les indicateurs : Hypervolume (I_{Hyp}) [Zitzler & Thiele 1999], Generational Distance (I_{GD}), Inverted Generational Distance (I_{IGD}), R_3-indicator (I_{R_3}) [Knowles & Corne 2002], Maximum Spread (I_{MS}) [Zitzler *et al.* 2000] et l'indicateur Unary Additive Epsilon ($I_{\varepsilon+}$) [Zitzler *et al.* 2003].

Les méthodes classiques pour générer les solutions Pareto optimales agrègent les fonctions objectif en une seule fonction paramétrée par analogie avec la prise de décision avant la recherche. Plusieurs cycles d'optimisation avec différents paramétrages sont effectués afin de parvenir à un ensemble de solutions qui approche l'ensemble optimal de Pareto. Les principales méthodes pour résoudre le problème de type MODM sont des techniques de Scalarization [Zeleny 1982, Miettinen 1999, Gass & Saaty 1955, Miettinen 1999, Das & Dennis 1998], des approches interactives [Sakawa 1982, Branke *et al.* 2008], Goal Programming [Charnes *et al.* 1955, Charnes & Cooper 1961, Ijiri 1965, Hwang & Masud 1979, Lee & Olson 1999, Jones & Tamiz 2010] et Fuzzy programming [Sakawa 1993].

Ces méthodes classiques présentent quelques inconvénients. Premièrement, les fonctions objectif et/ou les contraintes du problème doivent satisfaire certaines hypothèses telles que la dérivabilité, la continuité, etc. Deuxièmement, ces méthodes ne donnent qu'une bonne solution à la fois. Troisièmement, ces méthodes nécessitent un temps de calcul long exponentiellement proportionnel à la taille du problème pour atteindre un ensemble de bonnes solutions. Quatrièmement, elles ne peuvent pas trouver toutes les solutions Pareto optimales lorsqu'il s'agit de problèmes particuliers avec les fronts de Pareto non convexes. Enfin, la plupart des méthodes classiques nécessitent une connaissance préalable, par exemple sur les poids appropriés ou valeurs ε [Deb 2001].

Pour pallier à ces limitations, les chercheurs ont trouvé que les métaheuristiques représentent un outil prometteur qui peut pallier aux inconvénients des méthodes MODM conventionnels. Ces techniques ont la capacité de trouver un ensemble de solutions optimales à chaque exécution de la simulation. Dans la section A.3 suivante, un bref aperçu des métaheuristiques est présenté.

A.3 Historique des Métaheuristiques

La recherche globale et les techniques d'optimisation peuvent être classés en deux catégories de base : déterministes et probabilistes. Les algorithmes déterministes sont le plus souvent utilisées dans le cas de problème de dimension raisonnable [Russell & Norvig 2010, Bednorz 2008,

Hwang & Masud 1979, Goldberg 1989, Mostaghim & Teich 2003]. Si la dimension de l'espace de recherche est très élevée, il devient plus difficile de résoudre un problème de manière déterministe. En outre, de nombreux problèmes d'optimisation multiobjectif sont de grande dimension, discontinus, multimodaux, et/ou NPcomplets ; ils sont appelés irréguliers [Veldhuizen 1999]. Par conséquent, les algorithmes probabilistes (stochastique) entrent en jeu comme pour produire des solutions de bonne qualité (quasi-optimales) dans un délai raisonnable [Robbins & Monro 1951, Bledsoe & Browning 1959, Bremermann 1962]. Les techniques de recherche probabilistes comprennent au moins une fonction à base de nombres aléatoires [Hromkovic 2005, Goldberg 1989, Michalewicz 1996]. Les méthodes stochastiques nécessitent une fonction qui attribue une valeur d'ajustement pour chaque solution possible, et un mécanisme de mise en correspondance entre le problème et les domaines de l'algorithme. Bien que certains d'entre eux puissent trouver l'optimum, la plupart ne peuvent pas garantir cette solution optimale [Goldberg 1989]. Dans les algorithmes d'optimisation globale, les heuristiques permettent de décider lequel d'un ensemble de solutions possibles doit être examinée par la suite. Les heuristiques sont habituellement utilisées dans les algorithmes déterministes pour définir l'ordre de traitement des solutions candidates, comme cela se fait dans une méthode glouton. Alors que les méthodes probabilistes ne peuvent tenir compte de ces éléments de l'espace de recherche qui ont été sélectionnés par l'heuristique dans d'autres calculs.

Une heuristique [Michalewicz & Fogel 2004] peut être définie comme une "technique qui vise les bonnes solutions (quasi-optimales) à un coût de calcul raisonnable sans être en mesure de garantir la faisabilité ou l'optimalité, voire dans de nombreux cas à indiquer comment approcher de l'optimalité une solution particulière réalisable" [Reeves 1993]. Au cours des trois dernières décennies, de nouveaux algorithmes heuristiques avancées communément appelés "Metaheuristiques" ont été largement développés et appliqués à une variété de problèmes d'optimisation [Reeves 1993, Voss *et al.* 1999, Glover & Kochenberger 2003, Aarts & Lenstra 1997, Osman & Laporte 1996, Rayward-Smith 1996]. Le terme "Métaheuristique" est d'abord introduit par Glover dans [Glover 1986]. Il peut être décrit comme une stratégie de recherche itérative qui guide le processus en dehors de l'espace de recherche dans l'espoir de trouver la solution optimale.

Selon Voss *et al.* [Voss *et al.* 1999] une métaheuristique est décrite comme "un processus itératif maître qui guide et modifie les opérations d'heuristiques subordonnées à produire efficacement des solutions de grande qualité. Il peut manipuler une solution unique complète (ou partielle) ou un ensemble de solutions à chaque itération. Les heuristiques subordonnées peuvent être de niveau élevé ou

faible, comme elle peuvent se résumer à une recherche locale simple, ou tout simplement une méthode de construction". Les métaheuristiques représentent une nouvelle classe d'algorithmes approximatifs qui tentent de combiner des méthodes heuristiques de base dans les méthodes de haut niveau afin d'explorer efficacement l'espace de recherche. Comme les algorithmes approximatifs, les métaheuristiques sacrifient la garantie de trouver des solutions optimales dans le but d'obtenir de bonnes solutions pour un temps réduit de façon significative. Nous nous référons à [Talbi 2009, Blum & Roli 2003, Blum & Roli 2008] pour plusieurs autres définitions proposées.

Le succès des métaheuristiques repose sur la fourniture d'un équilibre dynamique et adaptatif entre l'exploitation (*intensification*) des expériences accumulées de recherche et l'exploration (*diversification*) de l'espace de recherche pour identifier de nouvelles régions. L'intensification permet de concentrer la recherche dans certaines régions de l'espace, tandis que la diversification permet l'expansion de la recherche en explorant les régions non visités de l'espace. Les mécanismes d'intensification et de diversification sont des éléments fondamentaux de toute méthode de recherche globale.

Les métaheuristiques peuvent être classées selon différents aspects qui sont généralement liés à la façon dont elles fonctionnent sur l'espace de recherche. Elles peuvent être classées comme "solution unique" versus "population" [Boussaïd *et al.* 2013], "déterministe" versus "stochastique", "inspiré de la nature" versus "non-inspiré de la nature" [Blum & Roli 2003], "avec mémoire" versus "sans mémoire" [Taillard *et al.* 2001], etc.

Pour les métaheuristiques n'utilisant qu'une solution, le processus de recherche commence par l'amélioration d'une solution initiale unique qui se déplace de manière itérative comme trajectoire dans l'espace de recherche [Crainic & Toulouse 2003]. Les algorithmes à base de solution unique comprennent une heuristique constructive comme les heuristiques gloutonnes [Edmonds 1971] et GRASP [Glover & Kochenberger 2003, Feo & Resende 1989, Feo & Resende 1995], la recherche locale simple [Aarts & Lenstra 1997] et ses extensions intelligentes qui améliorent ses capacités d'échapper à l'optimum local tel que recherche locale réitérée [41], la recherche à voisinage variable [Mladenović & Hansen 1997], la recherche locale guidée [Voudouris 1997, Voudouris & Tsang 1999], le recuit simulé [Kirkpatrick *et al.* 1983, Černỳ 1985] et la méthode tabou [Glover 1986, Glover 1989].

Pour les métaheuristiques à base de population, une population de solutions est adoptée plutôt que d'opérer sur une solution unique. Les algorithmes à base de populations les plus couramment utilisés sont les algorithmes évolutionnaires

et les algorithmes basés sur l'intelligence en essaim (SI). Les algorithmes évolutionnaires simulent le processus d'évolution naturelle qui se base sur le concept darwinien de la "survie du plus fort". Ils abordent les problèmes d'optimisation difficiles grâce à l'amélioration d'une population de solutions initiales en utilisant des opérateurs de sélection, de recombinaison et de mutation tels que des algorithmes génétiques (GA) [Holland 1975], les stratégies évolutives (ES), évolution différentielle (DE) [Price *et al.* 2005], la recherche de nuages [Laguna *et al.* 2003], le le Path Relinking [Glover 1996, Glover *et al.* 2000], les algorithmes mémétiques (MA) [Moscato 1989, Moscato 1999], etc. Dans les algorithmes SI, l'idée est de produire l'intelligence artificielle en exploitant l'analogie avec l'interaction sociale, plutôt que les capacités cognitives purement individuelles, tels que l'optimisation par essaims particulaires (PSO) [Eberhart *et al.* 2001, Kennedy & Eberhart 1995] et l'optimisation par colonie de fourmis (ACO) [Dorigo *et al.* 1996], etc.

La résolution de problèmes d'optimisation multi-objectifs à l'aide de métaheuristiques devient un champ de recherche très actif, en particulier avec les métaheuristiques à base de populations. Dans leur fonctionnement, les métaheuristiques de trois composants importants : la stratégie de l'affectation de l'ajustement, la préservation de la diversité et de l'élitisme. L'affectation de l'ajustement assigne une forme scalaire à un vecteur de fonctions d'objectif afin de guider l'algorithme de recherche vers les solutions optimales de Pareto. Il existe quatre grandes catégories dans les stratégies d'affectation de d'ajustement utilisés dans les métaheuristiques multi-critères, les approches scalaires [Ulungu *et al.* 1999, Hansen 1997], les approches fondée sur des critères [Fishburn 1974, Fishburn 1974], les approches à base du Pareto [Goldberg 1989, Zitzler *et al.* 2004] et les approches fondées sur des indicateurs [Zitzler *et al.* 2004]. La stratégie de préservation de la diversité contribue à générer un ensemble de bonnes solutions dans l'espace de l'objectif et/ou de décision. L'élitisme ou stratégie d'archivage est un mécanisme servant à maintenir les solutions de bonne qualité rencontrés pendant le processus de recherche. Ainsi, une convergence rapide de la population peut être atteinte [Zitzler & Thiele 1999, Laumanns *et al.* 2002, Hernández-Díaz *et al.* 2007].

De nombreux chercheurs ont naturellement développé des algorithmes évolutionnaires pour résoudre des problèmes multi-objectifs (MOEAs). Les MOEAs peuvent être classés en quatre catégories. Les algorithmes fondés sur des critères sont marquées à l'aide de schéma de sélection en fonction e chaque objectif séparément comme dans (VEGA) [Schaffer 1985]. Les algorithmes de Pareto utilisent un système de sélection basé sur le concept d'optimum de Pareto. Ils peuvent être divisés en deux générations. Dans la première génération [Goldberg 1989],

l'utilisation du partage d'ajustement et le partage nichage combiné avec classement Pareto est considérée. Elle contient des approches telles que NSGA [Srinivas & Deb 1994], NPGA [Horn *et al.* 1994] et MOGA [Fonseca *et al.* 1993]. La deuxième génération est née avec l'introduction de la notion d'élitisme. Il contient des approches telles que la SPEA [Zitzler & Thiele 1999], SPEA2 [Zitzler *et al.* 2001], PAES [Knowles & Corne 2000], NSGA-II [Deb *et al.* 2000], NPGA2 [Erickson *et al.* 2001, Horn *et al.* 1994, Oei *et al.* 1991], PESA [Corne *et al.* 2000] et MOPSO [Coello Coello & Lechuga 2002, Hu & Eberhart 2002, Parsopoulos & Vrahatis 2002, Mostaghim & Teich 2003]. Les approches fondées sur des indicateurs dont la recherche est guidée par un indicateur de qualité de la performance [Zitzler & Künzli 2004]. Et les approches à base de décomposition où le MOP est divisé en série de N problèmes mono-objectif qui commencent simultanément par les algorithmes tels que MOGLS [Jaszkiewicz 2002b], MOEA/D [Zhang & Li 2007] et dMOPSO [Martínez & Coello 2011].

Deux ou plusieurs algorithmes métaheuristiques peuvent être combinées pour développer une approche hybride mieux adaptée pour un problème donné [Glover & Kochenberger 2003, Raidl 2006, Cotta-Porras 1998, Talbi 2002, BLUM *et al.* 2005, El-Abd & Kamel 2005, Puchinger & Raidl 2005, Puchinger & Raidl 2005]. La motivation principale de concepts d'hybridation des métaheuristiques est d'obtenir des systèmes plus performants qui exploitent et combinent les avantages les méthodes employées séparément.

L'hybridation des métaheuristiques dans le domaine d'optimisation multicritère suit généralement trois schémas de base. Le premier est la combinaison entre les différentes métaheuristiques qui fournit un comportement plus efficace et de plus grande flexibilité que d'utiliser les métaheuristiques pures comme indiqué dans [Talbi 2002, Ishibuchi & Murata 1998, Jaszkiewicz 2002b]. Le second associe les métaheuristiques avec des méthodes exactes, telles que les techniques de Branch and Bound [Basseur *et al.* 2004]. Le troisième schéma associe les métaheuristiques avec des techniques d'exploration de données, comme dans [Jourdan *et al.* 2005, Coyne & Paton 1994, Basseur *et al.* 2003]. Dans cette thèse, certaines métaheuristiques hybrides pour résoudre des problèmes d'optimisation multi-objectifs seront exposées.

A.4 Contribution de la thèse

Les principales contributions de cette thèse sont résumées dans les points suivants:

- Tout d'abord, nous avons proposé l'algorithme HEMH (Hybrid Evolutionary MetaHeuristic). HEMH utilise l'algorithme DM-GRASP pour con-

struire une population initiale de solutions de bonne qualité dispersées le long de l'ensemble des solutions Pareto optimales. La phase de génération de populations successives à partir de cette population initiale est assurée par, soit des opérateurs de reproduction (croisement, mutation), soit par l'algorithme du Path Relinking glouton randomisé. L'ensemble des bonnes solutions trouvées au cours de cette phase est collecté dans une archive externe. Les résultats montrent que HEMH génère un ensemble de solutions de grande qualité. Une étude comparative a été réalisée pour étudier l'effet de l'hybridation de différentes méthodes de génération de solutions. Quatre variantes d'hybridation ont été étudiées : MOEAD_{de}, MOEAD_{pr}, MOEAD_{dp1} et MOEAD_{dp2}. Les résultats expérimentaux montrent la supériorité de toutes les variantes hybrides proposées sur les algorithmes originaux : MOEA/D et SPEA2. Malgré ces bons résultats, notre approche possède quelques limitations qui sont levées dans une version améliorée de HEMH : HEMH2 et deux autres variantes, appelées HEMH_{de} et HEMH_{pr}.

- Contrairement à HEMH, HEMH2 utilise un algorithme glouton inverse simple pour constituer la population initiale. La phase de génération de populations est assurée cette fois-ci par la combinaison de l'algorithme du Path Relinking et du Adaptive Binary DE. Pendant cette phase, une structure de voisinage dynamique est adoptée pour réduire/augmenter le croisement/mise à jour des solutions. En outre, le concept pareto adaptive epsilon est utilisé pour contrôler le processus d'archivage avec la préservation des solutions extrêmes. Les résultats expérimentaux montrent la supériorité de toutes les propositions HEMH2 et HEMH_{de} sur la MOEA/D et HEMH. Il est clair que, le Adaptive Binary DE inclus dans les HEMH2 et HEMH_{de} a de meilleures capacités d'exploration qui pallient aux capacités de recherche locales contenues dans la HEMH original. Ainsi, HEMH2 et HEMH_{de} surpassent HEMH.
- Motivés par ces résultats dans un espace discret, nous avons proposé un nouvel algorithme baptisé HESSA (Hybrid Evolutionary approach with Search Strategy Adaptation) pour explorer un espace continu de recherche. Dans HESSA, le processus de recherche, pendant le processus d'évolution, peut être réalisé par différentes stratégies de recherche. Cela donne la possibilité à HESSA d'adopter la stratégie de recherche appropriée en fonction du problème étudié. En outre, la coopération entre les différentes stratégies conduit à améliorer l'exploration et l'exploitation de l'espace de recherche. L'ensemble proposé est combiné à un cadre évolutif adapté pour favoriser l'intégration et la coopération. L'ensemble des bonnes solutions trouvées au cours de la recherche est stocké dans un référentiel externe qui est utilisé

comme guide global. Les résultats expérimentaux montrent la supériorité de HESSA à la fois sur MOEA/D et dMOPSO dans la plupart des tests.

Tous les algorithmes proposés ont été vérifiés, testés et comparés à certaines méthodes de l'état de l'art des MOEAs, en utilisant un ensemble d'exemples couramment utilisés dans la littérature. Les résultats expérimentaux indiquent que toutes les propositions sont très compétitives et peuvent être considérés comme une alternative fiable.

A.5 HEMH: Métaheuristiques évolutionnaire hybrides

Dans cette recherche, une nouvelle approche hybride appelée HEMH [Kafafy *et al.* 2011] est proposée. HEMH intègre à la fois le DM-GRASP [Ribeiro *et al.* 2006] et le Path Relinking dans le cadre MOEA/D. Cela permet de tirer profit des avantages de ces techniques à des fins de coopération, d'intégration et d'équilibre adéquat entre l'intensification et la diversification pour améliorer les capacités de recherche. L'utilisation de HEMH est motivée par le fait que :

- l'utilisation Data Mining pour extraire les bons motifs, qui seront réutilisés pour construire de nouvelles solutions, permettra d'atteindre la coopération entre les itérations du GRASP.
- la reproduction à partir de solutions de grande qualité conduit souvent à produire une descendance de meilleure qualité.
- l'intégration du Path Relinking aidera à trouver de meilleures solutions grâce à l'intensification dans ces régions.
- le Path Relinking donne la possibilité d'explorer les régions non-convexes pour découvrir des solutions prometteuses.

A.5.1 Contexte du HEMH

Comme MOEA/D [Zhang & Li 2007], HEMH utilise une technique de décomposition (approche basée sur la somme pondérée) pour convertir MOKP formulée dans Equ. 2.7 par N problèmes mono-objectifs en utilisant un ensemble de N vecteurs poids uniformément répartis $\{\Lambda^1, ..., \Lambda^N\}$. HEMH optimise simultanément ces N sous-problèmes. L'ensemble des voisins du $i^{ème}$ sous-problème inclut tous les sous-problèmes avec les T vecteurs de poids $\{\Lambda^1, ..., \Lambda^N\}$ les plus proches du vecteur de poids Λ^i en cours. Le MOEA/D est utilisé dans HEMH avec quelques légères modifications. Il contient deux populations : la population

principale et l'archive. La population principale se compose de N membres dans lesquels un individu est maintenu dans chaque direction de recherche. Il est également utilisé pour définir les voisinages pour chaque sous-problème. L'archive est utilisée pour collecter toutes les bonnes solutions explorées au cours de l'ensemble du processus de recherche. L'archive est périodiquement mise à jour par les nouvelles solutions explorées en ajoutant des solutions non dominées et en enlevant les dominées. Le processus de recherche d'HEMH est divisé en deux phases de base : "*l'initialisation*" et "*La boucle principale*". La phase d'initialisation utilise le DM-GRASP [Jourdan *et al.* 2006, Ribeiro *et al.* 2006] pour obtenir une première série de solutions de bonne qualité uniformément dispersées dans le front de Pareto. Alors que la phase de la "boucle principale" est utilisée pour l'amélioration de cet ensemble et la découverte de plus de solutions dans les régions les plus prometteuses. Dans ce sens, le Path Relinking et les opérateurs génétiques sont appliqués sur l'ensemble des solutions obtenues dans la phase d'initialisation jusqu'à ce que le critère d'arrêt soit atteint.

A.5.2 L'algorithme HEMH

Dans la phase d'initialisation, le DM-GRASP génère une première série de solutions de bonne qualité qui permet d'obtenir la population principale. Tout d'abord, le GRASP initial [Glover *et al.* 2003] est appliqué sur chaque fonction objectif séparément pour construire un ensemble de solutions "d'élite". A partir de cet ensemble on extrait un ensemble de bons motifs en utilisant le data mining. Ensuite, pour chaque sous-problème, l'un des motifs extraits est choisi comme solution partielle pour construire la solution en cours. Dans la "boucle principale", on applique le Path Relinking glouton aléatoire ou l'opérateur de reproduction classique sur les solutions précédemment obtenues dans la phase d'initialisation. Ceci permet d'intensifier le processus de recherche dans les régions entourant le front de Pareto. Cela conduit à concentrer les efforts de recherche sur les régions prometteuses pour la découverte de nouvelles solutions de bonne qualité.

Comme expliqué dans Alg.4.8, La procédure HEMH commence par identifier l'ensemble des voisinages pour chaque sous-ensemble i en calculant la distance euclidienne entre Λ^i et chaque vecteur de poids dans l'ensemble $\{\Lambda^1, ..., \Lambda^N\}$ et en choisissant les T sous-problèmes les plus proches. Les premiers membres de la population sont initialisés à l'aide DM-GRASP. Tout d'abord, le GRASP est appliqué séparément sur chaque fonction objectif en collectant des solutions d'élite dans l'archive. Ensuite, l'extraction de motifs est appliquée sur l'archive pour extraire l'ensemble des motifs $\mathcal{P}$. Pour chaque élément de la population, un motif $p \in \mathcal{P}$ est considéré comme une entrée de la procédure de construction de la solution complète (où la recherche locale est appliquée). Le résultat est le $i^{ème}$ membre dans la population initiale. Dans la deuxième phase, le processus de

recherche est intensifié dans les régions prometteuses au delà des solutions déjà obtenues précédemment, et ce grâce à l'application du Path Relinking glouton randomisé ou encore la reproduction (croisement et mutation). Pour chaque sous-ensemble i, l'intervalle de reproduction/mise à jour (M) est déterminé pour être soit son voisinage (local) avec une probabilité égale à δ, soit l'ensemble de la population (global). Pour générer la nouvelle descendance y, deux parents x_j et x_k sont choisis aléatoirement. Ensuite, la distance de Hamming $\Delta(x_j, x_k)$ entre x_j et x_k est calculée. Le Path Relinking glouton randomisé est appliquée pour générer y seulement si $\Delta(x_j, x_k) \geq \varepsilon$. Dans le cas contraire, la reproduction est prise en compte. Si y n'est pas réalisable, alors y est corrigée. La descendance y est utilisée pour mettre à jour à la fois la solution du $i^{ème}$ sous-problème et des t solutions de M. L'archive est également mise à jour par chaque descendance générée. L'ensemble du processus est répété jusqu'à ce que le critère d'arrêt soit atteint ; l'archive est retournée comme sortie.

A.5.3 Résultats du HEMH

Le HEMH proposé a été testé à l'aide d'un ensemble de tests MOKP couramment utilisé dans la littérature. HEMH est comparé avec quatre MOEAs les plus connus de l'état de l'art : NSGA-II, SPEA2, MOEA/D et GRASPM. Un ensemble d'indicateurs d'évaluation de qualité a également été utilisé pour évaluer la performance de tous les MOEAs comparés tels que I_C[Zitzler & Thiele 1999], I_{Hyp}[Zitzler & Thiele 1999], I_{GD}, I_{IGD} and I_{MS}[Zitzler *et al.* 2000]. Les résultats expérimentaux montrent la supériorité de la décomposition basée sur les MOEAs à base de dominance de Pareto. Ils indiquent également la supériorité de la recherche locale utilisée avec les MOEAs, en particulier le HEMH. En conséquence, le rendement moyen est très compétitif par rapport aux MOEAs. La principale contribution de notre algorithme est la combinaison entre les différentes techniques de métaheuristiques qui intensifient le processus de recherche pour découvrir les régions les plus prometteuses dans l'espace de recherche et améliorent la capacité d'explorer des solutions de bonne qualité. La seconde contribution est la capacité à trouver un bon ensemble d'approximation des solutions de bonne qualité utilisant un petit ensemble de directions de recherche uniformément réparties grâce à l'utilisation du Path Relinking et les stratégies de recherche locale.

A.6 Métaheuristiques hybride basées sur le MOEA/D

Nous étudions ici l'effet de l'utilisation, à la fois de l'opérateur Adaptive Binary DE proposé dans [Zhang *et al.* 2009] et/ou du Path Relinking glouton

[Glover *et al.* 2000] comme opérateur de reproduction à la place de la reproduction standard (croisement et mutation) dans MOEA/D. La motivation de base de ce travail est d'obtenir les combinaisons les plus appropriées d'opérateurs de recherche afin d'améliorer la performance des MOEA/D ; et donc, utiliser ces combinaisons pour améliorer notre HEMH précédemment exposé. Nous avons ici quatre variantes de l'algorithme, la première variante appelée MOEAD$_{de}$, dans laquelle l'Adaptive Binary DE remplace complètement les opérateurs de croisement et de mutation dans MOEA/D. La seconde variante est appelée MOEAD$_{pr}$ dans laquelle l'opérateur du Path Relinking glouton est appliqué avec le croisement et mutation standard après un certain nombre d'évaluations pour garantir l'existence de solutions de bonne qualité. Dans la troisième et la quatrième variante, l'Adaptive Binary DE et le Path Relinking remplacent le croisement et la mutation, ils sont appelés respectivement MOEAD$_{dp1}$ et MOEAD$_{dp2}$.

A.6.1 Algorithmes pour les variantes hybrides

Dans les méthodes MOEAD$_{de}$ et MOEAD$_{pr}$ présentés respectivement dans Alg. 5.5 et Alg. 5.4, l'ensemble Λ de vecteurs uniformes de poids est généré, suivi de la construction de la structure de voisinage. La population initiale est également générée de façon aléatoire. Ensuite, la boucle principale est exécutée jusqu'à la réalisation des évaluations maximales. Afin de générer une nouvelle descendance pour chaque sous-ensemble i, la région de reproduction/mise à jour (M) est déterminée pour être à la fois le voisinage du $i^{ème}$ sous-problème (*local*) ou l'ensemble de la population (*globale*) en fonction d'une certaine probabilité (σ). Cela peut donner de meilleures chances de sélection de parents distincts, ce qui encourage le Path Relinking à être utilisé dans MOEAD$_{pr}$, ou permet l'évolution différentielle de fonctionner sur des individus distincts dans MOEAD$_{de}$. La sélection des parents est ensuite effectuée. Dans le cas de MOEAD$_{pr}$, les deux parents x^j et x^k sont choisis au hasard à partir de M. Ensuite, l'opérateur du Path Relinking n'est utilisée que si la distance de Hamming entre les deux parents sélectionnés est supérieure à une certaine valeur ε, et le nombre d'évaluations $Eval$ dépasse un certain ratio (γ) des évaluations maximales autorisées pour garantir l'application du Path Relinking sur des solutions de bonne qualité. Sinon, l'opérateur de reproduction standard est appliqué pour générer la nouvelle descendance. Dans le cas de la variante MOEAD$_{de}$, trois individus parents distincts sont choisis au hasard pour appliquer l'Adaptive Binary DE. La nouvelle descendance générée est évaluée et utilisée pour mettre à jour le point de référence z et mettre à jour également la population en fonction du paramètre t. Celui-ci permet de limiter le nombre de solutions remplacées. Enfin, l'ensemble des bonnes solutions mis dans la population finale est retourné comme une sortie.

Dans les deux variantes MOEAD$_{dp1}$ et MOEAD$_{dp2}$ présentées dans Alg. 5.6

et Alg. 5.7, quelques modifications sont appliquées sur MOEAD$_{de}$. Ces modifications peuvent être opérées de la manière suivante : lorsque le nombre d'évaluations $Eval$ dépasse une certaine valeur ($\gamma \times MaxEvals$) préalablement déterminée pour utiliser le Path Relinking, nous avons trois parents sélectionnés x^a, x^b et x^c dans l'étape de sélection. Si nous choisissons au hasard deux d'entre eux qui ont distance de Hamming supérieure à une certaine valeur (ε) pour appliquer le Path Relinking au lieu du Adaptive Binary DE, nous obtiendrons la variante MOEAD$_{dp1}$. D'autre part, en supposant que les conditions de distance de Hamming[1] sont remplies, si nous appliquons le Path Relinking sur les trois parents sélectionnés (x^a, x^b et x^c) de la manière suivante : choisir aléatoirement deux individus (x^a,x^c) pour appliquer le Path Relinking générant un nouvel individu y, ensuite appliquant le Path Relinking de nouveau sur y et x^b, nous obtenons la variante MOEAD$_{dp2}$.

A.6.2 Les résultats des variantes hybrides

Dans cette partie, les quatre propositions ont été comparées avec MOEA/D et SPEA2 d'origine en utilisant un ensemble d'exemples MOKP de la littérature. Un ensemble d'indicateurs d'évaluation de la qualité, y compris I_C[Zitzler & Thiele 1999], I_{Rhyp}[Zitzler & Thiele 1999], I_{GD}, I_{IGD} and I_{R_3}[Knowles & Corne 2002], a également été utilisée pour évaluer la performance. Dans la plupart des cas, les résultats expérimentaux montrent la supériorité de toutes les variantes hybrides proposées sur le MOEA/D initial et SPEA2. Dans le cas des tests bi-objectifs, nous avons constaté que le MOEAD$_{pr}$ est meilleur, tandis que le MOEAD$_{de}$ a une moins bonne performance. D'autre part, dans le cas des exemples avec trois ou quatre objectifs, la performance du Adaptive Binary DE est améliorée. Par conséquent, toutes les variantes proposées permettent d'atteindre une meilleure performance. Ils ont un rendement moyen très compétitif par rapport à la MOEA/D et SPEA2 basés sur les indicateurs d'évaluation utilisés dans cette étude. La conclusion générale que nous avons retenu est pour les exemples de test MOKP bi-objectifs, l'opérateur du Path Relinking a le mieux classé, suivi par le croisement standard, la mutation et l'Adaptive Binary DE. Cependant, dans les cas des tests MOKP avec trois ou quatre objectifs, l'du Adaptive Binary DE et le Path Relinking ont un meilleur rendement que le croisement standard et la mutation.

[1] cela veut dire que x^j et x^k peut être choisi pour le Path Relinking seulement si: $\Delta(x^j, x^k) \geq \varepsilon$

A.7 HEMH2: Un HEMH amélioré

Motivés par les résultats obtenus dans [Kafafy *et al.* 2012b], ce travail est une extension de HEMH [Kafafy *et al.* 2011] basé sur l'élaboration d'une nouvelle version appelée HEMH2 avec deux autres variantes: HEMH_{de} et HEMH_{pr}. Les principales motivations de ce travail ont pour but de surmonter les limites des performances de HEMH. Les principales différences entre HEMH2 et son prédécesseur sont brièvement présentées ici :

- la population initiale est créée en utilisant le glouton inverse simple dans chaque direction de recherche plutôt que le DM-GRASP. La qualité des solutions initiales obtenues sera affectée, mais cela nous donnera une meilleure chance lors de la seconde phase d'améliorer et de renforcer le processus de recherche.
- au lieu de recueillir toutes les bonnes solutions, la dominance epsilon du Pareto-adaptatif ($pa\varepsilon$-dominance) [Hernández-Díaz *et al.* 2007] est adoptée pour contrôler la quantité des bonnes solutions recueillies dans les archives.
- la taille du voisinage dynamique, qui permet de diminuer/augmenter le voisinage pour chaque sous-problème, est prise en compte.
- le DE binaire adaptatif est utilisé comme opérateur de reproduction au lieu du croisement et de la mutation avec le Path Relinking.
- le Path Relinking est appliqué seulement après un certain nombre d'évaluations comme stratégie de post-optimisation. Cette action garantit l'existence de solutions de meilleure qualité. En outre, le Path Relinking retourne deux bits à chaque étape.
- Avec HEMH2, la recherche locale est évitée soit après le Path Relinking soit après la construction du glouton inverse tel que proposé dans HEMH.

Pourquoi ces changements? Tout d'abord, il ne fait aucun doute que la génération de la population initiale avec DMGRASP peut construire de meilleures solutions, mais elle nous contraint à utiliser de petites populations. Dans certains cas, la recherche locale nécessite beaucoup plus de temps et d'évaluations pour explorer une petite région spécifiée dans l'espace de recherche. Par conséquent, la "boucle principale" a une petite chance d'améliorer le processus de recherche. Pour contourner cette limitation, la construction du glouton inverse a été proposée. A partir de résultats empiriques, on s'est aperçut que le glouton inverse donne des solutions aussi proches que possible des régions limitrophes qu'un glouton simple. Deuxièmement, en l'utilisation de $pa\varepsilon$-dominance va contrôler la taille de l'archive, en particulier dans de le cas où le nombre d'objectifs est grand. Par conséquent, on économise davantage de ressources en temps et en espace de stockage

tout en conservant la qualité des solutions recueillies. Troisièmement, les faibles performances de la mutation différentielle binaire se produit lors du traitement des individus différentiels avec une grande distance de Hamming. Sélectionner les parents à partir de toute la population peut encourager ce scénario. Avec HEMH2, les parents de chaque sous-problème sont toujours choisis à partir du voisinage ayant une taille dynamique, ceci garantit l'obtention d'individus avec des distances de Hamming appropriées. Quatrièmement, l'Adaptive Binary DE a empiriquement la possibilité d'explorer l'espace de recherche mieux que le croisement et la mutation classique. Ainsi, la performance de HEMH sera améliorée par l'adoption du Adaptive Binary DE pour la reproduction plutôt que le croisement. Enfin, le Path Relinking proposée applique deux aller/retour sur les bits, ce qui minimise tout le temps de réédition des liens. Enfin, éviter la recherche locale permet de gagner en temps et en évaluations.

A.7.1 L'algorithme HEMH2

La procédure HEMH2 est expliquée dans Alg. 6.4. Tout d'abord, un ensemble de N vecteurs de poids uniformément répartis est créé. Ensuite, la structure de voisinage est construite pour chaque sous-problème i en attribuant tous les sous-problèmes, triés par ordre croissant de la distance euclidienne entre les vecteurs de poids et le vecteur de poids courant Λ_i. Ensuite, la population initiale P est créée en appliquant le glouton inverse dans chaque direction de recherche. La boucle principale est exécutée jusqu'à la réalisation des évaluations maximales $MaxEvals$. Pour chaque sous-problème i, la routine de sélection est utilisée pour déterminer la taille actuelle du voisinage B_i tels que: $|B_i| = T + r$, où T et r représentent le nombre de solutions différentes et répétés dans B_i, respectivement. Cela signifie que la routine de sélection étend la taille B_i pour garantir l'existence d'au moins T solutions différentes et choisit au hasard trois d'entre eux x^a, x^b et x^c pour la reproduction. Deux des trois parents sélectionnés x^j et x^k sont choisis au hasard. Ensuite, le Path Relinking n'est utilisé que si la distance de Hamming $\Delta(x^j, x^k)$ est supérieure à une certaine valeur ε et que le nombre d'évaluations $Eval$ dépasse un certain ratio γ des évaluations maximales $MaxEvals$. Sinon, l'Adaptive Binary DE est appliqué pour générer une nouvelle descendance y. La nouvelle descendance générée y est évaluée et utilisée pour mettre à jour le voisinage (B_i) en fonction du paramètre t, ce qui détermine le nombre de solutions remplacées. L'archive est également mise à jour par y selon $pa\varepsilon$-domination [Hernández-Díaz *et al.* 2007]. Enfin, les solutions extrêmes sont mises à jour dans l'archive et retournées comme sortie.

Pour étudier les effets de le DE binaire adaptatif et les opérateurs du Path Relinking distinctement, deux variantes supplémentaires de l'algorithme appelées HEMH_{de} et HEMH_{pr} sont considérées. Toutes les deux ont la même procédure

que HEMH2 à la seule différence que le HEMH$_{de}$ utilise l'Adaptive Binary DE pour la reproduction. Tandis que le HEMH$_{pr}$ remplace l'Adaptive Binary DE dans la procédure HEMH2 par le croisement et la mutation.

A.7.2 Résultats du HEMH2

La méthode HEMH2 et deux autres variantes appelées HEMH$_{de}$ et HEMH$_{pr}$ ont pour but d'améliorer les performances du HEMH. Les propositions ont été comparées avec le MOEA/D et le HEMH d'origine à l'aide d'exemples MOKP de la littérature. Un ensemble d'indicateurs de qualité, comme I_C[Zitzler & Thiele 1999], I_{Rhyp}[Zitzler & Thiele 1999], I_{GD}, I_{IGD} and I_{R_3}[Knowles & Corne 2002] a également été utilisé pour évaluer les performances. Les résultats expérimentaux montrent la supériorité de toutes les propositions sur la MOEA/D et HEMH d'origine en se basant sur les indicateurs d'évaluation utilisés. Selon les résultats, nous pouvons déduire que l'Adaptive Binary DE inclus dans HEMH2 et HEMH$_{de}$ a de meilleures capacités d'exploration qui pallient aux capacités de recherche locales contenues dans le HEMH originel. On peut donc dire que les deux algorithmes HEMH2 et HEMH$_{de}$ sont meilleurs que HEMH. Dans certains cas, HEMH$_{de}$ peut atteindre des résultats très compétitifs par rapport à HEMH2 (basé sur l'Adaptive Binary DE) qui peut atteindre de meilleures performances que le Path Relinking.

A.8 HESSA: Une stratégie de recherche basée sur l'adaptation

Dans les travaux de recherche dans [Kafafy *et al.* 2011, Kafafy *et al.* 2012b, Kafafy *et al.* 2012a], l'influence de l'intégration des différentes métaheuristiques coopératives dans le MOEA/D a été examinée pour les espaces de recherche discrets. Les résultats obtenus nous ont incités à étendre l'idée au cas continu, en élaborant une approche évolutionnaires hybride (HESSA) qui intègre un ensemble de stratégies de recherche adaptatives dans le MOEA/D. L'objectif principal est de tirer parti des avantages de ces stratégies grâce à la coopération et l'intégration. L'objectif est également de rendre l'approche capable de sélectionner la stratégie de recherche adaptée en fonction du problème étudié.

A.8.1 Adaptation de la stratégie de recherche

Dans HESSA, au lieu d'utiliser une seule stratégie, un groupement de plusieurs stratégies de recherche est adopté pour générer les nouvelles solutions (descendances). Pour générer une nouvelle descendance, l'ensemble des candidats est accessible pour sélectionner une stratégie de recherche pour chaque individu cible

dans la population actuelle. Au cours de l'évolution, chaque élément du bassin est considéré lors de la période d'apprentissage (LP). La meilleure stratégie obtenue durant la période d'apprentissage précédente est utilisée pour générer les solutions prometteuses, et la plus probable sera choisie dans la période d'apprentissage actuel et sera utilisée pour générer les nouvelles solutions des descendants. A chaque phase d'apprentissage, la somme des probabilités de sélection de chaque stratégie dans le bassin de candidats est égale à 1. Ces probabilités sont adaptées progressivement au cours du processus de l'évolution. Dans la période initiale d'apprentissage, toutes les stratégies ont la même chance d'être choisies, à savoir, chaque stratégie k a une probabilité $p_k = \frac{1}{K}$, où K est le nombre total de stratégies dans le bassin de candidats. Au cours de chaque période d'apprentissage, chaque stratégie k peut être choisie afin de générer la nouvelle solution en fonction de sa probabilité p_k en utilisant la sélection stochastique universelle [Baker 1987]. Le nombre de sélections de chaque stratégie k est représenté par $calls_k$. Chaque stratégie est considérée pour obtenir un succès si elle a la capacité de générer une descendance capable de mettre à jour la population actuelle. Le nombre d'appels réussis pour chaque stratégie k est inscrit dans suc_k. Le nombre total dans le bassin de candidats est exprimée comme suit : $calls_{tot} = \sum_{l=1}^{L} \sum_{k=1}^{K} calls_{k,l}$, où L est le nombre total de périodes d'apprentissage dans l'ensemble du processus d'évolution. Cependant, après chaque période d'apprentissage l (quand $calls_{tot} \% LP = 0$), la probabilité de sélection de chaque stratégie k pour la prochaine période d'apprentissage $p_{k,l+1}$ sera adaptée selon les formules suivantes :

$$p_{k,l+1} = \frac{sucR_{k,l}}{\sum_{k=1}^{K} sucR_{k,l}} \tag{A.2}$$

$$sucR_{k,l} = \begin{cases} \frac{suc_{k,l}}{calls_{k,l}} + \varepsilon & \text{if } calls_{k,l} > 0, \forall k, l \\ \varepsilon & \text{otherwise} \end{cases} \tag{A.3}$$

où $sucR_{k,l}$ est le taux de réussite de la $i_{ème}$ stratégie dans la période d'apprentissage l. La petite valeur constante $\varepsilon = 0.01$ est utilisée pour éviter les taux de réussite nuls. Par conséquent, les stratégies ayant un taux de réussite nul ont une chance d'être choisies pour générer une descendance. Les deux quantités $suc_{k,l}$ et $calls_{k,l}$ représentent le nombre de succès (invoke) et le nombre total de (invoke) de la $k^{ème}$ stratégie dans la période d'apprentissage l.

A.8.2 La méthode HESSA

Comme MOEA/D [Zhang & Li 2007], HESSA utilise le Tchebycheff pondéré comme technique de décomposition pour convertir la MOP en un ensemble de sous-problèmes mono-objectif. Si nous avons un ensemble de N vecteurs

poids uniformément répartis $\{\Lambda^1, ..., \Lambda^N\}$ suite à la décomposition, nous avons N sous-problèmes mono-objectif. HESSA tente d'optimiser simultanément ces sous-problèmes. Chaque sous-problème i a son propre ensemble de voisins appelé B_i, qui inclut tous les sous-problèmes avec les T vecteurs poids les plus proches $\{\Lambda^{i1}, ..., \Lambda^{iT}\}$ de Λ^i en termes de distance euclidienne. La structure du cadre proposé est résumée comme suit :

- une population P of N individus, $P = \{x^1, ..., x^N\}$, où x^i représente la solution en cours du $i^{ème}$ sous-problème. Chaque individu x^i a sa propre vitesse v^i, sa meilleure position personnelle x^i_{pb} et son d'âge a_i.

- un ensemble de N vecteurs de poids uniformément répartis $\{\Lambda^1,...,\Lambda^N\}$, correspondrait aux N sous-problèmes. Chaque $\Lambda = [\lambda_1,..., \lambda_m]$ possède m composantes correspondant aux m-objectifs, tels que: $\sum_{i=1}^{m} \lambda_i = 1$, $\forall \lambda_i \in \{0/H, 1/H, \cdots, H/H\}$ and $N = C_{m-1}^{H+m-1}$, $\forall H \in \mathbb{Z}^+$.

- un voisinage B_i pour chaque sous-problème $i \in \{1, ..., N\}$, qui comprend tous les sous-problèmes avec les T vecteurs poids les plus proches $\{\Lambda^{i1}, \cdots, \Lambda^{iT}\}$ de Λ^i.

- un ensemble de stratégies de reproduction adaptative contenues dans un bassin (*Pool*) pour générer de nouvelles solutions. Chaque stratégie est choisie en fonction de sa probabilité, comme mentionné ci-dessus. Le tableau A.1 résume l'ensemble des stratégies adoptées.

- une *archive* externe pour collecter les bonnes solutions explorées lors du processus de recherche. L'*archive* joue également le rôle de référentiel des "*leaders*" globaux.

Table A.1: Ensemble des stratégies de reproduction utilisées

Stratégie	**Description de la stratégie**
SBXPM	Le croisement SBX [Deb & Agrawal 1995] est appliqué sur les deux parents suivi par une mutation polynomiale [Deb & Agrawal 1995].
DEXPM	L'évolution différentielle [Chakraborty 2010, Price *et al.* 2005] est appliquée sur trois parents sélectionnés suivie d'une mutation polynomiale [Deb & Agrawal 1995].
MPCPM	Le croisement multiple de parents[Elsayed *et al.* 2011] est appliqué sur trois parents sélectionnés suivis de la mutation polynomiale.
GM	La mutation guidée [Hsieh *et al.* 2007] est utilisée pour produire une descendance depuis son parent et la solution globale guidée.
PSO	Les essaims particulaires [Kennedy & Eberhart 1995] calculent une nouvelle position à partir du parent actuel, de son record personnel et du guide global.

A.8.3 L'algorithme HESSA

Après la construction du cadre proposé, HESSA met en oeuvre deux phases principales. La première est la phase d'*initialisation*, dans laquelle une population initiale est générée de manière aléatoire. La seconde phase représente la *boucle principale* dans laquelle les recherches sont effectuées afin d'améliorer la population initiale. Alg. 7.1 explique la procédure HESSA. Tout d'abord, un ensemble de N vecteurs de poids uniformément répartis est initialisé. Ensuite, la structure de voisinage B_i est construite pour chaque sous-ensemble i en attribuant tous les sous-problèmes correspondants aux T vecteurs de poids les plus proches de Λ_i. Le bassin ($Pool$) de candidats est également construit en utilisant les stratégies de reproduction adoptées. Les archives et le compteur d'évaluation sont initialisés. Dans un second temps, la population initiale est construite. Pour chaque sous-population i, la solution actuelle x_i est initialisée aléatoirement. Ensuite, x_i est évaluée et utilisée pour mettre à jour le point de référence r^* [Zhang & Li 2007], le record personnel x^i_{pb} et l'archive. La vitesse v^i et l'âge a_i sont également initialisés à 0. Le $i_{ème}$ sous-problème est affecté à la population P. Par la suite, la boucle principale est exécutée jusqu'à la réalisation des évaluations maximales $MaxEvals$. Pour chaque sous-ensemble i, l'intervalle d'accouplement/mise à jour M_i est choisi pour être soit le voisinage B_i soit la population entière. Puis, trois solutions parentes différentes sont choisies au hasard à partir de M_i pour la reproduction. Le leader global x^i_{gb} est sélectionné aléatoirement à partir des archives. Une stratégie globale de reproduction S_k est également sélectionnée dans le bassin pour générer la nouvelle descendance y. Selon la stratégie S_k choisie, la descendance y est générée. En cas de l'utilisation de la mutation guidée ou des essaims particulaires, le paramètre âge a_i contrôle le processus de génération. Dans ce cas, si a_i dépasse le l'âge maximum autorisé T_a, une valeur gaussienne : $N(\frac{1}{2}[x^i_{gb} - x^i_{pb}], |x^i_{gb} - x^i_{pb}|)$ est affecté à y. Ensuite, le descendant y est évalué et utilisé pour mettre à jour le point de référence r^*. La population actuelle P est mise à jour en invoquant le module UpdateSolutions. L'archive est également mise à jour par y en fonction de la distance de surpeuplement. Le compteur d'évaluation est mis à jour et vérifié. A la fin de chaque période d'apprentissage, le bassin ($Pool$) est adapté en calculant la probabilité p_k pour chaque stratégie k selon (Equ. A.2). A la fin de l'évolution, l'*archive* est retournée.

Dans le module UpdateSolutions (Alg. 7.3), la descendance y met à jour la population P comme suit : un indice aléatoire j est choisi dans la gamme M_i des mises à jour. Ensuite, la solution actuelle du $j_{ème}$ sous-problème x_j est actualisée si et seulement si y réalise une meilleure performance. Dans ce cas, la réussite de la stratégie choisie Suc_k est augmentée ; l'âge a_j est réinitialisé. Aussi, le record personnel x^j_{pb} est mis à jour de la même manière. Enfin l'index sélectionné j est

éliminé de M_i. Si la solution courante x_j n'est pas mise à jour, son âge a_j est augmenté. Ce processus se poursuit jusqu'à ce que la mise à jour t ou solution M_i devienne vide.

A.8.4 Résultats de la méthode HESSA

HESSA a été testée à l'aide d'un ensemble de tests MOPs [Coello *et al.* 2006, Zitzler *et al.* 2000, Deb *et al.* 2005a] couramment utilisé dans la domaine. HESSA a également été comparée à trois MOEAs de l'état de l'art : MOEA/D_1 [Zhang & Li 2007], MOEA/D_2 [Li & Zhang 2009] et dMOPSO [Martínez & Coello 2011]. Un ensemble d'indicateurs de qualité, incluant I_C[Zitzler & Thiele 1999], I_{Rhyp}[Zitzler & Thiele 1999], I_{GD}, I_{IGD} and $I_{\varepsilon+}$[Zitzler *et al.* 2000] a également été examinée pour évaluer la performance de tous les MOEAs comparés. Les résultats expérimentaux montrent la supériorité de HESSA à la fois sur MOEA/D et dMOPSO dans la plupart des tests effectués. Ils indiquent également que HESSA a une performance moyenne très compétitive par rapport aux MOEAs basés sur les indicateurs d'évaluation utilisés dans cette étude. La contribution de HESSA réside dans la combinaison des différents opérateurs de recherche coopératifs qui intensifient le processus de recherche pour découvrir les régions prometteuses dans l'espace de recherche et d'améliorer la capacité d'explorer des solutions de bonne qualité. La seconde contribution est la capacité d'adapter le processus de recherche en utilisant l'opérateur de recherche approprié au problème étudié.

A.9 Conclusions et perspectives

Dans cette thèse, nous avons proposé plusieurs approches de métaheuristiques hybrides. Ces approches comprennent HEMH, MOEAD$_{de}$, MOEAD$_{pr}$, MOEAD$_{dp1}$, MOEAD$_{dp2}$, HEMH2, HEMH$_{de}$ et HEMH$_{pr}$ pour le domaine discret de recherche et HESSA pour le domaine continu. Ces approches ont été testées et validées sur un ensemble de problèmes d'optimisation multi-objectifs de la littérature. Un ensemble d'indicateurs d'évaluation de la qualité tels que I_C, I_{Hype}, I_{GD}, I_{IGD}, I_{R_3}, I_{MS} et $I_{\varepsilon+}$ a été utilisé pour évaluer les performances des approches hybrides proposées. Toutes les approches proposées donnent des résultats très compétitifs par rapport aux autres méthodes comparables dans la littérature. Les approches proposées sont en mesure d'intensifier le processus de recherche de régions prometteuses de l'espace de recherche et d'améliorer la capacité d'explorer des solutions de bonne qualité. La combinaison de différentes techniques métaheuristiques qui s'intègrent les uns aux autres, a un rôle important. Dans les travaux futurs, les paramètres de réglage des algorithmes proposés seront étudiés ainsi que l'analyse de leur convergence. Nous allons étendre les algorithmes pro-

posés à des applications de la vie réelle. Nous allons examiner aussi comment inclure le décideur dans le processus de recherche.

Appendix B

Detailed Statistical Results

In this appendix, the whole tables of the experimentation that carried out to verify the proposed algorithms in the whole thesis are provided. For the experimental results that provided in chapter 4 to verify the HEMH, the details of the statistical analysis including Coverage indicator I_C (Table B.1), hypervolume I_{Hyp} (Table B.2), generational distance I_{GD} (Table B.3), inverted generational distance I_{IGD} (Table B.4) and the maximum spread I_{MS} (Table B.5) indicators are presented respectively.

For the experimental results provided in chapter 5, the statistical analysis of this comparative study includes the coverage indicator I_C (Table B.6), the referenced hypervolume I_{Rhyp} (Table B.7), the generational distance I_{GD} (Table B.8), the inverted generational distance I_{IGD} (Table B.9) and the R3 I_{R_3} (Table B.10) quality assessment indicators are showed respectively.

With respect to the experimentation that carried out to verify the HEMH2 in chapter 6. The details of the statistical analysis which includes the coverage indicator I_C (Table B.11), the referenced hypervolume I_{Rhyp} (Table B.12), the generational distance I_{GD} (Table B.13), the inverted generational distance I_{IGD} (Table B.14) and the R3 I_{R_3} (Table B.15) indicators are provided respectively.

For the experimentation which carried out to verify the proposed HESSA in chapter 7. The details of the statistical analysis of this comparative study including the coverage indicator I_C (Table B.16), the referenced hypervolume I_{Rhyp} (Table B.17), the generational distance I_{GD} (Table B.18), inverted generational distance I_{IGD} (Table B.19) and the unary additive epsilon $I_{\varepsilon+}$ (Table B.20) indicators are presented respectively.

Table B.1: Detailed results of the Coverage I_c indicator (Median)(Ch. 4)

Coverage Indicator I_c	**Test Instances**								
	KSP252	**KSP502**	**KSP752**	**KSP253**	**KSP503**	**KSP753**	**KSP254**	**KSP504**	**KSP754**
I_c(HEMH,GRASPM)	$6.795E-01$	8.426E-01	8.594E-01	5.312E-01	4.457E-01	3.521E-01	3.429E-01	1.485E-01	8.214E-02
I_c(GRASPM,HEMH)	1.719E-01	8.924E-02	8.451E-02	1.913E-01	1.329E-01	1.182E-01	1.407E-01	9.011E-02	6.674E-02
I_c(HEMH,MOEAD)	9.862E-01	1.000E+00	1.000E+00	8.665E-01	9.139E-01	9.624E-01	6.074E-01	7.057E-01	8.505E-01
I_c(MOEAD,HEMH)	0.000E+00	0.000E+00	0.000E+00	2.553E-02	1.034E-02	1.427E-03	5.195E-02	1.291E-02	2.160E-03
I_c(HEMH,SPEA2)	1.000E+00	1.000E+00	1.000E+00	9.980E-01	1.000E+00	1.000E+00	9.799E-01	1.000E+00	1.000E+00
I_c(SPEA2,HEMH)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	1.321E-04	0.000E+00	0.000E+00
I_c(HEMH,NSGAII)	1.000E+00	1.000E+00	1.000E+00	1.000E+00	1.000E+00	1.000E+00	9.997E-01	1.000E+00	1.000E+00
I_c(NSGAII,HEMH)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(GRASPM,MOEAD)	9.708E-01	1.000E+00	9.739E-01	7.720E-01	8.329E-01	9.078E-01	4.755E-01	5.781E-01	7.077E-01
I_c(MOEAD,GRASPM)	9.615E-03	0.000E+00	1.064E-02	6.227E-02	3.595E-02	3.404E-03	9.661E-02	2.437E-02	1.942E-03
I_c(GRASPM,SPEA2)	1.000E+00	1.000E+00	1.000E+00	9.908E-01	9.991E-01	1.000E+00	8.909E-01	9.449E-01	9.812E-01
I_c(SPEA2,MGRASP)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	5.706E-05	0.000E+00	0.000E+00
I_c(GRASPM,NSGAII)	1.000E+00	1.000E+00	1.000E+00	9.990E-01	1.000E+00	1.000E+00	9.953E-01	1.000E+00	1.000E+00
I_c(NSGAII,GRASPM)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD,SPEA2)	7.031E-01	8.803E-01	1.000E+00	7.494E-01	7.813E-01	9.819E-01	6.123E-01	5.938E-01	6.526E-01
I_c(SPEA2,MOEAD)	7.062E-02	1.087E-02	0.000E+00	1.742E-02	3.969E-03	0.000E+00	8.658E-03	1.726E-03	4.141E-04
I_c(MOEAD,NSGAII)	7.656E-01	9.013E-01	1.000E+00	8.927E-01	9.558E-01	1.000E+00	9.573E-01	1.000E+00	1.000E+00
I_c(NSGAII,MOEAD)	5.882E-02	1.047E-02	0.000E+00	6.954E-03	5.460E-04	0.000E+00	3.517E-04	0.000E+00	0.000E+00
I_c(SPEA2,NSGAII)	4.407E-01	4.182E-01	1.351E-01	5.335E-01	5.955E-01	6.239E-01	6.659E-01	8.934E-01	9.272E-01
I_c(NSGAII,SPEA2)	3.279E-01	3.962E-01	6.740E-01	1.633E-01	6.473E-02	3.059E-02	3.309E-02	0.000E+00	0.000E+00

Table B.2: Detailed results of the hypervolume indicator I_{Hyp} (Ch. 4)

Inst.	Stat.	Algorithms				
		NSGAII	SPEA2	MOEAD	GRASPM	HEMH
KSP252	Max.	6.997E-01	6.795E-01	7.826E-01	7.957E-01	7.980E-01
	Min.	6.006E-01	6.102E-01	7.707E-01	7.940E-01	7.972E-01
	Mean	6.680E-01	6.576E-01	7.763E-01	7.948E-01	7.976E-01
	Med.	6.688E-01	6.606E-01	7.760E-01	7.949E-01	7.976E-01
	S.D.	2.085E-02	1.441E-02	3.255E-03	4.128E-04	2.407E-04
KSP502	Max.	6.193E-01	6.037E-01	7.548E-01	7.718E-01	7.764E-01
	Min.	5.480E-01	5.624E-01	7.442E-01	7.704E-01	7.752E-01
	Mean	5.889E-01	5.842E-01	7.492E-01	7.710E-01	7.757E-01
	Med.	5.906E-01	5.863E-01	7.497E-01	7.710E-01	7.757E-01
	S.D.	1.685E-02	1.208E-02	2.898E-03	3.370E-04	2.766E-04
KSP752	Max.	5.728E-01	5.662E-01	7.599E-01	7.707E-01	7.756E-01
	Min.	5.305E-01	5.294E-01	7.464E-01	7.692E-01	7.746E-01
	Mean	5.516E-01	5.469E-01	7.540E-01	7.702E-01	7.751E-01
	Med.	5.513E-01	5.463E-01	7.543E-01	7.702E-01	7.751E-01
	S.D.	1.186E-02	1.020E-02	2.662E-03	3.393E-04	2.427E-04
KSP253	Max.	4.297E-01	4.220E-01	5.382E-01	5.547E-01	5.587E-01
	Min.	3.909E-01	3.822E-01	5.285E-01	5.532E-01	5.569E-01
	Mean	4.129E-01	3.994E-01	5.342E-01	5.538E-01	5.580E-01
	Med.	4.147E-01	3.987E-01	5.339E-01	5.537E-01	5.581E-01
	S.D.	9.997E-03	9.128E-03	2.341E-03	3.627E-04	3.890E-04
KSP503	Max.	3.325E-01	3.313E-01	5.047E-01	5.255E-01	5.318E-01
	Min.	2.998E-01	2.878E-01	4.933E-01	5.239E-01	5.300E-01
	Mean	3.175E-01	3.070E-01	4.982E-01	5.247E-01	5.308E-01
	Med.	3.190E-01	3.052E-01	4.976E-01	5.246E-01	5.308E-01
	S.D.	9.236E-03	1.053E-02	2.666E-03	4.383E-04	4.036E-04
KSP753	Max.	2.851E-01	2.728E-01	4.919E-01	5.220E-01	5.283E-01
	Min.	2.428E-01	2.449E-01	4.813E-01	5.202E-01	5.263E-01
	Mean	2.665E-01	2.599E-01	4.861E-01	5.211E-01	5.270E-01
	Med.	2.675E-01	2.604E-01	4.863E-01	5.211E-01	5.271E-01
	S.D.	9.554E-03	6.641E-03	2.345E-03	3.933E-04	4.992E-04
KSP254	Max.	2.227E-01	2.200E-01	3.373E-01	3.512E-01	3.561E-01
	Min.	1.987E-01	1.985E-01	3.303E-01	3.495E-01	3.537E-01
	Mean	2.122E-01	2.094E-01	3.334E-01	3.502E-01	3.553E-01
	Med.	2.123E-01	2.102E-01	3.334E-01	3.502E-01	3.554E-01
	S.D.	6.346E-03	5.724E-03	1.851E-03	4.046E-04	5.008E-04
KSP504	Max.	1.467E-01	1.575E-01	2.968E-01	3.245E-01	3.319E-01
	Min.	1.144E-01	1.399E-01	2.878E-01	3.229E-01	3.297E-01
	Mean	1.325E-01	1.498E-01	2.922E-01	3.235E-01	3.306E-01
	Med.	1.312E-01	1.485E-01	2.921E-01	3.234E-01	3.306E-01
	S.D.	7.694E-03	4.868E-03	2.368E-03	4.383E-04	5.611E-04
KSP754	Max.	1.105E-01	1.232E-01	2.745E-01	3.130E-01	3.222E-01
	Min.	8.366E-02	1.058E-01	2.606E-01	3.116E-01	3.207E-01
	Mean	9.766E-02	1.145E-01	2.666E-01	3.124E-01	3.216E-01
	Med.	9.798E-02	1.155E-01	2.669E-01	3.125E-01	3.217E-01
	S.D.	6.995E-03	3.900E-03	3.676E-03	3.598E-04	3.681E-04

Table B.3: Detailed results of the Generational distance indicator I_{GD} (Ch. 4)

Inst.	Stat.	Algorithms				
		NSGAII	SPEA2	MOEAD	GRASPM	HEMH
KSP252	Max.	4.829E-03	5.008E-03	1.738E-03	6.272E-04	5.294E-04
	Min.	2.012E-03	1.701E-03	1.032E-03	3.165E-04	1.933E-04
	Mean	3.240E-03	3.142E-03	1.457E-03	4.020E-04	2.307E-04
	Med.	3.221E-03	3.136E-03	1.450E-03	3.635E-04	2.157E-04
	S.D.	6.467E-04	6.855E-04	1.766E-04	8.522E-05	6.703E-05
KSP502	Max.	7.153E-03	6.835E-03	1.821E-03	4.066E-04	2.113E-04
	Min.	2.555E-03	2.856E-03	1.154E-03	3.191E-04	1.548E-04
	Mean	4.424E-03	4.555E-03	1.458E-03	3.500E-04	1.747E-04
	Med.	4.358E-03	4.395E-03	1.420E-03	3.460E-04	1.746E-04
	S.D.	1.022E-03	9.488E-04	1.804E-04	2.013E-05	1.525E-05
KSP752	Max.	5.686E-03	6.866E-03	1.294E-03	3.161E-04	1.718E-04
	Min.	2.981E-03	3.586E-03	7.719E-04	2.667E-04	1.224E-04
	Mean	4.171E-03	4.993E-03	1.009E-03	2.889E-04	1.462E-04
	Med.	4.185E-03	5.015E-03	1.009E-03	2.893E-04	1.469E-04
	S.D.	6.408E-04	8.126E-04	1.074E-04	1.154E-05	9.424E-06
KSP253	Max.	2.041E-03	1.734E-03	5.314E-04	1.855E-04	1.316E-04
	Min.	1.302E-03	8.992E-04	3.928E-04	1.702E-04	1.198E-04
	Mean	1.622E-03	1.377E-03	4.457E-04	1.771E-04	1.261E-04
	Med.	1.593E-03	1.363E-03	4.366E-04	1.773E-04	1.263E-04
	S.D.	1.840E-04	1.872E-04	3.828E-05	3.574E-06	2.858E-06
KSP503	Max.	2.907E-03	2.562E-03	5.129E-04	1.371E-04	9.769E-05
	Min.	1.863E-03	1.500E-03	3.770E-04	1.266E-04	8.631E-05
	Mean	2.369E-03	1.984E-03	4.468E-04	1.312E-04	9.126E-05
	Med.	2.360E-03	2.010E-03	4.501E-04	1.310E-04	9.113E-05
	S.D.	2.342E-04	2.569E-04	3.233E-05	2.758E-06	2.568E-06
KSP753	Max.	4.594E-03	3.726E-03	5.252E-04	1.110E-04	8.206E-05
	Min.	2.422E-03	2.243E-03	3.989E-04	1.003E-04	7.149E-05
	Mean	3.345E-03	2.912E-03	4.760E-04	1.041E-04	7.739E-05
	Med.	3.412E-03	2.820E-03	4.746E-04	1.041E-04	7.816E-05
	S.D.	4.527E-04	4.380E-04	2.975E-05	2.714E-06	2.505E-06
KSP254	Max.	1.972E-03	1.226E-03	3.186E-04	1.563E-04	1.181E-04
	Min.	1.185E-03	8.447E-04	2.498E-04	1.463E-04	1.096E-04
	Mean	1.538E-03	1.042E-03	2.849E-04	1.516E-04	1.140E-04
	Med.	1.469E-03	1.036E-03	2.864E-04	1.518E-04	1.140E-04
	S.D.	1.973E-04	1.065E-04	1.620E-05	2.532E-06	2.028E-06
KSP504	Max.	3.344E-03	2.078E-03	3.566E-04	9.880E-05	9.686E-05
	Min.	1.848E-03	1.252E-03	2.981E-04	9.216E-05	8.455E-05
	Mean	2.571E-03	1.576E-03	3.203E-04	9.534E-05	8.983E-05
	Med.	2.605E-03	1.539E-03	3.201E-04	9.507E-05	8.999E-05
	S.D.	3.175E-04	2.080E-04	1.465E-05	1.932E-06	3.098E-06
KSP754	Max.	4.050E-03	2.592E-03	4.415E-04	7.332E-05	8.556E-05
	Min.	2.639E-03	1.637E-03	3.426E-04	6.442E-05	7.660E-05
	Mean	3.173E-03	2.017E-03	4.009E-04	6.864E-05	8.047E-05
	Med.	3.099E-03	1.971E-03	4.021E-04	6.866E-05	8.027E-05
	S.D.	4.121E-04	2.646E-04	2.471E-05	1.931E-06	2.478E-06

Table B.4: Detailed results of the inverted Generational distance I_{IGD} (Ch. 4)

Inst.	Stat.	Algorithms				
		NSGAII	SPEA2	MOEAD	GRASPM	HEMH
KSP252	Max.	1.117E-02	1.135E-02	1.075E-03	4.923E-04	4.931E-04
	Min.	4.945E-03	7.062E-03	5.981E-04	2.799E-04	2.131E-04
	Mean	7.899E-03	8.608E-03	8.094E-04	3.468E-04	3.161E-04
	Med.	7.866E-03	8.623E-03	8.029E-04	3.296E-04	2.921E-04
	S.D.	1.280E-03	9.324E-04	1.078E-04	5.493E-05	7.437E-05
KSP502	Max.	9.713E-03	9.792E-03	1.004E-03	2.781E-04	2.509E-04
	Min.	7.057E-03	7.878E-03	6.489E-04	2.311E-04	1.228E-04
	Mean	8.438E-03	8.595E-03	8.236E-04	2.467E-04	1.717E-04
	Med.	8.385E-03	8.529E-03	8.098E-04	2.428E-04	1.633E-04
	S.D.	6.428E-04	4.465E-04	9.653E-05	1.133E-05	3.123E-05
KSP752	Max.	9.136E-03	8.825E-03	7.622E-04	2.246E-04	1.714E-04
	Min.	7.455E-03	7.074E-03	4.661E-04	1.880E-04	1.161E-04
	Mean	8.295E-03	8.126E-03	5.864E-04	2.055E-04	1.378E-04
	Med.	8.238E-03	8.213E-03	5.845E-04	2.066E-04	1.350E-04
	S.D.	4.265E-04	4.790E-04	6.343E-05	9.935E-06	1.529E-05
KSP253	Max.	1.237E-03	1.412E-03	2.283E-04	1.011E-04	8.882E-05
	Min.	7.728E-04	9.801E-04	1.701E-04	9.630E-05	8.327E-05
	Mean	1.007E-03	1.153E-03	1.921E-04	9.910E-05	8.606E-05
	Med.	9.951E-04	1.159E-03	1.909E-04	9.930E-05	8.618E-05
	S.D.	1.030E-04	9.227E-05	1.473E-05	1.313E-06	1.384E-06
KSP503	Max.	1.128E-03	1.263E-03	2.100E-04	9.393E-05	7.525E-05
	Min.	9.057E-04	9.969E-04	1.477E-04	8.705E-05	7.004E-05
	Mean	1.028E-03	1.143E-03	1.791E-04	9.015E-05	7.263E-05
	Med.	1.020E-03	1.166E-03	1.796E-04	9.018E-05	7.272E-05
	S.D.	6.349E-05	7.278E-05	1.255E-05	1.966E-06	1.340E-06
KSP753	Max.	1.124E-03	1.211E-03	1.834E-04	8.325E-05	6.535E-05
	Min.	9.681E-04	1.027E-03	1.439E-04	7.850E-05	5.890E-05
	Mean	1.045E-03	1.124E-03	1.673E-04	8.099E-05	6.300E-05
	Med.	1.052E-03	1.130E-03	1.659E-04	8.103E-05	6.313E-05
	S.D.	4.279E-05	3.856E-05	9.302E-06	1.204E-06	1.438E-06
KSP254	Max.	4.190E-04	4.612E-04	1.151E-04	8.340E-05	7.474E-05
	Min.	3.441E-04	3.623E-04	1.005E-04	8.034E-05	7.093E-05
	Mean	3.838E-04	4.127E-04	1.075E-04	8.232E-05	7.264E-05
	Med.	3.832E-04	4.124E-04	1.071E-04	8.238E-05	7.264E-05
	S.D.	1.683E-05	2.491E-05	3.966E-06	7.525E-07	9.043E-07
KSP504	Max.	4.255E-04	4.187E-04	1.140E-04	7.803E-05	6.330E-05
	Min.	3.635E-04	3.638E-04	9.692E-05	7.524E-05	5.983E-05
	Mean	3.899E-04	3.968E-04	1.037E-04	7.686E-05	6.203E-05
	Med.	3.878E-04	3.989E-04	1.034E-04	7.695E-05	6.213E-05
	S.D.	1.407E-05	1.518E-05	3.757E-06	6.674E-07	7.498E-07
KSP754	Max.	4.260E-04	4.302E-04	1.169E-04	7.132E-05	5.790E-05
	Min.	3.766E-04	3.914E-04	9.931E-05	6.965E-05	5.489E-05
	Mean	4.040E-04	4.081E-04	1.084E-04	7.057E-05	5.611E-05
	Med.	4.043E-04	4.064E-04	1.079E-04	7.058E-05	5.609E-05
	S.D.	1.279E-05	1.020E-05	5.263E-06	3.188E-07	7.786E-07

Table B.5: Detailed results of the Maximum Spread indicator I_{MS} (Ch. 4)

Inst.	Stat.	Algorithms				
		NSGAII	**SPEA2**	**MOEAD**	**GRASPM**	**HEMH**
KSP252	**Max.**	7.251E-01	5.757E-01	1.446E+00	1.452E+00	1.432E+00
	Min.	3.294E-01	3.074E-01	1.307E+00	1.295E+00	1.314E+00
	Mean	5.168E-01	4.705E-01	1.373E+00	1.374E+00	1.360E+00
	Med.	5.100E-01	4.643E-01	1.369E+00	1.367E+00	1.356E+00
	S.D.	8.844E-02	6.244E-02	4.005E-02	4.533E-02	2.986E-02
KSP502	**Max.**	4.802E-01	4.328E-01	1.382E+00	1.450E+00	1.425E+00
	Min.	2.769E-01	2.840E-01	1.238E+00	1.325E+00	1.312E+00
	Mean	3.788E-01	3.678E-01	1.309E+00	1.393E+00	1.371E+00
	Med.	3.826E-01	3.718E-01	1.304E+00	1.389E+00	1.365E+00
	S.D.	4.890E-02	3.511E-02	2.957E-02	3.208E-02	3.078E-02
KSP752	**Max.**	3.239E-01	3.627E-01	1.356E+00	1.402E+00	1.386E+00
	Min.	1.871E-01	2.056E-01	1.285E+00	1.336E+00	1.322E+00
	Mean	2.598E-01	2.736E-01	1.317E+00	1.367E+00	1.354E+00
	Med.	2.664E-01	2.693E-01	1.315E+00	1.369E+00	1.355E+00
	S.D.	3.565E-02	3.868E-02	1.740E-02	1.716E-02	1.649E-02
KSP253	**Max.**	1.046E+00	8.647E-01	1.698E+00	1.735E+00	1.715E+00
	Min.	7.342E-01	6.212E-01	1.610E+00	1.666E+00	1.642E+00
	Mean	8.916E-01	7.604E-01	1.650E+00	1.702E+00	1.677E+00
	Med.	8.985E-01	7.631E-01	1.653E+00	1.703E+00	1.679E+00
	S.D.	7.537E-02	5.273E-02	2.518E-02	1.848E-02	1.774E-02
KSP503	**Max.**	7.563E-01	6.336E-01	1.704E+00	1.736E+00	1.745E+00
	Min.	5.819E-01	4.894E-01	1.603E+00	1.686E+00	1.671E+00
	Mean	6.653E-01	5.536E-01	1.653E+00	1.708E+00	1.703E+00
	Med.	6.641E-01	5.488E-01	1.651E+00	1.708E+00	1.703E+00
	S.D.	4.855E-02	4.251E-02	2.425E-02	1.157E-02	1.463E-02
KSP753	**Max.**	5.481E-01	4.685E-01	1.683E+00	1.755E+00	1.748E+00
	Min.	3.980E-01	3.243E-01	1.616E+00	1.699E+00	1.681E+00
	Mean	4.758E-01	3.851E-01	1.644E+00	1.725E+00	1.713E+00
	Med.	4.774E-01	3.826E-01	1.640E+00	1.722E+00	1.713E+00
	S.D.	4.469E-02	3.035E-02	1.958E-02	1.665E-02	1.647E-02
KSP254	**Max.**	1.409E+00	1.100E+00	1.946E+00	2.012E+00	1.989E+00
	Min.	1.124E+00	9.098E-01	1.853E+00	1.932E+00	1.896E+00
	Mean	1.234E+00	9.954E-01	1.903E+00	1.981E+00	1.944E+00
	Med.	1.226E+00	9.853E-01	1.910E+00	1.981E+00	1.945E+00
	S.D.	7.022E-02	5.198E-02	2.511E-02	1.964E-02	2.503E-02
KSP504	**Max.**	1.153E+00	9.137E-01	1.938E+00	2.020E+00	2.011E+00
	Min.	9.762E-01	7.053E-01	1.839E+00	1.961E+00	1.934E+00
	Mean	1.066E+00	7.832E-01	1.902E+00	1.985E+00	1.975E+00
	Med.	1.066E+00	7.820E-01	1.906E+00	1.981E+00	1.977E+00
	S.D.	5.056E-02	4.412E-02	2.216E-02	1.333E-02	1.819E-02
KSP754	**Max.**	9.144E-01	6.423E-01	1.890E+00	1.988E+00	1.994E+00
	Min.	7.653E-01	5.135E-01	1.796E+00	1.925E+00	1.924E+00
	Mean	8.273E-01	5.803E-01	1.838E+00	1.960E+00	1.958E+00
	Med.	8.235E-01	5.840E-01	1.836E+00	1.953E+00	1.953E+00
	S.D.	3.196E-02	3.765E-02	2.146E-02	1.628E-02	1.855E-02

Table B.6: Detailed results of the Coverage I_c indicator (Median) (Ch. 5)

Coverage Indicator I_c	**Test Instances**								
	KSP252	**KSP502**	**KSP752**	**KSP253**	**KSP503**	**KSP753**	**KSP254**	**KSP504**	**KSP754**
I_c(SPEA2,MOEAD)	9.202E-02	2.469E-02	0.000E+00	9.592E-03	3.802E-03	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD,SPEA2)	6.615E-01	8.212E-01	1.000E+00	5.350E-01	4.720E-01	6.967E-01	3.640E-01	3.500E-01	3.200E-01
I_c(SPEA2,MOEAD$_{de}$)	1.490E-01	8.223E-02	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD$_{de}$,SPEA2)	5.605E-01	6.160E-01	1.000E+00	9.100E-01	9.960E-01	1.000E+00	6.640E-01	9.300E-01	9.943E-01
I_c(SPEA2,MOEAD$_{pr}$)	3.774E-02	1.465E-02	1.424E-02	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD$_{pr}$,SPEA2)	8.032E-01	8.630E-01	8.491E-01	8.050E-01	7.360E-01	9.333E-01	5.740E-01	8.533E-01	8.371E-01
I_c(SPEA2,MOEAD$_{dp1}$)	1.046E-01	2.850E-02	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD$_{dp1}$,SPEA2)	6.564E-01	8.435E-01	1.000E+00	9.250E-01	9.960E-01	1.000E+00	6.920E-01	9.767E-01	9.971E-01
I_c(SPEA2,MOEAD$_{dp2}$)	1.245E-01	6.345E-02	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_c(MOEAD$_{dp2}$,SPEA2)	5.500E-01	6.994E-01	1.000E+00	9.300E-01	9.960E-01	1.000E+00	7.000E-01	9.767E-01	1.000E+00
I_c(MOEAD,MOEAD$_{de}$)	6.468E-01	7.806E-01	4.059E-01	9.009E-03	7.092E-03	9.063E-03	3.676E-03	0.000E+00	0.000E+00
I_c(MOEAD$_{de}$,MOEAD)	2.145E-01	8.860E-02	4.056E-01	4.212E-01	6.067E-01	6.855E-01	1.585E-01	3.701E-01	6.113E-01
I_c(MOEAD,MOEAD$_{pr}$)	2.130E-01	1.333E-01	1.808E-01	1.563E-02	1.316E-02	6.757E-03	8.333E-03	0.000E+00	0.000E+00
I_c(MOEAD$_{pr}$,MOEAD)	6.308E-01	7.466E-01	6.914E-01	3.704E-01	4.453E-01	5.965E-01	1.259E-01	3.639E-01	4.610E-01
I_c(MOEAD,MOEAD$_{dp1}$)	5.409E-01	6.997E-01	3.916E-01	8.850E-03	3.534E-03	8.955E-03	3.534E-03	0.000E+00	0.000E+00
I_c(MOEAD$_{dp1}$,MOEAD)	2.806E-01	1.612E-01	4.318E-01	4.631E-01	6.841E-01	7.481E-01	1.899E-01	5.170E-01	7.583E-01
I_c(MOEAD,MOEAD$_{dp2}$)	5.919E-01	7.718E-01	3.471E-01	8.439E-03	3.759E-03	9.259E-03	3.559E-03	0.000E+00	0.000E+00
I_c(MOEAD$_{dp2}$,MOEAD)	2.438E-01	1.033E-01	4.773E-01	4.771E-01	6.667E-01	7.540E-01	1.879E-01	4.983E-01	7.690E-01
I_c(MOEAD$_{de}$,MOEAD$_{pr}$)	1.176E-01	4.016E-02	2.430E-01	1.015E-01	1.818E-01	1.568E-01	4.472E-02	2.235E-02	5.530E-02
I_c(MOEAD$_{pr}$,MOEAD$_{de}$)	7.698E-01	8.939E-01	6.044E-01	7.759E-02	8.348E-02	1.391E-01	2.405E-02	3.769E-02	3.356E-02
I_c(MOEAD$_{de}$,MOEAD$_{dp1}$)	3.672E-01	3.468E-01	3.292E-01	7.033E-02	5.396E-02	5.178E-02	2.612E-02	1.042E-02	8.909E-03
I_c(MOEAD$_{dp1}$,MOEAD$_{de}$)	4.698E-01	5.267E-01	4.928E-01	1.176E-01	1.480E-01	1.574E-01	4.553E-02	8.057E-02	9.204E-02
I_c(MOEAD$_{de}$,MOEAD$_{dp2}$)	4.311E-01	4.590E-01	2.671E-01	6.667E-02	5.968E-02	5.075E-02	2.509E-02	1.096E-02	6.452E-03
I_c(MOEAD$_{dp2}$,MOEAD$_{de}$)	4.120E-01	4.255E-01	5.518E-01	1.245E-01	1.378E-01	1.538E-01	4.514E-02	7.371E-02	9.761E-02
I_c(MOEAD$_{pr}$,MOEAD$_{dp1}$)	6.748E-01	8.388E-01	5.878E-01	5.442E-02	4.727E-02	1.032E-01	1.468E-02	1.241E-02	1.215E-02
I_c(MOEAD$_{dp1}$,MOEAD$_{pr}$)	1.696E-01	7.216E-02	2.615E-01	1.212E-01	2.257E-01	2.022E-01	5.501E-02	5.217E-02	1.067E-01
I_c(MOEAD$_{pr}$,MOEAD$_{dp2}$)	7.385E-01	8.744E-01	5.625E-01	5.106E-02	5.729E-02	1.126E-01	1.527E-02	1.295E-02	1.072E-02
I_c(MOEAD$_{dp2}$,MOEAD$_{pr}$)	1.315E-01	4.819E-02	2.697E-01	1.269E-01	2.190E-01	2.054E-01	5.579E-02	4.794E-02	1.152E-01
I_c(MOEAD$_{dp1}$,MOEAD$_{dp2}$)	4.729E-01	5.533E-01	3.550E-01	8.190E-02	9.735E-02	8.946E-02	3.214E-02	3.235E-02	2.559E-02
I_c(MOEAD$_{dp2}$,MOEAD$_{dp1}$)	3.719E-01	3.254E-01	4.505E-01	9.211E-02	8.362E-02	9.422E-02	3.333E-02	2.513E-02	3.516E-02

Table B.7: Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 5)

Inst.	Stat.	Algorithms					
		SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	**Max.**	5.318E-01	6.050E-02	3.002E-01	4.449E-02	4.177E-01	3.081E-01
	Min.	3.407E-01	2.726E-02	6.865E-02	1.255E-02	6.032E-02	8.009E-02
	Mean	4.455E-01	4.069E-02	1.476E-01	2.962E-02	1.407E-01	1.433E-01
	Med.	4.525E-01	4.084E-02	1.454E-01	3.116E-02	1.251E-01	1.321E-01
	S.D.	5.222E-02	9.318E-03	5.081E-02	8.030E-03	6.951E-02	5.735E-02
KSP502	**Max.**	8.219E-01	5.805E-02	3.131E-01	5.239E-02	3.528E-01	3.374E-01
	Min.	5.323E-01	3.490E-02	7.901E-02	1.763E-02	6.990E-02	6.354E-02
	Mean	6.508E-01	4.756E-02	1.715E-01	3.061E-02	1.685E-01	1.697E-01
	Med.	6.385E-01	4.793E-02	1.542E-01	3.098E-02	1.707E-01	1.495E-01
	S.D.	5.121E-02	6.684E-03	6.261E-02	7.408E-03	5.920E-02	6.454E-02
KSP752	**Max.**	7.954E-01	5.275E-02	2.134E-01	3.890E-02	1.902E-01	2.270E-01
	Min.	6.849E-01	2.660E-02	5.594E-02	1.634E-02	6.669E-02	7.014E-02
	Mean	7.320E-01	3.968E-02	1.194E-01	2.859E-02	1.130E-01	1.114E-01
	Med.	7.321E-01	3.908E-02	1.125E-01	2.732E-02	1.080E-01	9.934E-02
	S.D.	2.912E-02	5.903E-03	3.703E-02	5.832E-03	2.909E-02	3.592E-02
KSP253	**Max.**	1.752E+00	2.341E-01	2.386E-01	1.618E-01	2.742E-01	2.652E-01
	Min.	1.421E+00	1.869E-01	1.254E-01	1.213E-01	1.181E-01	1.181E-01
	Mean	1.582E+00	2.093E-01	1.802E-01	1.404E-01	1.644E-01	1.666E-01
	Med.	1.557E+00	2.099E-01	1.794E-01	1.417E-01	1.603E-01	1.589E-01
	S.D.	9.482E-02	1.068E-02	2.984E-02	9.520E-03	3.303E-02	3.030E-02
KSP503	**Max.**	2.271E+00	2.868E-01	2.934E-01	1.701E-01	2.706E-01	2.640E-01
	Min.	2.017E+00	2.261E-01	1.406E-01	1.100E-01	1.315E-01	1.215E-01
	Mean	2.113E+00	2.499E-01	2.117E-01	1.402E-01	1.861E-01	1.937E-01
	Med.	2.113E+00	2.517E-01	2.087E-01	1.439E-01	1.790E-01	1.896E-01
	S.D.	6.484E-02	1.434E-02	3.862E-02	1.238E-02	3.819E-02	4.150E-02
KSP753	**Max.**	2.479E+00	3.149E-01	4.899E-01	1.898E-01	4.069E-01	3.945E-01
	Min.	2.310E+00	2.595E-01	1.840E-01	8.832E-02	1.617E-01	1.542E-01
	Mean	2.407E+00	2.846E-01	2.803E-01	1.213E-01	2.619E-01	2.526E-01
	Med.	2.407E+00	2.852E-01	2.521E-01	1.214E-01	2.500E-01	2.508E-01
	S.D.	4.348E-02	1.559E-02	7.967E-02	1.994E-02	6.326E-02	5.118E-02
KSP254	**Max.**	4.219E+00	9.091E-01	7.524E-01	6.732E-01	7.031E-01	7.608E-01
	Min.	3.800E+00	7.450E-01	5.394E-01	5.745E-01	5.174E-01	5.469E-01
	Mean	4.002E+00	8.336E-01	6.491E-01	6.112E-01	6.100E-01	6.102E-01
	Med.	4.004E+00	8.317E-01	6.518E-01	6.069E-01	6.069E-01	6.099E-01
	S.D.	1.233E-01	4.693E-02	4.734E-02	2.407E-02	4.694E-02	5.151E-02
KSP504	**Max.**	5.251E+00	1.119E+00	8.203E-01	5.321E-01	6.837E-01	7.018E-01
	Min.	4.766E+00	1.002E+00	5.529E-01	4.569E-01	4.815E-01	4.656E-01
	Mean	5.050E+00	1.056E+00	6.550E-01	4.920E-01	5.663E-01	5.656E-01
	Med.	5.040E+00	1.063E+00	6.485E-01	4.944E-01	5.446E-01	5.621E-01
	S.D.	1.114E-01	3.031E-02	6.270E-02	1.846E-02	6.193E-02	5.935E-02
KSP754	**Max.**	5.939E+00	1.359E+00	8.623E-01	6.278E-01	8.075E-01	7.911E-01
	Min.	5.542E+00	1.210E+00	6.057E-01	4.441E-01	4.790E-01	4.916E-01
	Mean	5.763E+00	1.288E+00	7.500E-01	5.070E-01	6.092E-01	5.788E-01
	Med.	5.764E+00	1.287E+00	7.564E-01	5.046E-01	6.011E-01	5.522E-01
	S.D.	1.015E-01	3.583E-02	6.046E-02	3.854E-02	6.978E-02	7.238E-02

Table B.8: Detailed results of the Generational distance indicator I_{GD} (Ch. 5)

Inst.	Stat.	Algorithms					
		SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	**Max.**	3.748E-03	1.905E-03	5.970E-03	1.664E-03	9.152E-03	6.650E-03
	Min.	1.662E-03	1.096E-03	8.868E-04	7.937E-04	4.860E-04	9.528E-04
	Mean	2.788E-03	1.395E-03	2.254E-03	1.117E-03	2.224E-03	2.327E-03
	Med.	2.901E-03	1.413E-03	2.046E-03	1.091E-03	1.653E-03	1.980E-03
	S.D.	6.424E-04	2.049E-04	1.116E-03	1.928E-04	1.655E-03	1.352E-03
KSP502	**Max.**	8.454E-03	2.047E-03	5.078E-03	1.789E-03	5.416E-03	5.430E-03
	Min.	2.268E-03	1.137E-03	7.823E-04	6.432E-04	4.252E-04	4.916E-04
	Mean	3.977E-03	1.528E-03	2.339E-03	1.096E-03	2.183E-03	2.357E-03
	Med.	3.909E-03	1.515E-03	2.243E-03	1.047E-03	2.091E-03	2.034E-03
	S.D.	1.258E-03	2.017E-04	1.159E-03	2.929E-04	1.091E-03	1.186E-03
KSP752	**Max.**	6.567E-03	1.576E-03	3.005E-03	1.826E-03	2.463E-03	2.798E-03
	Min.	2.622E-03	9.311E-04	6.044E-04	8.018E-04	4.188E-04	5.883E-04
	Mean	4.615E-03	1.277E-03	1.377E-03	1.233E-03	1.267E-03	1.243E-03
	Med.	4.591E-03	1.293E-03	1.295E-03	1.201E-03	1.239E-03	1.152E-03
	S.D.	8.286E-04	1.610E-04	6.039E-04	2.754E-04	4.740E-04	5.606E-04
KSP253	**Max.**	4.521E-03	2.412E-03	1.004E-03	1.338E-03	1.014E-03	1.074E-03
	Min.	2.791E-03	1.555E-03	5.824E-04	7.052E-04	5.290E-04	4.821E-04
	Mean	3.710E-03	1.880E-03	7.393E-04	9.769E-04	6.552E-04	6.303E-04
	Med.	3.715E-03	1.850E-03	7.109E-04	9.432E-04	6.338E-04	5.898E-04
	S.D.	3.756E-04	1.680E-04	1.212E-04	1.316E-04	1.138E-04	1.253E-04
KSP503	**Max.**	4.772E-03	2.388E-03	1.083E-03	1.620E-03	9.580E-04	8.978E-04
	Min.	3.253E-03	1.845E-03	4.582E-04	8.939E-04	4.171E-04	3.993E-04
	Mean	4.055E-03	2.133E-03	7.124E-04	1.237E-03	5.929E-04	6.079E-04
	Med.	4.022E-03	2.154E-03	7.008E-04	1.219E-03	5.477E-04	5.769E-04
	S.D.	4.473E-04	1.577E-04	1.532E-04	1.741E-04	1.415E-04	1.564E-04
KSP753	**Max.**	5.093E-03	2.412E-03	1.848E-03	1.963E-03	1.281E-03	1.138E-03
	Min.	3.161E-03	1.646E-03	5.023E-04	6.964E-04	3.440E-04	3.124E-04
	Mean	4.180E-03	2.100E-03	8.055E-04	1.047E-03	6.549E-04	6.369E-04
	Med.	4.143E-03	2.108E-03	6.756E-04	9.896E-04	6.241E-04	6.486E-04
	S.D.	4.641E-04	1.518E-04	3.305E-04	2.753E-04	2.108E-04	1.823E-04
KSP254	**Max.**	5.919E-03	2.760E-03	1.204E-03	1.632E-03	1.003E-03	9.995E-04
	Min.	4.592E-03	1.958E-03	8.584E-04	1.121E-03	7.645E-04	7.602E-04
	Mean	5.174E-03	2.375E-03	9.844E-04	1.333E-03	8.759E-04	8.608E-04
	Med.	5.136E-03	2.377E-03	9.730E-04	1.361E-03	8.636E-04	8.505E-04
	S.D.	3.724E-04	1.989E-04	8.002E-05	1.185E-04	6.831E-05	6.681E-05
KSP504	**Max.**	7.383E-03	3.548E-03	1.060E-03	1.258E-03	7.939E-04	8.340E-04
	Min.	5.224E-03	2.836E-03	8.333E-04	7.444E-04	4.863E-04	5.124E-04
	Mean	6.345E-03	3.182E-03	9.583E-04	1.018E-03	5.975E-04	6.175E-04
	Med.	6.403E-03	3.173E-03	9.472E-04	1.019E-03	5.794E-04	6.117E-04
	S.D.	6.138E-04	1.663E-04	6.640E-05	1.299E-04	8.075E-05	6.993E-05
KSP754	**Max.**	7.880E-03	4.079E-03	1.216E-03	2.156E-03	9.110E-04	8.094E-04
	Min.	5.680E-03	3.327E-03	8.349E-04	8.301E-04	4.825E-04	4.505E-04
	Mean	6.749E-03	3.690E-03	9.731E-04	1.394E-03	6.080E-04	5.506E-04
	Med.	6.728E-03	3.702E-03	9.499E-04	1.327E-03	5.930E-04	5.341E-04
	S.D.	4.891E-04	2.032E-04	9.070E-05	2.864E-04	1.001E-04	7.967E-05

Table B.9: Detailed results of the inverted generational distance I_{IGD} (Ch. 5)

Inst.	Stat.	Algorithms					
		SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	**Max.**	1.308E-02	1.405E-03	4.218E-03	1.177E-03	6.107E-03	4.496E-03
	Min.	8.295E-03	7.725E-04	1.194E-03	5.911E-04	1.238E-03	1.175E-03
	Mean	1.094E-02	1.001E-03	2.310E-03	8.373E-04	2.397E-03	2.253E-03
	Med.	1.113E-02	9.677E-04	2.255E-03	8.109E-04	2.042E-03	2.098E-03
	S.D.	1.412E-03	1.685E-04	7.278E-04	1.391E-04	1.086E-03	8.431E-04
KSP502	**Max.**	1.446E-02	1.319E-03	3.680E-03	1.042E-03	4.469E-03	3.944E-03
	Min.	9.317E-03	7.389E-04	7.384E-04	5.022E-04	8.030E-04	6.635E-04
	Mean	1.157E-02	9.725E-04	1.892E-03	7.049E-04	1.885E-03	1.888E-03
	Med.	1.147E-02	9.644E-04	1.815E-03	7.018E-04	1.875E-03	1.720E-03
	S.D.	1.126E-03	1.368E-04	7.859E-04	1.420E-04	7.652E-04	8.199E-04
KSP752	**Max.**	1.429E-02	9.765E-04	2.403E-03	9.891E-04	2.026E-03	2.367E-03
	Min.	1.169E-02	6.011E-04	5.810E-04	4.850E-04	7.723E-04	7.501E-04
	Mean	1.272E-02	7.963E-04	1.290E-03	7.041E-04	1.200E-03	1.164E-03
	Med.	1.258E-02	7.981E-04	1.257E-03	6.680E-04	1.142E-03	1.067E-03
	S.D.	6.852E-04	8.925E-05	4.196E-04	1.296E-04	3.090E-04	3.809E-04
KSP253	**Max.**	2.638E-03	7.180E-04	5.069E-04	5.350E-04	5.098E-04	5.086E-04
	Min.	1.875E-03	5.622E-04	4.259E-04	4.815E-04	4.057E-04	4.051E-04
	Mean	2.237E-03	6.106E-04	4.475E-04	4.995E-04	4.384E-04	4.348E-04
	Med.	2.243E-03	6.049E-04	4.418E-04	4.964E-04	4.314E-04	4.298E-04
	S.D.	1.959E-04	3.094E-05	1.852E-05	1.438E-05	2.055E-05	2.192E-05
KSP503	**Max.**	3.053E-03	6.580E-04	4.821E-04	5.284E-04	4.559E-04	4.496E-04
	Min.	2.556E-03	5.704E-04	3.705E-04	4.366E-04	3.684E-04	3.627E-04
	Mean	2.833E-03	6.123E-04	4.114E-04	4.821E-04	3.991E-04	4.008E-04
	Med.	2.825E-03	6.145E-04	4.109E-04	4.838E-04	3.916E-04	3.999E-04
	S.D.	1.325E-04	2.559E-05	2.355E-05	2.134E-05	2.427E-05	2.478E-05
KSP753	**Max.**	3.356E-03	6.524E-04	5.460E-04	5.748E-04	4.652E-04	4.389E-04
	Min.	2.917E-03	5.048E-04	3.305E-04	3.744E-04	3.212E-04	3.224E-04
	Mean	3.147E-03	5.932E-04	3.891E-04	4.270E-04	3.789E-04	3.703E-04
	Med.	3.144E-03	5.952E-04	3.800E-04	4.188E-04	3.703E-04	3.642E-04
	S.D.	8.229E-05	3.026E-05	5.059E-05	4.128E-05	3.611E-05	2.902E-05
KSP254	**Max.**	1.616E-03	7.028E-04	5.733E-04	6.274E-04	5.922E-04	5.723E-04
	Min.	1.314E-03	6.519E-04	5.433E-04	5.933E-04	5.339E-04	5.304E-04
	Mean	1.455E-03	6.795E-04	5.543E-04	6.073E-04	5.534E-04	5.489E-04
	Med.	1.466E-03	6.784E-04	5.530E-04	6.051E-04	5.500E-04	5.486E-04
	S.D.	7.174E-05	1.678E-05	7.721E-06	8.953E-06	1.415E-05	1.129E-05
KSP504	**Max.**	1.848E-03	6.950E-04	4.807E-04	5.071E-04	4.622E-04	4.583E-04
	Min.	1.555E-03	6.228E-04	4.353E-04	4.665E-04	4.255E-04	4.248E-04
	Mean	1.686E-03	6.510E-04	4.565E-04	4.821E-04	4.406E-04	4.372E-04
	Med.	1.692E-03	6.503E-04	4.560E-04	4.820E-04	4.385E-04	4.364E-04
	S.D.	6.662E-05	1.543E-05	1.011E-05	8.329E-06	9.948E-06	8.849E-06
KSP754	**Max.**	2.027E-03	7.402E-04	4.618E-04	5.482E-04	4.352E-04	4.435E-04
	Min.	1.829E-03	6.444E-04	4.208E-04	4.471E-04	3.940E-04	3.908E-04
	Mean	1.913E-03	6.882E-04	4.400E-04	4.807E-04	4.137E-04	4.073E-04
	Med.	1.913E-03	6.878E-04	4.401E-04	4.770E-04	4.137E-04	4.066E-04
	S.D.	4.701E-05	2.391E-05	1.103E-05	2.129E-05	1.011E-05	9.965E-06

Table B.10: Detailed results of the $R3$ indicator I_{R3} (Ch. 5)

Inst.	Stat.	Algorithms					
		SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KSP252	**Max.**	5.620E-02	8.040E-03	3.069E-02	6.359E-03	4.740E-02	3.292E-02
	Min.	2.729E-02	3.829E-03	3.737E-03	2.324E-03	2.506E-03	3.957E-03
	Mean	4.276E-02	5.269E-03	1.080E-02	3.852E-03	1.095E-02	1.118E-02
	Med.	4.264E-02	5.253E-03	9.793E-03	3.868E-03	8.057E-03	9.217E-03
	S.D.	8.257E-03	8.906E-04	5.959E-03	8.723E-04	8.960E-03	7.194E-03
KSP502	**Max.**	1.173E-01	9.085E-03	3.206E-02	7.217E-03	3.924E-02	3.590E-02
	Min.	5.652E-02	4.798E-03	3.886E-03	3.000E-03	2.252E-03	2.123E-03
	Mean	7.976E-02	6.663E-03	1.402E-02	4.641E-03	1.331E-02	1.402E-02
	Med.	7.767E-02	6.600E-03	1.284E-02	4.592E-03	1.244E-02	1.127E-02
	S.D.	1.058E-02	1.038E-03	7.562E-03	1.144E-03	7.525E-03	7.955E-03
KSP752	**Max.**	1.120E-01	7.852E-03	1.926E-02	8.386E-03	1.551E-02	1.874E-02
	Min.	8.604E-02	4.389E-03	3.065E-03	3.563E-03	1.639E-03	2.965E-03
	Mean	9.610E-02	6.456E-03	8.113E-03	5.638E-03	7.231E-03	7.162E-03
	Med.	9.577E-02	6.471E-03	7.226E-03	5.472E-03	6.860E-03	6.379E-03
	S.D.	6.514E-03	8.864E-04	4.056E-03	1.169E-03	3.092E-03	3.755E-03
KSP253	**Max.**	7.031E-02	1.269E-02	8.000E-03	7.731E-03	8.001E-03	9.371E-03
	Min.	5.007E-02	9.638E-03	5.363E-03	5.647E-03	4.967E-03	4.847E-03
	Mean	6.068E-02	1.091E-02	6.286E-03	6.649E-03	5.819E-03	5.792E-03
	Med.	6.061E-02	1.095E-02	6.151E-03	6.633E-03	5.601E-03	5.603E-03
	S.D.	5.549E-03	7.884E-04	6.146E-04	4.666E-04	6.841E-04	9.135E-04
KSP503	**Max.**	1.109E-01	1.530E-02	1.022E-02	9.152E-03	8.552E-03	8.195E-03
	Min.	9.064E-02	1.168E-02	5.522E-03	5.665E-03	4.736E-03	4.793E-03
	Mean	9.741E-02	1.317E-02	7.081E-03	7.263E-03	5.909E-03	6.272E-03
	Med.	9.729E-02	1.330E-02	6.944E-03	7.185E-03	5.774E-03	6.141E-03
	S.D.	5.071E-03	7.779E-04	1.084E-03	7.550E-04	9.936E-04	1.030E-03
KSP753	**Max.**	1.280E-01	1.590E-02	1.581E-02	1.130E-02	1.219E-02	1.133E-02
	Min.	1.138E-01	1.255E-02	5.618E-03	5.071E-03	4.393E-03	4.268E-03
	Mean	1.215E-01	1.451E-02	8.344E-03	6.521E-03	7.248E-03	7.059E-03
	Med.	1.216E-01	1.452E-02	7.551E-03	6.163E-03	6.852E-03	7.281E-03
	S.D.	3.621E-03	7.520E-04	2.533E-03	1.229E-03	1.928E-03	1.645E-03
KSP254	**Max.**	8.006E-02	1.685E-02	1.132E-02	1.223E-02	1.100E-02	1.102E-02
	Min.	6.590E-02	1.392E-02	9.068E-03	1.013E-02	8.846E-03	8.693E-03
	Mean	7.239E-02	1.553E-02	1.031E-02	1.110E-02	9.660E-03	9.627E-03
	Max.	7.223E-02	1.569E-02	1.039E-02	1.113E-02	9.530E-03	9.495E-03
	S.D.	3.786E-03	8.313E-04	4.766E-04	5.810E-04	5.563E-04	5.609E-04
KSP504	**Max.**	1.129E-01	2.169E-02	1.237E-02	9.675E-03	1.073E-02	1.135E-02
	Min.	9.573E-02	1.923E-02	9.903E-03	8.317E-03	7.879E-03	7.768E-03
	Mean	1.058E-01	2.046E-02	1.081E-02	9.093E-03	8.931E-03	9.005E-03
	Med.	1.054E-01	2.038E-02	1.078E-02	9.161E-03	8.666E-03	8.889E-03
	S.D.	3.993E-03	6.233E-04	6.714E-04	3.160E-04	8.179E-04	7.112E-04
KSP754	**Max.**	1.446E-01	2.720E-02	1.350E-02	1.278E-02	1.152E-02	1.164E-02
	Min	1.288E-01	2.315E-02	1.028E-02	8.414E-03	7.780E-03	7.743E-03
	Mean	1.377E-01	2.540E-02	1.208E-02	1.018E-02	9.174E-03	8.808E-03
	Med.	1.375E-01	2.534E-02	1.210E-02	1.008E-02	8.899E-03	8.621E-03
	S.D.	4.173E-03	8.719E-04	7.780E-04	9.183E-04	9.378E-04	8.216E-04

Table B.11: Detailed results of the Coverage I_c indicator (Median) (Ch. 6)

Coverage Indicator I_c	**Test Instances**								
	KSP252	**KSP502**	**KSP752**	**KSP253**	**KSP503**	**KSP753**	**KSP254**	**KSP504**	**KSP754**
I_C(HEMH2,HEMH$_{de}$)	2.391E-01	2.184E-01	2.072E-01	7.287E-02	7.419E-02	8.205E-02	2.961E-02	2.625E-02	2.421E-02
I_C(HEMH$_{de}$,HEMH2)	2.426E-01	2.510E-01	2.148E-01	7.059E-02	6.646E-02	6.194E-02	2.894E-02	1.671E-02	1.259E-02
I_C(HEMH2,HEMH$_{pr}$)	5.704E-01	6.378E-01	6.102E-01	1.392E-01	1.007E-01	8.219E-02	4.965E-02	3.140E-02	2.444E-02
I_C(HEMH$_{pr}$,HEMH2)	1.091E-01	7.173E-02	3.802E-02	3.137E-02	3.797E-02	4.627E-02	1.307E-02	1.412E-02	1.323E-02
I_C(HEMH2,HEMH)	4.777E-01	6.971E-01	7.566E-01	1.256E-01	9.319E-02	7.808E-02	5.614E-02	2.597E-02	1.743E-02
I_C(HEMH,HEMH2)	2.143E-01	6.883E-02	3.536E-02	3.226E-02	2.903E-02	3.295E-02	6.667E-03	7.075E-03	5.650E-03
I_C(HEMH2,MOEAD)	9.820E-01	1.000E+00	1.000E+00	5.837E-01	7.981E-01	9.041E-01	2.478E-01	5.691E-01	8.205E-01
I_C(MOEAD,HEMH2)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_C(HEMH$_{de}$,HEMH$_{pr}$)	5.693E-01	6.485E-01	6.140E-01	1.366E-01	9.459E-02	7.418E-02	4.853E-02	2.619E-02	1.873E-02
I_C(HEMH$_{pr}$,HEMH$_{de}$)	1.165E-01	6.375E-02	3.529E-02	3.252E-02	4.101E-02	5.455E-02	1.307E-02	1.896E-02	1.869E-02
I_C(HEMH$_{de}$,HEMH)	4.836E-01	7.086E-01	7.600E-01	1.227E-01	8.791E-02	7.077E-02	5.535E-02	2.105E-02	1.313E-02
I_C(HEMH,HEMH$_{de}$)	2.212E-01	6.531E-02	3.187E-02	3.320E-02	3.195E-02	4.103E-02	6.944E-03	1.152E-02	7.533E-03
I_C(HEMH$_{de}$,MOEAD)	9.831E-01	1.000E+00	1.000E+00	5.787E-01	7.909E-01	8.936E-01	2.397E-01	5.355E-01	7.973E-01
I_C(MOEAD,HEMH$_{de}$)	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_C(HEMH$_{pr}$,HEMH)	2.929E-01	4.355E-01	5.877E-01	7.965E-02	7.326E-02	7.080E-02	3.759E-02	2.344E-02	1.677E-02
I_C(HEMH,HEMH$_{pr}$)	4.419E-01	2.634E-01	1.359E-01	8.072E-02	5.882E-02	4.845E-02	2.007E-02	1.467E-02	1.124E-02
I_C(HEMH$_{pr}$,MOEAD)	9.322E-01	1.000E+00	1.000E+00	4.976E-01	7.776E-01	8.988E-01	1.992E-01	5.542E-01	8.248E-01
I_C(MOEAD,HEMH$_{pr}$)	2.055E-02	0.000E+00	0.000E+00	4.444E-03	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
I_C(HEMH,MOEAD)	9.444E-01	9.880E-01	9.774E-01	4.933E-01	7.547E-01	8.742E-01	1.673E-01	5.236E-01	7.750E-01
I_C(MOEAD,HEMH)	1.987E-02	0.000E+00	0.000E+00	4.566E-03	0.000E+00	0.000E+00	3.676E-03	0.000E+00	0.000E+00

Table B.12: Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 6)

Inst.	Stat.	Algorithms				
		MOEA/D	**HEMH**	**HEMH$_{de}$**	**HEMH$_{pr}$**	**HEMH2**
KSP252	**Max.**	6.021E-02	1.494E-02	7.324E-03	1.095E-02	6.831E-03
	Min.	3.228E-02	7.413E-03	5.002E-03	7.871E-03	5.124E-03
	Mean	4.664E-02	1.024E-02	6.065E-03	9.328E-03	6.085E-03
	Med.	4.718E-02	9.902E-03	6.041E-03	9.374E-03	6.080E-03
	S.D.	7.220E-03	2.022E-03	5.871E-04	5.516E-04	3.885E-04
KSP502	**Max.**	6.743E-02	2.558E-02	1.105E-02	1.700E-02	9.554E-03
	Min.	4.294E-02	9.317E-03	3.198E-03	8.948E-03	3.376E-03
	Mean	5.666E-02	1.554E-02	5.568E-03	1.156E-02	5.557E-03
	Med.	5.542E-02	1.579E-02	5.264E-03	1.133E-02	5.232E-03
	S.D.	6.593E-03	2.969E-03	1.770E-03	1.809E-03	1.497E-03
KSP752	**Max.**	5.800E-02	2.120E-02	4.557E-03	8.001E-03	4.940E-03
	Min.	3.836E-02	1.230E-02	3.084E-03	6.251E-03	2.868E-03
	Mean	4.735E-02	1.639E-02	3.844E-03	7.344E-03	3.939E-03
	Med.	4.681E-02	1.611E-02	3.813E-03	7.407E-03	3.856E-03
	Med.	4.566E-03	2.143E-03	4.717E-04	3.826E-04	5.285E-04
KSP253	**Max.**	2.857E-01	1.413E-01	9.922E-02	1.211E-01	9.298E-02
	Min.	2.031E-01	1.065E-01	8.211E-02	9.786E-02	8.249E-02
	Mean	2.243E-01	1.209E-01	8.854E-02	1.094E-01	8.766E-02
	Med.	2.252E-01	1.210E-01	8.910E-02	1.096E-01	8.736E-02
	S.D.	1.675E-02	6.960E-03	3.594E-03	5.035E-03	2.206E-03
KSP503	**Max.**	3.034E-01	1.322E-01	7.725E-02	9.852E-02	7.848E-02
	Min.	2.495E-01	1.067E-01	6.907E-02	8.386E-02	6.921E-02
	Mean	2.755E-01	1.159E-01	7.388E-02	9.006E-02	7.393E-02
	Med.	2.749E-01	1.159E-01	7.407E-02	8.998E-02	7.371E-02
	S.D.	1.365E-02	5.455E-03	2.068E-03	3.131E-03	2.388E-03
KSP753	**Max.**	3.346E-01	9.951E-02	6.745E-02	8.273E-02	6.891E-02
	Min.	2.556E-01	7.949E-02	6.254E-02	7.202E-02	5.863E-02
	Mean	2.891E-01	8.854E-02	6.480E-02	7.633E-02	6.392E-02
	Med.	2.868E-01	8.842E-02	6.475E-02	7.554E-02	6.391E-02
	S.D.	1.900E-02	4.021E-03	1.185E-03	2.445E-03	2.321E-03
KSP254	**Max.**	9.020E-01	5.928E-01	4.471E-01	5.162E-01	4.544E-01
	Min.	7.855E-01	5.184E-01	4.022E-01	4.636E-01	4.064E-01
	Mean	8.565E-01	5.548E-01	4.256E-01	4.904E-01	4.268E-01
	Med.	8.551E-01	5.594E-01	4.256E-01	4.896E-01	4.247E-01
	S.D.	2.337E-02	1.839E-02	1.123E-02	1.135E-02	1.209E-02
KSP504	**Max.**	1.121E+00	4.715E-01	4.055E-01	4.304E-01	4.052E-01
	Min.	9.843E-01	4.361E-01	3.795E-01	3.869E-01	3.659E-01
	Mean	1.067E+00	4.533E-01	3.955E-01	4.097E-01	3.844E-01
	Med.	1.074E+00	4.552E-01	3.964E-01	4.080E-01	3.839E-01
	S.D.	2.779E-02	1.012E-02	6.406E-03	1.067E-02	9.509E-03
KSP754	**Max.**	1.300E+00	4.145E-01	3.687E-01	3.726E-01	3.642E-01
	Min.	1.158E+00	3.615E-01	3.438E-01	3.504E-01	3.240E-01
	Mean	1.226E+00	3.934E-01	3.540E-01	3.644E-01	3.407E-01
	Med.	1.228E+00	3.965E-01	3.525E-01	3.670E-01	3.398E-01
	S.D.	3.220E-02	1.322E-02	6.451E-03	6.808E-03	8.409E-03

Table B.13: Detailed results of the Generational distance indicator I_{GD} (Ch. 6)

Inst.	Stat.	Algorithms				
		MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	Max.	1.897E-03	6.878E-04	4.263E-04	6.836E-04	3.858E-04
	Min.	1.117E-03	3.982E-04	2.690E-04	4.463E-04	2.821E-04
	Mean	1.496E-03	5.172E-04	3.399E-04	5.740E-04	3.373E-04
	Med.	1.439E-03	5.207E-04	3.323E-04	5.738E-04	3.297E-04
	S.D.	2.089E-04	6.227E-05	4.033E-05	4.528E-05	2.759E-05
KSP502	Max.	1.771E-03	7.116E-04	2.347E-04	3.697E-04	2.332E-04
	Min.	1.271E-03	3.199E-04	1.102E-04	2.665E-04	1.250E-04
	Mean	1.534E-03	4.411E-04	1.441E-04	3.119E-04	1.538E-04
	Med.	1.540E-03	4.206E-04	1.369E-04	3.099E-04	1.504E-04
	S.D.	1.338E-04	7.166E-05	2.750E-05	2.865E-05	2.434E-05
KSP752	Max.	1.666E-03	5.630E-04	8.311E-05	2.105E-04	8.511E-05
	Min.	9.312E-04	3.811E-04	6.842E-05	1.726E-04	6.800E-05
	Mean	1.330E-03	4.775E-04	7.516E-05	1.914E-04	7.809E-05
	Med.	1.313E-03	4.830E-04	7.515E-05	1.903E-04	7.804E-05
	S.D.	1.778E-04	4.004E-05	3.507E-06	1.099E-05	4.203E-06
KSP253	Max.	2.366E-03	7.851E-04	4.666E-04	7.650E-04	4.984E-04
	Min.	1.673E-03	6.093E-04	3.434E-04	6.174E-04	3.345E-04
	Mean	1.982E-03	6.833E-04	3.922E-04	6.823E-04	3.879E-04
	Med.	1.975E-03	6.894E-04	3.908E-04	6.791E-04	3.828E-04
	S.D.	1.809E-04	4.018E-05	2.829E-05	3.693E-05	3.694E-05
KSP503	Max.	2.577E-03	6.025E-04	3.018E-04	4.724E-04	2.953E-04
	Min.	1.886E-03	4.171E-04	2.501E-04	3.900E-04	2.437E-04
	Mean	2.253E-03	4.954E-04	2.756E-04	4.246E-04	2.701E-04
	Med.	2.267E-03	4.953E-04	2.761E-04	4.217E-04	2.708E-04
	S.D.	1.434E-04	4.085E-05	1.390E-05	2.077E-05	1.244E-05
KSP753	Max.	2.704E-03	4.159E-04	2.341E-04	3.070E-04	2.576E-04
	Min.	1.835E-03	3.190E-04	2.028E-04	2.572E-04	1.789E-04
	Mean	2.163E-03	3.633E-04	2.168E-04	2.766E-04	2.071E-04
	Med.	2.174E-03	3.595E-04	2.155E-04	2.789E-04	2.060E-04
	S.D.	1.873E-04	2.611E-05	9.193E-06	1.252E-05	1.384E-05
KSP254	Max.	2.795E-03	1.347E-03	6.585E-04	1.042E-03	6.715E-04
	Min.	2.250E-03	1.031E-03	5.658E-04	8.295E-04	5.605E-04
	Mean	2.524E-03	1.140E-03	6.069E-04	9.133E-04	6.137E-04
	Med.	2.507E-03	1.130E-03	6.016E-04	9.109E-04	6.130E-04
	S.D.	1.405E-04	7.578E-05	2.699E-05	4.798E-05	3.048E-05
KSP504	Max.	3.928E-03	7.390E-04	4.927E-04	5.699E-04	4.371E-04
	Min.	3.174E-03	5.822E-04	3.728E-04	4.721E-04	3.110E-04
	Mean	3.455E-03	6.628E-04	4.082E-04	5.142E-04	3.621E-04
	Med.	3.434E-03	6.632E-04	4.055E-04	5.166E-04	3.594E-04
	S.D.	1.700E-04	3.349E-05	2.509E-05	2.434E-05	2.487E-05
KSP754	Max.	4.064E-03	5.387E-04	3.082E-04	3.809E-04	2.812E-04
	Min.	3.437E-03	4.463E-04	2.569E-04	3.324E-04	2.299E-04
	Mean	3.788E-03	4.907E-04	2.876E-04	3.558E-04	2.525E-04
	Med.	3.788E-03	4.901E-04	2.894E-04	3.518E-04	2.522E-04
	S.D.	1.403E-04	2.640E-05	1.311E-05	1.401E-05	1.282E-05

Table B.14: Detailed results of Inverted Generational distance I_{IGD} (Ch. 6)

Inst.	Stat.	Algorithms				
		MOEA/D	**HEMH**	**HEMH$_{de}$**	**HEMH$_{pr}$**	**HEMH2**
KSP252	**Max.**	1.228E-03	7.532E-04	3.910E-04	5.079E-04	3.892E-04
	Min.	7.499E-04	3.797E-04	3.102E-04	3.957E-04	3.223E-04
	Mean	9.318E-04	4.827E-04	3.485E-04	4.510E-04	3.501E-04
	Med.	9.100E-04	4.391E-04	3.471E-04	4.512E-04	3.508E-04
	S.D.	1.217E-04	8.776E-05	2.021E-05	2.216E-05	1.691E-05
KSP502	**Max.**	8.747E-04	3.360E-04	2.191E-04	2.671E-04	2.159E-04
	Min.	6.092E-04	2.225E-04	1.112E-04	1.806E-04	1.173E-04
	Mean	7.308E-04	2.717E-04	1.311E-04	2.096E-04	1.325E-04
	Med.	7.179E-04	2.655E-04	1.245E-04	2.051E-04	1.284E-04
	S.D.	6.640E-05	3.047E-05	2.363E-05	1.931E-05	1.726E-05
KSP752	**Max.**	6.536E-04	3.206E-04	8.775E-05	1.206E-04	8.372E-05
	Min.	4.176E-04	2.012E-04	7.410E-05	1.072E-04	7.586E-05
	Mean	5.413E-04	2.435E-04	7.886E-05	1.141E-04	7.952E-05
	Med.	5.464E-04	2.395E-04	7.834E-05	1.141E-04	7.973E-05
	S.D.	6.291E-05	2.740E-05	2.883E-06	3.843E-06	2.133E-06
KSP253	**Max.**	7.126E-04	4.639E-04	3.933E-04	4.642E-04	4.118E-04
	Min.	5.801E-04	4.264E-04	3.699E-04	4.256E-04	3.718E-04
	Mean	6.311E-04	4.485E-04	3.828E-04	4.408E-04	3.845E-04
	Med.	6.250E-04	4.468E-04	3.819E-04	4.393E-04	3.838E-04
	S.D.	3.393E-05	9.077E-06	5.635E-06	8.951E-06	7.950E-06
KSP503	**Max.**	5.604E-04	3.457E-04	2.672E-04	3.090E-04	2.687E-04
	Min.	4.891E-04	3.224E-04	2.513E-04	2.891E-04	2.541E-04
	Mean	5.282E-04	3.327E-04	2.578E-04	2.987E-04	2.601E-04
	Med.	5.281E-04	3.311E-04	2.581E-04	2.985E-04	2.596E-04
	S.D.	1.869E-05	6.802E-06	3.773E-06	4.749E-06	3.509E-06
KSP753	**Max.**	5.230E-04	2.519E-04	1.974E-04	2.325E-04	2.021E-04
	Min.	4.080E-04	2.369E-04	1.884E-04	2.177E-04	1.884E-04
	Mean	4.554E-04	2.465E-04	1.933E-04	2.237E-04	1.947E-04
	Med.	4.580E-04	2.471E-04	1.931E-04	2.233E-04	1.948E-04
	S.D.	2.511E-05	3.920E-06	2.502E-06	4.130E-06	3.335E-06
KSP254	**Max.**	6.973E-04	5.909E-04	5.027E-04	5.472E-04	5.053E-04
	Min.	6.389E-04	5.514E-04	4.777E-04	5.218E-04	4.768E-04
	Mean	6.746E-04	5.693E-04	4.883E-04	5.327E-04	4.906E-04
	Med.	6.713E-04	5.688E-04	4.883E-04	5.319E-04	4.903E-04
	S.D.	1.436E-05	9.560E-06	6.426E-06	5.985E-06	7.154E-06
KSP504	**Max.**	6.431E-04	4.114E-04	3.549E-04	3.793E-04	3.540E-04
	Min.	5.698E-04	3.930E-04	3.336E-04	3.578E-04	3.377E-04
	Mean	5.946E-04	4.042E-04	3.463E-04	3.671E-04	3.458E-04
	Med.	5.915E-04	4.047E-04	3.464E-04	3.667E-04	3.461E-04
	S.D.	1.673E-05	4.424E-06	4.250E-06	4.945E-06	3.893E-06
KSP754	**Max.**	5.717E-04	3.349E-04	2.771E-04	2.932E-04	2.753E-04
	Min.	5.139E-04	3.213E-04	2.649E-04	2.791E-04	2.641E-04
	Mean	5.457E-04	3.276E-04	2.711E-04	2.846E-04	2.704E-04
	Med.	5.466E-04	3.283E-04	2.711E-04	2.845E-04	2.706E-04
	S.D.	1.498E-05	3.772E-06	2.756E-06	2.930E-06	2.816E-06

Table B.15: Detailed results of R3 indicator I_{R3} (Ch. 6)

Inst.	Stat.	Algorithms				
		MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KSP252	Max.	7.752E-03	2.255E-03	1.593E-03	2.858E-03	1.587E-03
	Min.	4.251E-03	1.595E-03	9.857E-04	1.873E-03	9.430E-04
	Mean	6.049E-03	1.880E-03	1.293E-03	2.321E-03	1.299E-03
	Med.	5.963E-03	1.886E-03	1.300E-03	2.324E-03	1.300E-03
	S.D.	8.222E-04	1.622E-04	1.425E-04	1.945E-04	1.495E-04
KSP502	Max.	8.721E-03	2.581E-03	7.981E-04	1.848E-03	8.199E-04
	Min.	5.713E-03	1.632E-03	6.239E-04	1.457E-03	6.401E-04
	Mean	7.303E-03	2.065E-03	6.885E-04	1.621E-03	7.115E-04
	Med.	7.236E-03	2.042E-03	6.830E-04	1.624E-03	7.048E-04
	S.D.	7.212E-04	1.789E-04	5.115E-05	9.450E-05	4.259E-05
KSP752	Max.	8.726E-03	2.887E-03	5.850E-04	1.319E-03	5.998E-04
	Min.	4.826E-03	2.194E-03	5.016E-04	1.079E-03	5.144E-04
	Mean	6.879E-03	2.470E-03	5.458E-04	1.210E-03	5.541E-04
	Med.	6.808E-03	2.475E-03	5.452E-04	1.204E-03	5.524E-04
	S.D.	9.836E-04	1.534E-04	2.047E-05	5.805E-05	2.086E-05
KSP253	Max.	1.238E-02	6.091E-03	4.838E-03	5.740E-03	4.795E-03
	Min.	1.003E-02	5.296E-03	4.201E-03	4.958E-03	4.241E-03
	Mean	1.112E-02	5.619E-03	4.490E-03	5.245E-03	4.481E-03
	Med.	1.108E-02	5.585E-03	4.508E-03	5.237E-03	4.478E-03
	S.D.	5.664E-04	2.031E-04	1.402E-04	1.859E-04	1.407E-04
KSP503	Max.	1.453E-02	5.696E-03	4.017E-03	4.958E-03	4.058E-03
	Min.	1.207E-02	4.720E-03	3.575E-03	4.295E-03	3.510E-03
	Mean	1.336E-02	5.161E-03	3.827E-03	4.604E-03	3.783E-03
	Med.	1.338E-02	5.145E-03	3.836E-03	4.604E-03	3.782E-03
	S.D.	6.632E-04	2.261E-04	1.255E-04	1.545E-04	1.206E-04
KSP753	Max.	1.565E-02	4.636E-03	3.576E-03	4.232E-03	3.701E-03
	Min.	1.192E-02	4.024E-03	3.207E-03	3.691E-03	3.146E-03
	Mean	1.424E-02	4.251E-03	3.381E-03	3.924E-03	3.320E-03
	Med.	1.436E-02	4.229E-03	3.366E-03	3.916E-03	3.311E-03
	S.D.	8.822E-04	1.478E-04	7.930E-05	1.381E-04	1.183E-04
KSP254	Max.	1.728E-02	1.025E-02	8.223E-03	9.222E-03	8.303E-03
	Min.	1.489E-02	8.879E-03	7.225E-03	8.232E-03	7.419E-03
	Mean	1.606E-02	9.704E-03	7.864E-03	8.841E-03	7.836E-03
	Med.	1.600E-02	9.678E-03	7.867E-03	8.859E-03	7.784E-03
	S.D.	5.778E-04	3.335E-04	2.196E-04	2.258E-04	2.406E-04
KSP504	Max.	2.105E-02	8.498E-03	7.632E-03	7.904E-03	7.338E-03
	Min.	1.869E-02	7.398E-03	6.778E-03	7.002E-03	6.748E-03
	Mean	2.024E-02	7.910E-03	7.182E-03	7.396E-03	7.063E-03
	Med.	2.027E-02	7.887E-03	7.194E-03	7.358E-03	7.064E-03
	S.D.	5.432E-04	2.411E-04	1.780E-04	2.014E-04	1.483E-04
KSP754	Max.	2.513E-02	7.625E-03	6.244E-03	6.548E-03	6.062E-03
	Min.	2.263E-02	6.535E-03	5.875E-03	5.774E-03	5.550E-03
	Mean	2.395E-02	7.022E-03	6.021E-03	6.260E-03	5.822E-03
	Med.	2.397E-02	6.920E-03	6.039E-03	6.282E-03	5.809E-03
	S.D.	5.792E-04	2.860E-04	1.013E-04	1.818E-04	1.288E-04

Table B.16: Detailed results of the Coverage I_C indicator (Median) (Ch. 7)

Coverage Indicator I_C	**Test Instances**										
	Fonseca	**Kursawe**	**ZDT1**	**ZDT2**	**ZDT3**	**ZDT4**	**ZDT6**	**DTLZ2**	**DTLZ4**	**DTLZ6**	**DTLZ7**
I_C(HESSA,dMOPSO)	$4.70e-01$	$5.87e-01$	$8.74e-01$	$9.52e-01$	$8.28e-01$	$1.70e-01$	$9.09e-02$	$7.66e-01$	$1.18e-01$	$6.67e-03$	$5.19e-01$
I_C(dMOPSO,HESSA)	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$1.00e-02$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$6.54e-03$	$0.00e+00$
I_C(HESSA,MOEAD$_2$)	$3.00e-02$	$9.30e-02$	$1.00e+00$	$1.00e+00$	$9.83e-01$	$1.00e+00$	$6.72e-01$	$1.28e-01$	$1.58e-01$	$4.24e-03$	$9.06e-01$
I_C(MOEAD$_2$,HESSA)	$2.10e-01$	$6.90e-02$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$8.37e-03$	$8.26e-03$	$2.05e-03$	$0.00e+00$
I_C(HESSA,MOEAD$_1$)	$1.80e-01$	$1.53e-01$	$6.67e-01$	$5.88e-02$	$4.60e-01$	$9.89e-01$	$9.90e-01$	$0.00e+00$	$0.00e+00$	$9.96e-01$	$1.73e-02$
I_C(MOEAD$_1$,HESSA)	$3.00e-02$	$5.81e-02$	$0.00e+00$	$1.00e-02$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$6.30e-02$	$4.13e-02$	$0.00e+00$	$1.22e-01$
I_C(dMOPSO,MOEAD$_2$)	$0.00e+00$	$1.15e-02$	$1.00e+00$	$1.00e+00$	$1.00e+00$	$9.00e-01$	$6.54e-01$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$9.07e-01$
I_C(MOEAD$_2$,dMOPSO)	$4.95e-01$	$5.63e-01$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$7.96e-01$	$2.39e-01$	$3.33e-03$	$0.00e+00$
I_C(dMOPSO,MOEAD$_1$)	$2.00e-02$	$1.16e-02$	$4.08e-02$	$3.03e-02$	$1.61e-02$	$8.87e-01$	$7.07e-01$	$0.00e+00$	$0.00e+00$	$8.92e-01$	$0.00e+00$
I_C(MOEAD$_1$,dMOPSO)	$4.20e-01$	$5.80e-01$	$5.16e-01$	$1.70e-01$	$6.72e-01$	$0.00e+00$	$2.00e-02$	$6.87e-01$	$1.01e-01$	$0.00e+00$	$4.61e-01$
I_C(MOEAD$_2$,MOEAD$_1$)	$2.50e-01$	$6.98e-02$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$1.08e-01$	$0.00e+00$	$0.00e+00$	$1.00e+00$	$0.00e+00$
I_C(MOEAD$_1$,MOEAD$_2$)	$0.00e+00$	$8.05e-02$	$9.81e-01$	$6.67e-01$	$9.80e-01$	$9.09e-01$	$6.54e-01$	$1.44e-01$	$1.15e-01$	$0.00e+00$	$7.71e-01$

Table B.17: Detailed results of referenced hypervolume indicator I_{Rhyp} (Ch. 7)

Inst.	Stat.	Algorithms			
		MOEA/D$_1$	MOEA/D$_2$	dMOPSO	HESSA
Fonseca	Max.	8.010E-03	3.738E-03	1.600E-02	4.403E-03
	Min.	4.523E-03	3.566E-03	9.968E-03	3.892E-03
	Mean	5.034E-03	3.636E-03	1.191E-02	4.081E-03
	Med.	4.831E-03	3.637E-03	1.177E-02	4.042E-03
	S.D.	6.643E-04	4.354E-05	1.408E-03	1.152E-04
Kursawe	Max.	3.236E-01	4.951E-01	2.066E+00	3.047E-01
	Min.	2.655E-01	3.303E-01	1.127E+00	2.585E-01
	Mean	2.935E-01	3.834E-01	1.494E+00	2.770E-01
	Med.	2.922E-01	3.853E-01	1.470E+00	2.765E-01
	S.D.	1.832E-02	3.812E-02	2.181E-01	1.293E-02
ZDT1	Max.	7.309E-02	5.705E-01	2.842E-02	5.958E-03
	Min.	1.152E-02	2.385E-01	1.910E-02	5.102E-03
	Mean	2.919E-02	4.502E-01	2.355E-02	5.496E-03
	Med.	2.492E-02	4.688E-01	2.377E-02	5.454E-03
	S.D.	1.531E-02	9.024E-02	2.569E-03	2.044E-04
ZDT2	Max.	2.963E-01	3.328E-01	3.328E-01	6.789E-03
	Min.	7.875E-03	3.328E-01	6.557E-03	4.748E-03
	Mean	1.871E-01	3.328E-01	1.230E-01	5.480E-03
	Med.	2.062E-01	3.328E-01	3.971E-02	5.434E-03
	S.D.	8.272E-02	2.823E-16	1.400E-01	4.734E-04
ZDT3	Max.	9.093E-02	7.072E-01	4.414E-02	6.493E-03
	Min.	8.700E-03	4.429E-01	1.682E-02	5.039E-03
	Mean	2.278E-02	6.221E-01	3.093E-02	5.654E-03
	Med.	1.258E-02	6.329E-01	3.103E-02	5.644E-03
	S.D.	2.365E-02	6.436E-02	6.268E-03	2.926E-04
ZDT4	Max.	3.629E-01	6.662E-01	4.402E-01	6.450E-03
	Min.	3.487E-02	6.662E-01	7.502E-03	4.832E-03
	Mean	1.050E-01	6.662E-01	1.007E-01	5.528E-03
	Med.	7.629E-02	6.662E-01	5.774E-02	5.532E-03
	S.D.	7.603E-02	1.129E-16	1.177E-01	4.412E-04
ZDT6	Max.	2.084E-02	2.659E-01	1.849E-02	2.727E-04
	Min.	1.088E-02	2.718E-04	3.427E-03	2.063E-04
	Mean	1.537E-02	1.324E-01	8.761E-03	2.160E-04
	Med.	1.448E-02	1.378E-01	8.067E-03	2.130E-04
	S.D.	2.622E-03	8.931E-02	3.686E-03	1.256E-05
DTLZ2	Max.	4.853E-02	5.204E-02	1.370E-01	4.844E-02
	Min.	4.400E-02	4.666E-02	1.096E-01	4.387E-02
	Mean	4.610E-02	4.839E-02	1.209E-01	4.644E-02
	Med.	4.622E-02	4.830E-02	1.210E-01	4.667E-02
	S.D.	1.344E-03	1.093E-03	6.858E-03	1.099E-03
DTLZ4	Max.	4.196E-01	4.349E-02	6.444E-02	3.489E-02
	Min.	3.049E-02	2.564E-02	3.350E-02	2.629E-02
	Mean	1.744E-01	2.873E-02	4.539E-02	3.019E-02
	Med.	1.573E-01	2.793E-02	4.471E-02	2.994E-02
	S.D.	1.272E-01	4.128E-03	6.556E-03	2.224E-03
DTLZ6	Max.	2.721E-02	2.860E-04	1.307E-03	3.167E-04
	Min.	1.512E-04	2.481E-04	5.223E-04	3.037E-04
	Mean	1.105E-02	2.667E-04	7.711E-04	3.108E-04
	Med.	1.082E-02	2.674E-04	7.416E-04	3.113E-04
	S.D.	7.024E-03	7.691E-06	1.631E-04	3.462E-06
DTLZ7	Max.	2.777E-01	7.238E-01	2.064E-01	1.764E-01
	Min.	1.564E-01	3.290E-01	1.275E-01	1.549E-01
	Mean	1.710E-01	5.359E-01	1.666E-01	1.692E-01
	Med.	1.661E-01	5.476E-01	1.623E-01	1.709E-01
	S.D.	2.168E-02	1.246E-01	2.035E-02	5.296E-03

Table B.18: Detailed results of the generational distance indicator I_{GD} (Ch. 7)

Inst.	Stat.	Algorithms			
		MOEA/D$_1$	MOEA/D$_2$	dMOPSO	HESSA
Fonseca	Max.	2.596E-03	1.057E-03	5.470E-03	1.304E-03
	Min.	1.321E-03	9.167E-04	3.650E-03	1.008E-03
	Mean	1.565E-03	9.962E-04	4.491E-03	1.135E-03
	Med.	1.515E-03	9.982E-04	4.457E-03	1.132E-03
	S.D.	2.480E-04	3.309E-05	5.200E-04	5.914E-05
Kursawe	Max.	1.182E-02	1.571E-02	7.051E-02	8.893E-03
	Min.	6.488E-03	7.653E-03	3.691E-02	6.436E-03
	Mean.	8.561E-03	1.089E-02	4.892E-02	7.460E-03
	Med.	8.430E-03	1.073E-02	4.702E-02	7.343E-03
	S.D.	1.256E-03	1.634E-03	8.805E-03	6.696E-04
ZDT1	Max.	9.072E-03	5.429E-01	1.646E-02	1.080E-03
	Min.	2.804E-03	1.531E-01	8.954E-03	6.334E-04
	Mean	5.149E-03	3.674E-01	1.288E-02	8.304E-04
	Med.	4.955E-03	3.892E-01	1.289E-02	8.115E-04
	S.D.	1.405E-03	9.837E-02	1.805E-03	1.240E-04
ZDT2	Max.	6.852E-03	7.787E-01	3.134E-02	1.747E-03
	Min.	2.259E-04	1.665E-01	0.000E+00	5.174E-04
	Mean	1.065E-03	4.582E-01	1.346E-02	9.119E-04
	Med.	4.474E-04	4.711E-01	1.685E-02	9.020E-04
	S.D.	1.443E-03	1.600E-01	1.141E-02	2.874E-04
ZDT3	Max.	3.435E-02	6.905E-01	1.233E-02	2.998E-03
	Min.	2.811E-03	2.347E-01	5.708E-03	2.366E-03
	Mean	5.832E-03	5.057E-01	8.874E-03	2.697E-03
	Med.	4.070E-03	5.129E-01	9.154E-03	2.710E-03
	S.D.	6.653E-03	1.022E-01	1.635E-03	1.613E-04
ZDT4	Max.	3.505E-01	3.685E+01	6.545E-03	1.292E-03
	Min.	1.004E-02	6.718E+00	7.607E-04	4.422E-04
	Mean	7.155E-02	1.754E+01	1.871E-03	8.383E-04
	Med.	4.303E-02	1.624E+01	1.360E-03	7.952E-04
	S.D.	8.253E-02	6.550E+00	1.327E-03	2.435E-04
ZDT6	Max.	1.825E-02	7.097E-01	4.024E-02	2.689E-03
	Min.	9.670E-03	2.412E-03	1.463E-03	2.558E-03
	Mean	1.345E-02	2.152E-01	5.453E-03	2.625E-03
	Med.	1.280E-02	1.848E-01	2.501E-03	2.633E-03
	S.D.	2.098E-03	1.980E-01	9.171E-03	3.745E-05
DTLZ2	Max.	6.069E-03	8.335E-03	8.320E-02	6.703E-03
	Min.	5.656E-03	7.203E-03	5.854E-02	5.980E-03
	Mean	5.850E-03	7.847E-03	6.851E-02	6.331E-03
	Med.	5.843E-03	7.871E-03	6.804E-02	6.308E-03
	S.D.	1.116E-04	2.426E-04	5.035E-03	1.776E-04
DTLZ4	Max.	3.459E-02	3.971E-02	6.267E-02	3.948E-02
	Min.	1.123E-08	2.782E-02	3.328E-02	3.150E-02
	Mean	2.438E-02	3.688E-02	4.615E-02	3.535E-02
	Med.	2.363E-02	3.720E-02	4.481E-02	3.516E-02
	S.D.	9.510E-03	2.545E-03	7.019E-03	1.772E-03
DTLZ6	Max.	1.166E-01	3.856E-03	4.886E-03	3.973E-03
	Min.	3.586E-03	3.529E-03	3.989E-03	3.719E-03
	Mean	4.189E-02	3.717E-03	4.423E-03	3.855E-03
	Med.	3.824E-02	3.718E-03	4.455E-03	3.852E-03
	S.D.	2.823E-02	8.566E-05	2.321E-04	6.941E-05
DTLZ7	Max.	2.359E-02	3.820E-01	6.076E-02	2.232E-02
	Min.	1.353E-02	7.373E-02	3.472E-02	2.091E-02
	Mean	2.246E-02	1.692E-01	5.200E-02	2.187E-02
	Med.	2.269E-02	1.461E-01	5.324E-02	2.201E-02
	S.D.	1.754E-03	7.546E-02	6.344E-03	4.044E-04

Table B.19: Detailed results of inverted generational distance I_{IGD} (Ch. 7)

Inst.	Stat.	Algorithms			
		MOEA/D$_1$	MOEA/D$_2$	dMOPSO	HESSA
Fonseca	Max.	5.980E-03	3.627E-03	3.101E-02	3.826E-03
	Min.	3.835E-03	3.534E-03	8.443E-03	3.618E-03
	Mean	4.212E-03	3.574E-03	1.627E-02	3.699E-03
	Med	4.038E-03	3.574E-03	1.525E-02	3.687E-03
	S.D.	4.590E-04	2.246E-05	5.371E-03	6.034E-05
Kursawe	Max.	4.669E-02	4.753E-02	1.591E-01	4.370E-02
	Min.	4.111E-02	4.235E-02	8.171E-02	4.145E-02
	Mean	4.238E-02	4.416E-02	1.059E-01	4.204E-02
	Med.	4.197E-02	4.422E-02	9.781E-02	4.195E-02
	S.D.	1.185E-03	1.244E-03	2.139E-02	5.275E-04
ZDT1	Max.	1.214E-01	5.462E-01	1.779E-02	4.266E-03
	Min.	8.070E-03	1.832E-01	1.210E-02	3.942E-03
	Mean	3.871E-02	3.904E-01	1.484E-02	4.053E-03
	Med.	2.755E-02	4.014E-01	1.508E-02	4.043E-03
	S.D.	3.208E-02	9.745E-02	1.565E-03	7.240E-05
ZDT2	Max.	4.354E-01	1.210E+00	6.096E-01	4.389E-03
	Min.	4.837E-03	5.494E-01	4.918E-03	3.844E-03
	Mean	2.457E-01	8.981E-01	1.953E-01	4.000E-03
	Med.	2.667E-01	8.789E-01	2.312E-02	3.988E-03
	S.D.	1.266E-01	1.528E-01	2.737E-01	1.199E-04
ZDT3	Max.	1.203E-01	5.960E-01	2.146E-02	1.088E-02
	Min.	1.096E-02	2.884E-01	1.143E-02	1.047E-02
	Mean	2.675E-02	4.664E-01	1.745E-02	1.064E-02
	Med.	1.434E-02	4.755E-01	1.772E-02	1.061E-02
	S.D.	2.768E-02	7.729E-02	2.416E-03	1.081E-04
ZDT4	Max.	3.073E-01	1.311E+01	6.306E-01	4.386E-03
	Min.	2.338E-02	1.538E+00	5.433E-03	3.903E-03
	Mean	1.055E-01	6.483E+00	1.602E-01	4.106E-03
	Med.	8.396E-02	6.197E+00	1.055E-01	4.096E-03
	S.D.	6.942E-02	2.804E+00	1.819E-01	1.272E-04
ZDT6	Max.	2.615E-02	7.416E-01	7.206E-03	1.913E-03
	Min.	1.337E-02	1.856E-03	2.343E-03	1.846E-03
	Mean	1.935E-02	1.635E-01	3.626E-03	1.890E-03
	Med.	1.888E-02	4.484E-02	3.250E-03	1.892E-03
	S.D.	3.426E-03	2.260E-01	1.105E-03	1.632E-05
DTLZ2	Max.	3.763E-02	3.881E-02	8.504E-02	3.780E-02
	Min.	3.680E-02	3.736E-02	6.456E-02	3.686E-02
	Mean	3.718E-02	3.808E-02	7.331E-02	3.727E-02
	Med.	3.719E-02	3.808E-02	7.426E-02	3.728E-02
	S.D.	2.027E-04	3.433E-04	4.707E-03	2.356E-04
DTLZ4	Max.	4.196E-01	4.349E-02	6.444E-02	3.489E-02
	Min.	3.049E-02	2.564E-02	3.350E-02	2.629E-02
	Mean	1.744E-01	2.873E-02	4.539E-02	3.019E-02
	Med.	1.573E-01	2.793E-02	4.471E-02	2.994E-02
	S.D.	1.272E-01	4.128E-03	6.556E-03	2.224E-03
DTLZ6	Max.	1.095E-01	4.447E-03	9.986E-03	4.566E-03
	Min.	3.973E-03	4.322E-03	6.670E-03	4.500E-03
	Mean	3.820E-02	4.394E-03	7.717E-03	4.519E-03
	Med.	3.456E-02	4.399E-03	7.492E-03	4.518E-03
	S.D.	2.623E-02	3.271E-05	7.274E-04	1.547E-05
DTLZ7	Max.	7.880E-01	9.099E-01	3.602E-01	1.234E-01
	Min.	1.129E-01	1.390E-01	6.877E-02	1.109E-01
	Mean	2.241E-01	3.761E-01	8.938E-02	1.151E-01
	Med.	1.265E-01	3.659E-01	7.957E-02	1.136E-01
	S.D.	1.560E-01	1.817E-01	5.145E-02	3.549E-03

Table B.20: Detailed results of unary additive Epsilon indicator I_{ε_+} (Ch. 7)

Inst.	Stat.	Algorithms			
		MOEA/D$_1$	MOEA/D$_2$	dMOPSO	HESSA
Fonseca	Max.	1.531E-02	6.806E-03	1.288E-01	8.026E-03
	Min.	7.058E-03	6.262E-03	1.689E-02	6.544E-03
	Mean	9.089E-03	6.469E-03	7.447E-02	7.069E-03
	Med.	8.567E-03	6.440E-03	7.416E-02	7.052E-03
	S.D.	2.021E-03	1.442E-04	3.366E-02	3.811E-04
Kursawe	Max.	1.489E-01	1.388E-01	1.126E+00	1.204E-01
	Min.	7.276E-02	8.438E-02	2.614E-01	7.288E-02
	Mean	8.787E-02	1.085E-01	4.575E-01	8.238E-02
	Med.	8.234E-02	1.071E-01	3.826E-01	7.970E-02
	S.D.	1.616E-02	1.379E-02	1.961E-01	9.132E-03
ZDT1	Max.	2.510E-01	6.790E-01	3.593E-02	9.272E-03
	Min.	2.323E-02	2.514E-01	2.222E-02	7.568E-03
	Mean	1.103E-01	4.884E-01	2.878E-02	8.248E-03
	Med.	8.952E-02	4.942E-01	2.835E-02	8.211E-03
	S.D.	7.164E-02	1.200E-01	3.777E-03	4.188E-04
ZDT2	Max.	9.574E-01	1.771E+00	1.000E+00	8.342E-03
	Min.	1.281E-02	1.060E+00	1.095E-02	6.542E-03
	Mean	7.207E-01	1.420E+00	3.264E-01	7.438E-03
	Med.	8.220E-01	1.385E+00	4.341E-02	7.400E-03
	S.D.	2.632E-01	1.693E-01	4.486E-01	4.578E-04
ZDT3	Max.	6.664E-01	1.271E+00	7.513E-02	1.568E-02
	Min.	1.547E-02	5.155E-01	2.339E-02	1.364E-02
	Mean	1.265E-01	8.656E-01	4.638E-02	1.514E-02
	Med.	2.362E-02	8.879E-01	4.563E-02	1.526E-02
	S.D.	1.875E-01	1.682E-01	1.182E-02	4.300E-04
ZDT4	Max.	4.518E-01	1.344E+01	7.695E-01	1.126E-02
	Min.	5.329E-02	1.837E+00	1.286E-02	7.724E-03
	Mean	2.132E-01	6.801E+00	2.535E-01	8.910E-03
	Med.	1.751E-01	6.519E+00	2.383E-01	8.876E-03
	S.D.	1.046E-01	2.813E+00	2.300E-01	8.514E-04
ZDT6	Max.	3.778E-02	8.324E-01	4.838E-02	5.686E-03
	Min.	1.972E-02	4.869E-03	1.499E-02	4.752E-03
	Mean	2.846E-02	3.088E-01	2.990E-02	5.026E-03
	Med.	2.756E-02	2.949E-01	2.788E-02	4.934E-03
	S.D.	4.977E-03	2.404E-01	8.734E-03	2.361E-04
DTLZ2	Max.	9.210E-02	9.176E-02	1.187E-01	9.236E-02
	Min.	7.582E-02	6.657E-02	9.394E-02	7.437E-02
	Mean	8.762E-02	7.829E-02	1.075E-01	8.443E-02
	Med.	8.857E-02	7.880E-02	1.075E-01	8.536E-02
	S.D.	4.302E-03	8.235E-03	6.170E-03	4.832E-03
DTLZ4	Max.	9.778E-01	2.374E-01	2.495E-01	8.665E-02
	Min.	6.017E-02	5.144E-02	1.083E-01	4.934E-02
	Mean	4.818E-01	7.273E-02	1.490E-01	6.233E-02
	Med.	6.319E-01	6.022E-02	1.422E-01	6.067E-02
	S.D.	3.082E-01	4.488E-02	2.992E-02	8.695E-03
DTLZ6	Max.	9.990E-02	1.054E-02	1.875E-02	1.061E-02
	Min.	9.118E-03	9.236E-03	1.051E-02	9.912E-03
	Mean	4.304E-02	1.019E-02	1.468E-02	1.036E-02
	Med.	4.304E-02	1.024E-02	1.468E-02	1.036E-02
	S.D.	2.087E-02	2.675E-04	1.968E-03	1.535E-04
DTLZ7	Max.	2.535E+00	2.724E+00	1.349E+00	1.728E-01
	Min.	1.653E-01	3.995E-01	1.508E-01	1.557E-01
	Mean	6.159E-01	9.651E-01	2.493E-01	1.679E-01
	Med.	1.717E-01	7.649E-01	2.049E-01	1.685E-01
	S.D.	6.363E-01	6.142E-01	2.131E-01	4.013E-03

Bibliography

[Aarts & Lenstra 1997] Emile Aarts and Jan K. Lenstra, editeurs. Local Search in Combinatorial Optimization. John Wiley & Sons, Inc., New York, NY, USA, 1st édition, 1997. (Cited on pages 39, 43, 191 and 192.)

[Ahuja *et al.* 2002] Ravindra K Ahuja, Özlem Ergun, James B Orlin and Abraham P Punnen. *A survey of very large-scale neighborhood search techniques*. Discrete Applied Mathematics, vol. 123, no. 1, pages 75–102, 2002. (Cited on page 87.)

[Al-Yamani *et al.* 2002] Ahmad Al-Yamani, Sadiq M Sait, Habib Youssef and Hassan Barada. *Parallelizing tabu search on a cluster of heterogeneous workstations*. Journal of Heuristics, vol. 8, no. 3, pages 277–304, 2002. (Cited on page 89.)

[Alba 2005] Enrique Alba. Parallel metaheuristics: a new class of algorithms, volume 47. Wiley-Interscience, 2005. (Cited on page 55.)

[Alsheddy 2011] Abdullah Alsheddy. *Empowerment scheduling: a multi-objective optimization approach using Guided Local Search*. PhD thesis, School of Computer Science and Electronic Engineering, University of Essex, 2011. (Cited on page 53.)

[Bäck 1996] Thomas Bäck. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press on Demand, 1996. (Cited on page 58.)

[Baker 1987] James E. Baker. *Reducing bias and inefficiency in the selection algorithm*. In Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc. (Cited on pages 160 and 204.)

[Baluja 1994] Shumeet Baluja. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Rapport technique CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994. (Cited on page 39.)

[Bandyopadhyay *et al.* 2008] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik and K. Deb. *A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA*. IEEE Trans. Evolutionary Computation, vol. 12, no. 3, pages 269–283, 2008. (Cited on pages 16 and 109.)

[Banks *et al.* 2007] Alec Banks, Jonathan Vincent and Chukwudi Anyakoha. *A review of particle swarm optimization. Part I: background and development.* Natural Computing: an international journal, vol. 6, no. 4, pages 467–484, December 2007. (Cited on page 73.)

[Basseur *et al.* 2003] M. Basseur, F. Seynhaeve and E. G. Talbi. *Adaptive mechanisms for multi-objective evolutionary algorithms.* In Congress on Engineering in System Application CESA'03, pages 72–86, Lille, France, 2003. (Cited on pages 90 and 194.)

[Basseur *et al.* 2004] Matthieu Basseur, Julien Lemesre, Clarisse Dhaenens and El-Ghazali Talbi. *Cooperation between Branch and Bound and Evolutionary Approaches to Solve a Bi-objective Flow Shop Problem.* In CelsoC. Ribeiro and SimoneL. Martins, editeurs, Experimental and Efficient Algorithms, volume 3059 of *Lecture Notes in Computer Science*, pages 72–86. Springer Berlin Heidelberg, 2004. (Cited on pages 89 and 194.)

[Battiti & Protasi 1996] Roberto Battiti and Marco Protasi. *Reactive Search, a history-based heuristic for MAX-SAT.* ACM Journal of Experimental Algorithmics, vol. 2, 1996. (Cited on page 57.)

[Battiti & Tecchiolli 1994] Roberto Battiti and Giampietro Tecchiolli. *The reactive tabu search.* ORSA journal on computing, vol. 6, no. 2, pages 126–140, 1994. (Cited on page 56.)

[Bednorz 2008] Witold Bednorz. *Advances in greedy algorithms.* Wienna: I-Tech Education and Publishing KG, 2008. (Cited on pages 37 and 191.)

[Benayoun *et al.* 1971] R Benayoun, J De Montgolfier, Jo Tergny and O Laritchev. *Linear programming with multiple objective functions: Step method (STEM).* Mathematical programming, vol. 1, no. 1, pages 366–375, 1971. (Cited on page 18.)

[Benlic & Hao 2013] Una Benlic and Jin-Kao Hao. *A Study of Adaptive Perturbation Strategy for Iterated Local Search.* In Martin Middendorf and Christian Blum, editeurs, EvoCOP, volume 7832 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2013. (Cited on page 50.)

[Bersini & Varela 1990] Hugues Bersini and Francisco J. Varela. *Hints for Adaptive Problem Solving Gleaned from Immune Network.* In Proceedings of the Workshop Parallel Problem Solving from Nature, LNCS 496, pages 343–354. Springer, 1990. (Cited on page 39.)

[Beyer & Schwefel 2002] Hans-Georg Beyer and Hans-Paul Schwefel. *Evolution strategies-A comprehensive introduction*. Natural Computing: an international journal, vol. 1, no. 1, pages 3–52, May 2002. (Cited on pages 55 and 63.)

[Bianchi *et al.* 2009] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella and Walter J Gutjahr. *A survey on metaheuristics for stochastic combinatorial optimization*. Natural Computing: an international journal, vol. 8, no. 2, pages 239–287, 2009. (Cited on page 58.)

[Bledsoe & Browning 1959] W. W. Bledsoe and I. Browning. *Pattern recognition and reading by machine*. In Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference, IRE-AIEE-ACM '59 (Eastern), pages 225–232, New York, NY, USA, 1959. ACM. (Cited on pages 37 and 191.)

[Blum & Roli 2001] Christian Blum and Andrea Roli. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. Rapport technique TR/IRIDIA/2001-13, IRIDIA, Belgium., 2001. (Cited on page 74.)

[Blum & Roli 2003] Christian Blum and Andrea Roli. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. ACM Computing Surveys, vol. 35, no. 3, pages 268–308, 2003. (Cited on pages 3, 40, 41, 43, 44, 92, 188 and 192.)

[Blum & Roli 2008] Christian Blum and Andrea Roli. *Hybrid metaheuristics: an introduction*. In Hybrid Metaheuristics, pages 1–30. Springer, 2008. (Cited on pages 40 and 192.)

[BLUM *et al.* 2005] CHRISTIAN BLUM, Andrea Roli and Enrique Alba. *An Introduction to Metaheuristic Techniques*. Parallel Metaheuristics: A New Class of Algorithms, vol. 47, page 1, 2005. (Cited on pages 85 and 194.)

[Boettcher & Percus 2002] Stefan Boettcher and Allon G Percus. *Optimization with extremal dynamics*. Complexity, vol. 8, no. 2, pages 57–62, 2002. (Cited on page 55.)

[Boussaïd *et al.* 2013] Ilhem Boussaïd, Julien Lepagnot and Patrick Siarry. *A survey on optimization metaheuristics*. Information Sciences, 2013. (Cited on pages 41 and 192.)

[Branke *et al.* 2008] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen and Roman Slowinski. Multiobjective optimization: Interactive and evolutionary approaches, volume 5252. Springer, 2008. (Cited on page 190.)

[Bremermann 1962] H. J. Bremermann. *Optimization through evolution and recombination.* In M. C. Yovits, G. T. Jacobi and G. D. Golstine, editeurs, Proceedings of the Conference on Self-Organizing Systems – 1962, pages 93–106, Washington, DC, 1962. Spartan Books. (Cited on pages 37 and 191.)

[Burke & Smith 2000] EK Burke and AJ Smith. *Hybrid evolutionary techniques for the maintenance scheduling problem.* Power Systems, IEEE Transactions on, vol. 15, no. 1, pages 122–128, 2000. (Cited on page 70.)

[Burke *et al.* 1998] Edmund K Burke, James P Newall and Rupert F Weare. *Initialization strategies and diversity in evolutionary timetabling.* Evolutionary computation, vol. 6, no. 1, pages 81–103, 1998. (Cited on page 43.)

[Calégari *et al.* 1999] Patrice Calégari, Giovanni Coray, Alain Hertz, Daniel Kobler and Pierre Kuonen. *A taxonomy of evolutionary algorithms in combinatorial optimization.* Journal of Heuristics, vol. 5, no. 2, pages 145–158, 1999. (Cited on page 85.)

[Canuto *et al.* 2001] S.A. Canuto, M.G.C. Resende and C.C. Ribeiro. *Local search with perturbation for the prize-collecting Steiner tree problems in graphs.* Networks, vol. 38, pages 50–58, 2001. (Cited on page 96.)

[Carrizosa *et al.* 2012] Emilio Carrizosa, Milan Dražić, Zorica Dražić and Nenad Mladenović. *Gaussian variable neighborhood search for continuous optimization.* Computers & Operations Research, vol. 39, no. 9, pages 2206–2213, 2012. (Cited on page 51.)

[Černỳ 1985] VLADIMÍR Černỳ. *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm.* Journal of optimization theory and applications, vol. 45, no. 1, pages 41–51, 1985. (Cited on pages 54 and 192.)

[Chakraborty 2008] Uday K Chakraborty. Advances in differential evolution, volume 143. Springer-Verlag New York Incorporated, 2008. (Cited on page 67.)

[Chakraborty 2010] U.K. Chakraborty. Advances in Differential Evolution. Studies in Computational Intelligence. Springer, 2010. (Cited on pages 139 and 205.)

[Chankong & Haimes 1983] V. Chankong and Y.Y. Haimes. Multiobjective Decision Making Theory and Methodology. Elsevier Science, New York, 1983. (Cited on pages 8 and 189.)

[Charnes & Cooper 1961] A. Charnes and W.W. Cooper. Management models and industrial applications of linear programming. Numeéro v. 1 de Management Models and Industrial Applications of Linear Programming. Wiley, 1961. (Cited on pages 31 and 190.)

[Charnes & Cooper 1977] A. Charnes and W.W. Cooper. *Goal programming and multiple objective optimizations: Part 1.* European Journal of Operational Research, vol. 1, no. 1, pages 39 – 54, 1977. (Cited on page 32.)

[Charnes *et al.* 1955] Abraham Charnes, William W Cooper and Robert O Ferguson. *Optimal estimation of executive compensation by linear programming.* Management science, vol. 1, no. 2, pages 138–151, 1955. (Cited on pages 31 and 190.)

[Chelouah & Siarry 2000] Rachid Chelouah and Patrick Siarry. *Tabu search applied to global optimization.* European Journal of Operational Research, vol. 123, no. 2, pages 256–270, 2000. (Cited on page 57.)

[Clerc & Kennedy 2002] M. Clerc and J. Kennedy. *The particle swarm - explosion, stability, and convergence in a multidimensional complex space.* Evolutionary Computation, IEEE Transactions on, vol. 6, no. 1, pages 58–73, 2002. (Cited on page 73.)

[Coello Coello & Lechuga 2002] Carlos A. Coello Coello and M.S. Lechuga. *MOPSO: a proposal for multiple objective particle swarm optimization.* In Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, volume 2, pages 1051–1056, 2002. (Cited on page 194.)

[Coello Coello 2002] Carlos 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, vol. 191, no. 11-12, pages 1245–1287, January 2002. (Cited on page 58.)

[Coello *et al.* 2006] Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen. Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. (Cited on pages 164, 188 and 207.)

[Coello *et al.* 2007] C.A. Coello Coello, G.L. Lamont and D.A. van Veldhuizen. Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation. Springer, Berlin, Heidelberg, 2nd édition, 2007. (Cited on pages 12, 13, 15, 58 and 80.)

[Coello 2000a] Carlos A. Coello. *An updated survey of GA-based multiobjective optimization techniques.* ACM Comput. Surv., vol. 32, no. 2, pages 109–143, June 2000. (Cited on page 58.)

[Coello 2000b] Carlos A. Coello. *An updated survey of GA-based multiobjective optimization techniques.* ACM Comput. Surv., vol. 32, no. 2, pages 109–143, June 2000. (Cited on page 92.)

[Corne & Ross 1996] David Corne and Peter Ross. *Peckish initialisation strategies for evolutionary timetabling.* In Edmund Burke and Peter Ross, editeurs, Practice and Theory of Automated Timetabling, volume 1153 of *Lecture Notes in Computer Science*, pages 227–240. Springer Berlin Heidelberg, 1996. (Cited on page 43.)

[Corne *et al.* 2000] David Corne, Joshua D. Knowles and Martin J. Oates. *The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation.* In Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, PPSN VI, pages 839–848, London, UK, UK, 2000. Springer-Verlag. (Cited on pages 79 and 194.)

[Cotta-Porras 1998] Carlos Cotta-Porras. *A study of hybridisation techniques and their application to the design of evolutionary algorithms.* AI Communications, vol. 11, no. 3, pages 223–224, 1998. (Cited on pages 85 and 194.)

[Cotta *et al.* 2005] C. Cotta, E g. Talbi and E. Alba. *Parallel hybrid metaheuristics.* In Parallel Metaheuristics, a New Class of Algorithms, pages 347–370. John Wiley, 2005. (Cited on page 86.)

[Courat *et al.* 1994] J.-P. Courat, G. Raynaud, I. Mrad and P. Siarry. *Electronic component model minimization based on log simulated annealing.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, vol. 41, no. 12, pages 790–795, 1994. (Cited on page 55.)

[Coyne & Paton 1994] John Coyne and Ray Paton. *Genetic algorithms and directed adaptation.* In Evolutionary Computing, pages 103–114. Springer, 1994. (Cited on pages 89 and 194.)

[Crainic & Toulouse 2003] TeodorGabriel Crainic and Michel Toulouse. *Parallel Strategies for Meta-Heuristics.* In Fred Glover and GaryA. Kochenberger, editeurs, Handbook of Metaheuristics, volume 57 of *International Series in Operations Research & Management Science*, pages 475–513. Springer US, 2003. (Cited on pages 43 and 192.)

[Cvijovic & Klinowski 1995] D. Cvijovic and J. Klinowski. *Taboo Search: An Approach to the Multiple Minima Problem.* Science, vol. 267, pages 664–666, February 1995. (Cited on page 57.)

[Das & Dennis 1997] I. Das and J.E. Dennis. *A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems.* Structural optimization, vol. 14, no. 1, pages 63–69, 1997. (Cited on page 25.)

[Das & Dennis 1998] Indraneel Das and J. E. Dennis. *Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems.* SIAM Journal on Optimization, vol. 8, no. 3, pages 631+, 1998. (Cited on pages 19, 27, 28, 29 and 190.)

[de la Peña 2004] M. Gulnara Baldoquín de la Peña. *Heuristics and metaheuristics approaches used to solve the rural postman problem: A comparative case study.* In Proceedings of 4th International ICSC Symposium on ENGINEERING OF INTELLIGENT SYSTEMS, EIS 2004. ICSC Interdisciplinary Research, 2004. (Cited on pages 48 and 94.)

[Deb & Agrawal 1995] Kalyanmoy Deb and Ram Bhushan Agrawal. *Simulated Binary Crossover for Continuous Search Space.* Complex Systems, vol. 9, pages 115–148, 1995. (Cited on pages 60, 61, 157 and 205.)

[Deb & Goel 2001] Kalyanmoy Deb and Tushar Goel. *A hybrid multi-objective evolutionary approach to engineering shape design.* Lecture notes in computer science, vol. 1993, pages 385–399, 2001. (Cited on page 89.)

[Deb & Sundar 2006] Kalyanmoy Deb and J. Sundar. *Reference point based multi-objective optimization using evolutionary algorithms.* In Mike Cattolico, editeur, GECCO, pages 635–642. ACM, 2006. (Cited on page 92.)

[Deb *et al.* 2000] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and T. Meyarivan. *A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation, vol. 6, pages 182–197, 2000. (Cited on pages 4, 16, 79, 82, 92, 97, 107, 109, 188 and 194.)

[Deb *et al.* 2002] Kalyanmoy Deb, Ashish Anand and Dhiraj Joshi. *A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization.* Evolutionary Computation, vol. 10, no. 4, pages 345–369, 2002. (Cited on pages 60 and 157.)

[Deb *et al.* 2003] K. Deb, M. Mohan and S. Mishra. *A fast multi-objective evolutionary algorithm for finding well-spread Pareto-optimal solutions.* Rapport technique 2003002, KanGAL, Indian Institute of Technology, Kanpur, India, 2003. (Cited on pages xiv and 83.)

[Deb *et al.* 2005a] K. Deb, L. Thiele, M. Laumanns and E. Zitzler. *Scalable Test Problems for Evolutionary Multi-Objective Optimization.* In A. Abraham, R. Jain and R. Goldberg, editeurs, Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, chapitre 6, pages 105–145. Springer, 2005. (Cited on pages 164 and 207.)

[Deb *et al.* 2005b] Kalyanmoy Deb, Manikanth Mohan and Shikhar Mishra. *Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions.* Evolutionary Computation, vol. 13, no. 4, pages 501–525, 2005. (Cited on page 142.)

[Deb 2001] K. Deb. Multi-Objective Optimization using Evolutionary Algorithms. Wiley Interscience Series in Systems and Optimization. Wiley, 2001. (Cited on pages 34, 58, 59, 60, 61, 80, 92 and 190.)

[Dor *et al.* 2012] Abbas Dor, Maurice Clerc and Patrick Siarry. *Hybridization of Differential Evolution and Particle Swarm Optimization in a New Algorithm: DEPSO-2S.* In Leszek Rutkowski, Marcin Korytkowski, Rafal Scherer, Ryszard Tadeusiewicz, LotfiA. Zadeh and JacekM. Zurada, editeurs, Swarm and Evolutionary Computation, volume 7269 of *Lecture Notes in Computer Science*, pages 57–65. Springer Berlin Heidelberg, 2012. (Cited on page 73.)

[Dorigo & Stutzle 2010] Marco Dorigo and Thomas Stutzle. *Ant Colony Optimization: Overview and Recent Advances.* In Michel Gendreau and Jean-Yves Potvin, editeurs, Handbook of Metaheuristics, volume 146 of *International Series in Operations Research & Management Science*, pages 227–263. Springer US, 2010. (Cited on page 74.)

[Dorigo *et al.* 1996] Marco Dorigo, Vittorio Maniezzo and Alberto Colorni. *The Ant System: Optimization by a colony of cooperating agents.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B, vol. 26, no. 1, pages 29–41, 1996. (Cited on pages 73, 74 and 193.)

[Dorigo 1992] M. Dorigo. *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano, Italy, 1992. (Cited on page 39.)

[Dueck & Scheuer 1990] Gunter Dueck and Tobias Scheuer. *Threshold accepting: a general purpose optimization algorithm appearing superior to simulated*

annealing. Journal of computational physics, vol. 90, no. 1, pages 161–175, 1990. (Cited on page 55.)

[Eberhart *et al.* 2001] Russell C. Eberhart, Yuhui Shi and James Kennedy. Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation). Morgan Kaufmann, 1 édition, April 2001. (Cited on pages 70 and 193.)

[Edgeworth 1961] F.Y. Edgeworth. Mathematical psychics: An essay on the application of mathematics to the moral sciences. Reprints of economics classics. Kegan Paul, 1881. [New York, A. M. Kelley, 1961. (Cited on page 8.)

[Edmonds 1971] J. Edmonds. *Matroids and the greedy algorithm*. Mathematical Programming, vol. 1, pages 127–136, 1971. (Cited on pages 43 and 192.)

[El-Abd & Kamel 2005] Mohammed El-Abd and Mohamed Kamel. *A taxonomy of cooperative search algorithms*. In Hybrid Metaheuristics, pages 32–41. Springer, 2005. (Cited on pages 85, 86 and 194.)

[Elsayed *et al.* 2011] Saber M. Elsayed, Ruhul A. Sarker and Daryl Essam. *GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems*. In IEEE Congress on Evolutionary Computation, pages 1034–1040. IEEE, 2011. (Cited on pages 157, 164 and 205.)

[Engelbrecht 2007] A.P. Engelbrecht. Computational Intelligence: An Introduction. Wiley, 2007. (Cited on pages 11 and 70.)

[Erickson *et al.* 2001] Mark Erickson, Alex Mayer and Jeffrey Horn. *The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems*. In Eckart Zitzler, Lothar Thiele, Kalyanmoy Deb, CarlosArtemio Coello Coello and David Corne, editeurs, Evolutionary Multi-Criterion Optimization, volume 1993 of *Lecture Notes in Computer Science*, pages 681–695. Springer Berlin Heidelberg, 2001. (Cited on page 194.)

[Eshelman & Schaffer 1993] L. J. Eshelman and J. D. Schaffer. *Real-coded genetic algorithms and interval-schemata*. In Darrell L. Whitley, editeur, Foundation of Genetic Algorithms 2, pages 187–202, San Mateo, CA, 1993. Morgan Kaufmann. (Cited on page 60.)

[Falkenauer 1996] Emanuel Falkenauer. *A hybrid grouping genetic algorithm for bin packing*. Journal of heuristics, vol. 2, no. 1, pages 5–30, 1996. (Cited on page 70.)

[Farina *et al.* 2004] Marco Farina, Kalyanmoy Deb and Paolo Amato. *Dynamic multiobjective optimization problems: test cases, approximations, and applications.* IEEE Trans. Evolutionary Computation, pages 425–442, 2004. (Cited on pages 92 and 188.)

[Farmer *et al.* 1986] J. D. Farmer, N. H. Packard and A. S. Perelson. *The immune system, adaptation, and machine learning.* Phys. D, vol. 2, no. 1-3, pages 187–204, 1986. (Cited on page 39.)

[Feo & Resende 1989] Thomas A. Feo and Mauricio G. C. Resende. *A probabilistic heuristic for a computationally difficult set covering problem.* Operations Research Letters, vol. 8, no. 2, pages 67 – 71, 1989. (Cited on pages 46, 94 and 192.)

[Feo & Resende 1995] Thomas A Feo and Mauricio GC Resende. *Greedy randomized adaptive search procedures.* Journal of global optimization, vol. 6, no. 2, pages 109–133, 1995. (Cited on pages 46 and 192.)

[Festa & Resende 2002] P. Festa and M. G. C. Resende. *GRASP: An annotated bibliography.* In Essays and Surveys in Metaheuristics, pages 325–367. Kluwer Academic Publishers, 2002. (Cited on page 49.)

[Festa & Resende 2009a] Paola Festa and Mauricio GC Resende. *An annotated bibliography of GRASP–Part I: Algorithms.* International Transactions in Operational Research, vol. 16, no. 1, pages 1–24, 2009. (Cited on page 49.)

[Festa & Resende 2009b] Paola Festa and Mauricio GC Resende. *An annotated bibliography of GRASP–Part II: Applications.* International Transactions in Operational Research, vol. 16, no. 2, pages 131–172, 2009. (Cited on page 49.)

[Festa & Resende 2013] Paola Festa and Mauricio GC Resende. *Hybridizations of GRASP with path-relinking.* In Hybrid Metaheuristics, pages 135–155. Springer, 2013. (Cited on page 48.)

[Fishburn 1974] Peter C Fishburn. *Lexicographic orders, utilities and decision rules: A survey.* Management science, pages 1442–1471, 1974. (Cited on pages 76 and 193.)

[Fleischer 1995] M. Fleischer. *Simulated annealing: past, present, and future.* In Simulation Conference Proceedings, 1995. Winter, pages 155–161, 1995. (Cited on page 55.)

[Fogel *et al.* 1966] L. J. Fogel, A. J. Owens and M. J. Walsh. Artificial Intelligence through Simulated Evolution. John Wiley, New York, USA, 1966. (Cited on page 57.)

[Fonseca & Fleming 1995] Carlos M. Fonseca and Peter J. Fleming. *An overview of evolutionary algorithms in multiobjective optimization.* Evol. Comput., vol. 3, no. 1, pages 1–16, March 1995. (Cited on page 58.)

[Fonseca *et al.* 1993] Carlos M Fonseca, Peter J Fleming*et al. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization.* In Proceedings of the fifth international conference on genetic algorithms, volume 423, pages 416–423. San Mateo, California, 1993. (Cited on pages 79, 80, 81 and 194.)

[Gál *et al.* 1999] T. Gál, T.J. Stewart and T. Hanne. Multicriteria Decision Making: Advances in McDm Models, Algorithms, Theory, and Applications. International Series in Operations Research & Management Science, 21. Kluwer Acad., 1999. (Cited on pages 8 and 188.)

[Gambardella & Dorigo 2000] Luca Maria Gambardella and Marco Dorigo. *An ant colony system hybridized with a new local search for the sequential ordering problem.* INFORMS Journal on Computing, vol. 12, no. 3, pages 237–255, 2000. (Cited on page 74.)

[Gass & Saaty 1955] S. Gass and T.L. Saaty. *The computational algorithm for the parametric objective function.* Naval Research Logistics Quarterly, vol. 2, page 39, 1955. (Cited on pages 24 and 190.)

[Gembicki & Haimes 1975] F. Gembicki and Y.Y. Haimes. *Approach to performance and sensitivity multiobjective optimization: The goal attainment method.* Automatic Control, IEEE Transactions on, vol. 20, no. 6, pages 769–771, 1975. (Cited on page 32.)

[Glover & Kochenberger 2003] F.E. Glover and G.A. Kochenberger. Handbook in Metaheuristics. International Series in Operations Research & Management Science, 57. Kluwer Academic Publishers, 2003. (Cited on pages 39, 46, 49, 84, 191, 192 and 194.)

[Glover & Laguna 1997] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, 1997. (Cited on page 57.)

[Glover *et al.* 2000] Fred Glover, Manuel Laguna and Rafael Martí. *Fundamentals of scatter search and path relinking.* CONTROL AND CYBERNETICS, vol. 39, pages 653–684, 2000. (Cited on pages 68, 87, 122, 193 and 199.)

[Glover *et al.* 2003] Fred Glover, Manuel Laguna and Rafael Marti. *Scatter Search and Path Relinking: Advances and Applications Handbook of Metaheuristics*. In Fred Glover and Gary A. Kochenberger, editeurs, Handbook of Metaheuristics, volume 57 of *International Series in Operations Research & Management Science*, chapitre 1, pages 1–35. Springer New York, Boston, 2003. (Cited on pages 68, 92 and 197.)

[Glover 1977] Fred Glover. *HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS*. Decision Sciences, vol. 8, no. 1, pages 156–166, 1977. (Cited on page 39.)

[Glover 1986] Fred Glover. *Future paths for integer programming and links to artificial intelligence*. Comput. Oper. Res., vol. 13, no. 5, pages 533–549, May 1986. (Cited on pages 39, 55, 191 and 192.)

[Glover 1989] Fred Glover. *Tabu search-part I*. ORSA Journal on computing, vol. 1, no. 3, pages 190–206, 1989. (Cited on pages 55 and 192.)

[Glover 1996] Fred Glover. *Tabu search and adaptive memory programing–Advances, applications and challenges*. In Interfaces in Computer Science and Operations Research, pages 1–75. Kluwer, 1996. (Cited on pages 68, 93 and 193.)

[Goldberg 1989] D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Artificial Intelligence. Addison-Wesley, 1989. (Cited on pages 37, 38, 58, 59, 76, 80, 87, 191 and 193.)

[Golovkin *et al.* 2002] Igor E Golovkin, Sushil J Louis and Roberto C Mancini. *Parallel implementation of niched Pareto genetic algorithm code for X-ray plasma spectroscopy*. In Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, volume 2, pages 1820–1824. IEEE, 2002. (Cited on page 89.)

[Haimes *et al.* 1971] Yacov Y. Haimes, Leon S. Lasdon and David A. Wismer. *On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization*. Systems, Man and Cybernetics, IEEE Transactions on, vol. SMC-1, no. 3, pages 296–297, 1971. (Cited on page 26.)

[Hansen *et al.* 2008] Pierre Hansen, Nenad Mladenovic and José Moreno Pérez. *Variable neighbourhood search: methods and applications*. 4OR, vol. 6, no. 4, pages 319–360, 2008. (Cited on page 51.)

[Hansen *et al.* 2010] Pierre Hansen, Nenad Mladenović and José A Moreno Pérez. *Variable neighbourhood search: methods and applications.* Annals of Operations Research, vol. 175, no. 1, pages 367–407, 2010. (Cited on page 51.)

[Hansen 1986] P. Hansen. *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming.* In Proceedings of the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986. (Cited on page 39.)

[Hansen 1997] Michael Pilegaard Hansen. *Tabu search for multiobjective optimization: MOTS.* In Proceedings of the 13th International Conference on Multiple Criteria Decision Making, pages 574–586. Citeseer, 1997. (Cited on pages 75 and 193.)

[Hernández-Díaz *et al.* 2007] Alfredo G. Hernández-Díaz, Luis V. Santana-Quintero, CarlosA. Coello Coello and Julián Molina Luque. *Pareto-adaptive epsilon-dominance.* Evolutionary Computation, vol. 15, no. 4, pages 493–517, 2007. (Cited on pages 4, 78, 142, 144, 146, 147, 193, 201 and 202.)

[Hertz & Kobler 2000] Alain Hertz and Daniel Kobler. *A framework for the description of evolutionary algorithms.* European Journal of Operational Research, vol. 126, no. 1, pages 1–12, 2000. (Cited on page 85.)

[Hirsch *et al.* 2007] MJ Hirsch, CN Meneses, PM Pardalos and MGC Resende. *Global optimization by continuous GRASP.* Optimization Letters, vol. 1, no. 2, pages 201–212, 2007. (Cited on page 49.)

[Holland 1975] J.H. Holland. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975. (Cited on pages 39, 57, 58 and 193.)

[Horn *et al.* 1994] J. Horn, N. Nafpliotis and D.E. Goldberg. *A niched Pareto genetic algorithm for multiobjective optimization.* In Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pages 82–87 vol.1, 1994. (Cited on pages 79, 80 and 194.)

[Hromkovic 2005] J. Hromkovic. Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2005. (Cited on pages 37 and 191.)

[Hsieh *et al.* 2007] Chang-Tai Hsieh, Chih-Ming Chen and Ying-ping Chen. *Particle swarm guided evolution strategy.* In Proceedings of the 9th annual conference on Genetic and evolutionary computation, GECCO '07, pages 650–657, New York, NY, USA, 2007. ACM. (Cited on pages 159 and 205.)

[Hu & Eberhart 2002] Xiaohui Hu and R. Eberhart. *Multiobjective optimization using dynamic neighborhood particle swarm optimization.* In Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, volume 2, pages 1677–1681, 2002. (Cited on page 194.)

[Hwang & Masud 1979] C.L. Hwang and A.S.M. Masud. Multiple objective decision making, methods and applications: a state-of-the-art survey. Lecture notes in economics and mathematical systems. Springer-Verlag, 1979. (Cited on pages 32, 37, 190 and 191.)

[Hwang *et al.* 1980] Ching-Lai Hwang, SR Paidy, K Yoon and ASM Masud. *Mathematical programming with multiple objectives: A tutorial.* Computers & Operations Research, vol. 7, no. 1, pages 5–31, 1980. (Cited on page 17.)

[Idoumghar *et al.* 2011] Lhassane Idoumghar, Mahmoud Melkemi, René Schott and Maha Idrissi Aouad. *Hybrid PSO-SA type algorithms for multimodal function optimization and reducing energy consumption in embedded systems.* Appl. Comp. Intell. Soft Comput., vol. 2011, pages 3:1–3:12, January 2011. (Cited on page 73.)

[Ijiri 1965] Y. Ijiri. Management goals and accounting for control. Studies in mathematical and managerial economics. North Holland Pub. Co., 1965. (Cited on pages 31 and 190.)

[Iorio & Li 2004] Antony W. Iorio and Xiaodong Li. *Solving rotated multiobjective optimization problems using Differential Evolution.* In In AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence, pages 861–872. press, 2004. (Cited on page 65.)

[Ishibuchi & Murata 1998] Hisao Ishibuchi and Tadahiko Murata. *Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling.* IEEE Transactions On Systems Man And Cybernetics, vol. 28, no. 3, pages 392–403, 1998. (Cited on pages 79, 87, 92, 97 and 194.)

[Ishibuchi *et al.* 2009] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto and Yusuke Nojima. *Effects of using two neighborhood structures on the performance of cellular evolutionary algorithms for many-objective optimization.* In IEEE Congress on Evolutionary Computation, pages 2508–2515. IEEE, 2009. (Cited on pages 96 and 99.)

[Jaszkiewicz 2002a] Andrzej Jaszkiewicz. *Genetic local search for multi-objective combinatorial optimization.* European journal of operational research, vol. 137, no. 1, pages 50–71, 2002. (Cited on page 70.)

[Jaszkiewicz 2002b] Andrzej Jaszkiewicz. *On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment.* IEEE Trans. Evolutionary Computation, vol. 6, no. 4, pages 402–412, 2002. (Cited on pages 79, 87, 92, 188 and 194.)

[Jaszkiewicz 2003] Andrzej Jaszkiewicz. *Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem.* IEEE Trans. Evolutionary Computation, vol. 7, no. 2, pages 133–143, 2003. (Cited on pages 92 and 188.)

[Jones & Tamiz 2010] Dylan Jones and Mehrdad Tamiz. Practical goal programming, volume 141. Springer, 2010. (Cited on pages 32 and 190.)

[Jourdan *et al.* 2005] Laetitia Jourdan, David Corne, Dragan Savic and Godfrey Walters. *Preliminary investigation of the learnable evolution model for faster/better multiobjective water systems design.* In Evolutionary Multi-Criterion Optimization, pages 841–855. Springer, 2005. (Cited on pages 89 and 194.)

[Jourdan *et al.* 2006] Laetitia Jourdan, Clarisse Dhaenens and El-Ghazali Talbi. *Using datamining techniques to help metaheuristics: a short survey.* In Proceedings of the Third international conference on Hybrid Metaheuristics, HM'06, pages 57–69, Berlin, Heidelberg, 2006. Springer-Verlag. (Cited on pages 94 and 197.)

[Kafafy *et al.* 2011] Ahmed Kafafy, Ahmed Bounekkar and Stéphane Bonnevay. *A Hybrid Evolutionary Metaheuristics (HEMH) Applied on 0/1 Multiobjective Knapsack Problems.* In Krasnogor & Lanzi [Krasnogor & Lanzi 2011], pages 497–504. (Cited on pages 120, 135, 137, 139, 155, 159, 189, 196, 201 and 203.)

[Kafafy *et al.* 2012a] Ahmed Kafafy, Ahmed Bounekkar and Stéphane Bonnevay. *HEMH2: An Improved Hybrid Evolutionary Metaheuristics for 0/1 Multiobjective Knapsack Problems.* In Proceedings of the 9th International

Conference (SEAL 2012), Hanoi, Vietnam, December 16-19, 2012, volume 7673 of *Lecture Notes in Computer Science*, pages 104–116, New York, Germany, 2012. (Cited on pages 43, 67, 155, 159, 189 and 203.)

[Kafafy *et al.* 2012b] Ahmed Kafafy, Ahmed Bounekkar and Stéphane Bonnevay. *Hybrid Metaheuristics based on MOEA/D for 0/1 multiobjective knapsack problems: A comparative study.* In IEEE Congress on Evolutionary Computation, pages 3616–3623. IEEE, 2012. (Cited on pages 137, 141, 143, 159, 189, 201 and 203.)

[Kafafy *et al.* 2013] Ahmed Kafafy, Stéphane Bonnevay and Ahmed Bounekkar. *A Hybrid Evolutionary Approach with Search Strategy Adaptation for Multiobjective Optimization problems.* In Proceedings of 15th annual conference on Genetic and evolutionary computation, GECCO 2013, Amesterdam, Neatherlands, GECCO '13, New York, NY, USA, July 6-10 2013. ACM. (Cited on pages 156 and 189.)

[Kennedy & Eberhart 1995] J. Kennedy and R. C. Eberhart. *Particle swarm optimization.* In IEEE international conference on neural networks IV, pages 1942–1948, 1995. (Cited on pages 39, 70, 158, 193 and 205.)

[Kennedy & Eberhart 1997] J. Kennedy and R. C. Eberhart. *A discrete binary version of the particle swarm algorithm.* In IEEE International Conference on Systems, Man and Cybernetics, volume 5, pages 4104–4108, 1997. (Cited on page 73.)

[Kennedy & Mendes 2002] J. Kennedy and R. Mendes. *Population structure and particle swarm performance.* In Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, volume 2, pages 1671–1676, 2002. (Cited on page 71.)

[Kirkpatrick *et al.* 1983] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. *Optimization by Simulated Annealing.* Science, Number 4598, 13 May 1983, vol. 220, 4598, pages 671–680, 1983. (Cited on pages 39, 54 and 192.)

[Knowles & Corne 1999] J. Knowles and D. Corne. *The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation.* In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 1, pages –105 Vol. 1, 1999. (Cited on page 79.)

[Knowles & Corne 2000] Joshua D. Knowles and David W. Corne. *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy.* Evol. Comput., vol. 8, no. 2, pages 149–172, June 2000. (Cited on pages 4 and 194.)

[Knowles & Corne 2002] J. Knowles and D. Corne. *On metrics for comparing nondominated sets.* Computational Intelligence, Proceedings of the World on Congress on, vol. 1, pages 711–716, 2002. (Cited on pages 22, 190, 200 and 203.)

[Konak *et al.* 2006] Abdullah Konak, David W Coit and Alice E Smith. *Multi-objective optimization using genetic algorithms: A tutorial.* Reliability Engineering & System Safety, vol. 91, no. 9, pages 992–1007, 2006. (Cited on page 62.)

[Koza 1992] John R. Koza. Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA, USA, 1992. (Cited on page 57.)

[Kramer 2010] Oliver Kramer. *A review of constraint-handling techniques for evolution strategies.* Appl. Comp. Intell. Soft Comput., vol. 2010, pages 3:1–3:19, January 2010. (Cited on page 63.)

[Krasnogor & Lanzi 2011] Natalio Krasnogor and Pier Luca Lanzi, editeurs. 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011. ACM, 2011. (Cited on pages 245 and 248.)

[Kursawe 1991] Frank Kursawe. *A Variant of Evolution Strategies for Vector Optimization.* In Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, PPSN I, pages 193–197, London, UK, UK, 1991. Springer-Verlag. (Cited on page 79.)

[Laguna & Martí 1999] Manuel Laguna and Rafael Martí. *GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization.* INFORMS Journal on Computing, vol. 11, no. 1, pages 44–52, 1999. (Cited on page 96.)

[Laguna *et al.* 2003] Manuel Laguna, Rafael Marti and Rafael Cunquero Martí. Scatter search: methodology and implementation in C, volume 24. Springer, 2003. (Cited on pages 67, 68 and 193.)

[Lamont & Veldhuizen 2002] Gary B. Lamont and David A. Van Veldhuizen. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Norwell, MA, USA, 2002. (Cited on page 92.)

[Laumanns *et al.* 2002] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb and Eckart Zitzler. *Combining convergence and diversity in evolutionary multiobjective optimization.* Evolutionary computation, vol. 10, no. 3, pages 263–282, 2002. (Cited on pages 78 and 193.)

[Lee & Olson 1999] Sang M Lee and David L Olson. *Goal programming.* In Multicriteria Decision Making, pages 203–235. Springer, 1999. (Cited on pages 31 and 190.)

[Li & Zhang 2009] Hui Li and Qingfu Zhang. *Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II.* Trans. Evol. Comp, vol. 13, no. 2, pages 284–302, April 2009. (Cited on pages 164 and 207.)

[Liang & Suganthan 2006] J.J. Liang and P.N. Suganthan. *Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism.* In Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pages 9–16, 2006. (Cited on page 71.)

[Liberti & Drazic 2005] Leo Liberti and M Drazic. *Variable neighbourhood search for the global optimization of constrained NLPs.* In Proceedings of GO Workshop, Almeria, Spain, volume 2005, 2005. (Cited on page 51.)

[Liu & Lampinen 2005] Jounhong Liu and Jouni Lampinen. *A fuzzy adaptive differential evolution algorithm.* Soft Computing, vol. 9, no. 6, pages 448–462, 2005. (Cited on page 66.)

[Lourenço *et al.* 2010] Helena R Lourenço, Olivier C Martin and Thomas Stützle. *Iterated local search: Framework and applications.* In Handbook of Metaheuristics, pages 363–397. Springer, 2010. (Cited on page 50.)

[Lozano & García-Martínez 2010] Manuel Lozano and Carlos García-Martínez. *Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report.* Computers & OR, vol. 37, no. 3, pages 481–497, 2010. (Cited on pages 3, 93 and 189.)

[Martello & Toth 1990] Silvano Martello and Paolo Toth. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc., New York, NY, USA, 1990. (Cited on page 16.)

[Martínez & Coello 2011] Saúl Zapotecas Martínez and Carlos A. Coello Coello. *A multi-objective particle swarm optimizer based on decomposition.* In Krasnogor & Lanzi [Krasnogor & Lanzi 2011], pages 69–76. (Cited on pages 4, 164, 194 and 207.)

[Merkle *et al.* 2002] Daniel Merkle, Martin Middendorf and Hartmut Schmeck. *Ant colony optimization for resource-constrained project scheduling.* Evolutionary Computation, IEEE Transactions on, vol. 6, no. 4, pages 333–346, 2002. (Cited on page 74.)

[Messac *et al.* 2003] A. Messac, A. Ismail-Yahaya and C.A. Mattson. *The normalized normal constraint method for generating the Pareto frontier.* Structural and Multidisciplinary Optimization, vol. 25, no. 2, pages 86–98, 2003. (Cited on pages 19 and 28.)

[Meunier *et al.* 2000] Herve Meunier, E-G Talbi and Philippe Reininger. *A multiobjective genetic algorithm for radio network optimization.* In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 1, pages 317–324. IEEE, 2000. (Cited on page 89.)

[Mezura-Montes *et al.* 2006] Efrñn Mezura-Montes, Jesús Velázquez-Reyes and Carlos A. Coello Coello. *A comparative study of differential evolution variants for global optimization.* In Proceedings of the 8th annual conference on Genetic and evolutionary computation, GECCO '06, pages 485–492, New York, NY, USA, 2006. ACM. (Cited on page 65.)

[Mezura-Montes *et al.* 2008] EfrÃ©n Mezura-Montes, Margarita Reyes-Sierra and CarlosA.Coello Coello. *Multi-objective Optimization Using Differential Evolution: A Survey of the State-of-the-Art.* In UdayK. Chakraborty, editeur, Advances in Differential Evolution, volume 143 of *Studies in Computational Intelligence*, pages 173–196. Springer Berlin Heidelberg, 2008. (Cited on page 67.)

[Michalewicz & Fogel 2004] Z. Michalewicz and D.B. Fogel. How to Solve It: Modern Heuristics. Springer, 2004. (Cited on pages 39 and 191.)

[Michalewicz 1996] Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Artificial intelligence. Springer, 1996. (Cited on pages 37, 58, 60, 61 and 191.)

[Miettinen 1999] K. Miettinen. Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston, 1999. (Cited on pages 9, 10, 11, 14, 17, 18, 24, 25, 26, 32, 79, 92, 121, 188 and 190.)

[Mladenović & Hansen 1997] Nenad Mladenović and Pierre Hansen. *Variable neighborhood search.* Computers & Operations Research, vol. 24, no. 11, pages 1097–1100, 1997. (Cited on pages 50 and 192.)

[Mladenović *et al.* 2008] Nenad Mladenović, Milan Dražić, Vera Kovačevic-Vujčić and Mirjana Čangalović. *General variable neighborhood search for the continuous optimization.* European Journal of Operational Research, vol. 191, no. 3, pages 753–770, 2008. (Cited on page 51.)

[Moscato 1989] Pablo Moscato. *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.* Caltech concurrent computation program, C3P Report, vol. 826, page 1989, 1989. (Cited on pages 70 and 193.)

[Moscato 1999] Pablo Moscato. *Memetic algorithms: a short introduction.* In David Corne, Marco Dorigo, Fred Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli and Kenneth V. Price, editeurs, New ideas in optimization, pages 219–234. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999. (Cited on pages 70 and 193.)

[Mostaghim & Teich 2003] S. Mostaghim and J. Teich. *Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO).* In Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, pages 26–33, 2003. (Cited on pages 37, 191 and 194.)

[Neri & Tirronen 2010] Ferrante Neri and Ville Tirronen. *Recent advances in differential evolution: a survey and experimental analysis.* Artificial Intelligence Review, vol. 33, no. 1-2, pages 61–106, 2010. (Cited on page 67.)

[Nomura 1997] Tatsuya Nomura. *An Analysis on Linear Crossover for Real Number Chromosomes in an Infinite Population Size.* In In Proc. ICEC'97, pages 111–114, 1997. (Cited on page 60.)

[Oei *et al.* 1991] C. K. Oei, David E. Goldberg and S. Chang. *Tournament selection, niching, and the preservation of diversity.* Rapport technique IlliGAL Report 91011, University of Illinois, 1991. (Cited on page 194.)

[Ono *et al.* 2003] Isao Ono, Hajime Kita and Shigenobu Kobayashi. *A real-coded genetic algorithm using the unimodal normal distribution crossover.* In Advances in evolutionary computing, pages 213–237. Springer, 2003. (Cited on page 60.)

[Osman & Laporte 1996] Ibrahim Osman and Gilbert Laporte. *Metaheuristics: A bibliography.* Annals of Operations Research, vol. 63, no. 5, pages 511–623, October 1996. (Cited on pages 39 and 191.)

[Pant *et al.* 2009] Millie Pant, Radha Thangaraj and Ajith Abraham. *Particle Swarm Optimization: Performance Tuning and Empirical Analysis.* In Ajith Abraham, Aboul-Ella Hassanien, Patrick Siarry and Andries Engelbrecht, editeurs, Foundations of Computational Intelligence Volume 3, volume 203 of *Studies in Computational Intelligence*, pages 101–128. Springer Berlin Heidelberg, 2009. (Cited on page 73.)

[Papadimitriou & Steiglitz 1998] C.H. Papadimitriou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Dover Books on Computer Science Series. DOVER PUBN Incorporated, 1998. (Cited on page 45.)

[Pareto 1964] V. Pareto. Cours d'économie politique. Droz, 1964. (Cited on page 8.)

[Parsopoulos & Vrahatis 2002] K.E. Parsopoulos and M.N. Vrahatis. *Recent approaches to global optimization problems through Particle Swarm Optimization.* Natural Computing, vol. 1, no. 2-3, pages 235–306, 2002. (Cited on page 194.)

[Pinto *et al.* 2005] Pedro Pinto, ThomasA. Runkler and Joao M. Sousa. *Wasp swarm optimization of logistic systems.* In Bernardete Ribeiro, RudolfF. Albrecht, Andrej Dobnikar, DavidW. Pearson and NigelC. Steele, editeurs, Adaptive and Natural Computing Algorithms, pages 264–267. Springer Vienna, 2005. (Cited on page 70.)

[Prais & Ribeiro 2000] Marcelo Prais and Celso C. Ribeiro. *Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment.* INFORMS Journal on Computing, vol. 12, no. 3, pages 164–176, 2000. (Cited on page 49.)

[Prestwich & Roli 2005] Steven Prestwich and Andrea Roli. *Symmetry Breaking and Local Search Spaces.* In Roman BartÃ¡k and Michela Milano, editeurs, Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, volume 3524 of *Lecture Notes in Computer Science*, pages 273–287. Springer Berlin Heidelberg, 2005. (Cited on page 45.)

[Price *et al.* 2005] K. Price, R.M. Storn and J. Lampinen. Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer, 2005. (Cited on pages 64, 65, 66, 139, 158, 193 and 205.)

[Puchinger & Raidl 2005] Jakob Puchinger and Günther R Raidl. *Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification.* In Artificial intelligence and knowledge engineering applications: a bioinspired approach, pages 41–53. Springer, 2005. (Cited on pages 85, 86 and 194.)

[Qin *et al.* 2009] A. K. Qin, V. L. Huang and P. N. Suganthan. *Differential evolution algorithm with strategy adaptation for global numerical optimization.*

Trans. Evol. Comp, vol. 13, no. 2, pages 398–417, April 2009. (Cited on page 65.)

[Radcliffe 1991] Nicholas J Radcliffe. *Equivalence class analysis of genetic algorithms*. Complex Systems, vol. 5, no. 2, pages 183–205, 1991. (Cited on page 60.)

[Raidl 2006] Günther R. Raidl. *A Unified View on Hybrid Metaheuristics*. In Francisco Almeida andMaría J. Blesa Aguilera andChristian Blum andJ. Marcos Moreno-Vega andMelquíades Pérez Pérez andAndrea Roli andMichael Sampels, editeur, Hybrid Metaheuristics, volume 4030 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. (Cited on pages xiv, 3, 85, 88, 92, 119 and 194.)

[Rayward-Smith 1996] V.J. Rayward-Smith. Modern heuristic search methods. Wiley, 1996. (Cited on pages 39 and 191.)

[Rechenberg 1965] I. Rechenberg. *Cybernetic solution path of an experimental problem*. In Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans. Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants, August 1965. (Cited on page 62.)

[Rechenberg 1973] I. Rechenberg. Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog, 1973. (Cited on pages 39, 57 and 62.)

[Reeves 1993] C.R. Reeves. Modern heuristic techniques for combinatorial problems. Advanced topics in computer science series. Halsted Press, 1993. (Cited on pages 39 and 191.)

[Reeves 1996] Colin Reeves. *Hybrid genetic algorithms for bin-packing and related problems*. Annals of Operations Research, vol. 63, no. 3, pages 371–396, 1996. (Cited on page 70.)

[Resende & Ribeiro 1997] Mauricio GC Resende and Celso C Ribeiro. *A GRASP for graph planarization*. Networks, vol. 29, no. 3, pages 173–189, 1997. (Cited on page 49.)

[Resende & Ribeiro 2003] M.G.C. Resende and C.C. Ribeiro. *Greedy randomized adaptive search procedures*. In F. Glover and G. Kochenberger, editeurs, Handbook of Metaheuristics, pages 219–249. Kluwer Academic Publishers, 2003. (Cited on pages 48 and 96.)

[Resende & Ribeiro 2005] Mauricio G. C. Resende and Celso C. Ribeiro. *GRASP with path-relinking: Recent advances and applications.* In Metaheuristics: Progress as Real Problem Solvers, pages 29–63. Springer, 2005. (Cited on page 96.)

[Resende & Werneck 2004] Mauricio G. C. Resende and Renato Fonseca F. Werneck. *A Hybrid Heuristic for the p-Median Problem.* J. Heuristics, vol. 10, no. 1, pages 59–88, 2004. (Cited on page 96.)

[Resende & Werneck 2006] Mauricio G. C. Resende and Renato Fonseca F. Werneck. *A hybrid multistart heuristic for the uncapacitated facility location problem.* European Journal of Operational Research, vol. 174, no. 1, pages 54–68, 2006. (Cited on page 96.)

[Resende *et al.* 2010a] Mauricio G. C. Resende, Rafael Martí, Micael Gallego and Abraham Duarte. *GRASP and path relinking for the max-min diversity problem.* Computers & OR, vol. 37, no. 3, pages 498–508, 2010. (Cited on page 96.)

[Resende *et al.* 2010b] MauricioG.C. Resende, CelsoC. Ribeiro, Fred Glover and Rafael MartÃ. *Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications.* In Michel Gendreau and Jean-Yves Potvin, editeurs, Handbook of Metaheuristics, volume 146 of *International Series in Operations Research & Management Science*, pages 87–107. Springer US, 2010. (Cited on page 68.)

[Resende 2008] M Resende. *Metaheuristic hybridization with greedy randomized adaptive search procedures.* TutORials in Operations Research, 2008. (Cited on page 48.)

[Reyes-Sierra & Coello 2006] Margarita Reyes-Sierra and CA Coello Coello. *Multi-objective particle swarm optimizers: A survey of the state-of-the-art.* International Journal of Computational Intelligence Research, vol. 2, no. 3, pages 287–308, 2006. (Cited on page 73.)

[Ribeiro *et al.* 2002] Celso C. Ribeiro, Eduardo Uchoa and Renato Fonseca F. Werneck. *A Hybrid GRASP with Perturbations for the Steiner Problem in Graphs.* INFORMS Journal on Computing, vol. 14, no. 3, pages 228–246, 2002. (Cited on pages 69, 95 and 141.)

[Ribeiro *et al.* 2006] Marcos Henrique Ribeiro, Alexandre Plastino and Simone L. Martins. *Hybridization of GRASP Metaheuristic with Data Mining Techniques.* J. Math. Model. Algorithms, vol. 5, no. 1, pages 23–41, 2006. (Cited on pages 93, 94, 196 and 197.)

[Robbins & Monro 1951] Herbert Robbins and Sutton Monro. *A Stochastic Approximation Method*. The Annals of Mathematical Statistics, vol. 22, no. 3, pages 400–407, 1951. (Cited on pages 37 and 191.)

[Rudolph & Agapie 2000] G. Rudolph and A. Agapie. *Convergence properties of some multi-objective evolutionary algorithms*. In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 2, pages 1010–1016 vol.2, 2000. (Cited on page 79.)

[Rudolph 1998] G. Rudolph. *On a multi-objective evolutionary algorithm and its convergence to the Pareto set*. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 511–516, 1998. (Cited on page 79.)

[Runkler 2008] Thomas A. Runkler. *Wasp swarm optimization of the c-means clustering model*. International Journal of Intelligent Systems, vol. 23, no. 3, pages 269–285, 2008. (Cited on page 70.)

[Russell & Norvig 2010] S.J. Russell and P. Norvig. Artificial intelligence: a modern approach. Prentice Hall series in artificial intelligence. Pearson Education/Prentice Hall, 2010. (Cited on pages 37 and 191.)

[Sakawa 1982] Masatoshi Sakawa. *Interactive multiobjective decision making by the sequential proxy optimization technique: SPOT*. European Journal of Operational Research, vol. 9, no. 4, pages 386–396, 1982. (Cited on page 190.)

[Sakawa 1993] M. Sakawa. Fuzzy sets and interactive multiobjective optimization. Numeéro v. 1 de Applied information technology. Plenum, 1993. (Cited on pages 11, 32 and 190.)

[Salehipour *et al.* 2011] Amir Salehipour, Kenneth Sörensen, Peter Goos and Olli Bräysy. *Efficient GRASP+ VND and GRASP+ VNS metaheuristics for the traveling repairman problem*. 4OR, vol. 9, no. 2, pages 189–209, 2011. (Cited on page 51.)

[Santos *et al.* 2008] Luis F. Santos, Simone L. Martins and Alexandre Plastino. *Applications of the DM-GRASP heuristic: a survey*. International Transactions in Operational Research, vol. 15, no. 4, pages 387–416, 2008. (Cited on page 94.)

[Schaffer 1985] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. In Proceedings of the 1st International

Conference on Genetic Algorithms, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc. (Cited on pages 76, 79, 80 and 193.)

[Shi & Eberhart 1998] Yuhui Shi and R. Eberhart. *A modified particle swarm optimizer*. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 69–73, 1998. (Cited on page 73.)

[Shmygelska & Hoos 2005] Alena Shmygelska and Holger H Hoos. *An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem*. BMC bioinformatics, vol. 6, no. 1, page 30, 2005. (Cited on page 74.)

[Sinha & Glodberg 2003] A. Sinha and D. Glodberg. *A Survey of Hybrid Genetic and Evolutionary algorithms*. Technical Report 2003004, Illinois Genetic Algorithms Laboratory, Illinois, 2003. (Cited on page 62.)

[Spears 2000] W.M. Spears. Evolutionary Algorithms: The Role of Mutation and Recombination. Natural Computing Series. Springer, 2000. (Cited on pages 60 and 61.)

[Srinivas & Deb 1994] N. Srinivas and Kalyanmoy Deb. *Muiltiobjective optimization using nondominated sorting in genetic algorithms*. Evol. Comput., vol. 2, no. 3, pages 221–248, September 1994. (Cited on pages 79, 80, 82 and 194.)

[Steuer & Choo 1983] R.E. Steuer and E. Choo. *An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming*. Mathematical Programming, vol. 26, no. 1, pages 326–344, 1983. (Cited on page 18.)

[Steuer 1986] R. E. Steuer. Multiple Criteria Optimization: Theory, Computation and Application. John Wiley, New York, 546 pp, 1986. (Cited on pages 9 and 17.)

[Stewart *et al.* 2004] Theodor J. Stewart, Ron Janssen and Marjan van Herwijnen. *A genetic algorithm approach to multiobjective land use planning*. Comput. Oper. Res., vol. 31, no. 14, pages 2293–2313, December 2004. (Cited on page 92.)

[Storn & Price 1995] R. Storn and K. Price. *Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. Rapport technique, International Computer Science Institute, Berkeley, CA, 1995. (Cited on page 64.)

[Storn & Price 1997] Rainer Storn and Kenneth Price. *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces.* Journal of global optimization, vol. 11, no. 4, pages 341–359, 1997. (Cited on page 64.)

[Storn 1996] Rainer Storn. *On the usage of differential evolution for function optimization.* In Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American, pages 519–523, 1996. (Cited on page 65.)

[Taillard *et al.* 2001] Éric D Taillard, Luca M Gambardella, Michel Gendreau and Jean-Yves Potvin. *Adaptive memory programming: A unified view of metaheuristics.* European Journal of Operational Research, vol. 135, no. 1, pages 1–16, 2001. (Cited on pages 41 and 192.)

[Taillard 1991] E Taillard. *Robust taboo search for the quadratic assignment problem.* Parallel computing, vol. 17, no. 4, pages 443–455, 1991. (Cited on page 57.)

[Tairan & Zhang 2010] Nasser Tairan and Qingfu Zhang. *Population-based guided local search: Some preliminary experimental results.* In Evolutionary Computation (CEC), 2010 IEEE Congress on, pages 1–5. IEEE, 2010. (Cited on page 53.)

[Talbi 2002] E.-G. Talbi. *A Taxonomy of Hybrid Metaheuristics.* Journal of Heuristics, vol. 8, no. 5, pages 541–564, 2002. (Cited on pages xiv, 85, 86, 87 and 194.)

[Talbi 2009] El-Ghazali Talbi. Metaheuristics - From Design to Implementation. Wiley, 2009. (Cited on pages 40, 46, 49, 50, 53 and 192.)

[Teng *et al.* 2009] Nga Sing Teng, Jason Teo and Mohd Hanafi A Hijazi. *Self-adaptive population sizing for a tune-free differential evolution.* Soft Computing, vol. 13, no. 7, pages 709–724, 2009. (Cited on page 66.)

[Triantaphyllou 2000] E. Triantaphyllou. Multi-Criteria Decision Making Methodologies: A Comparative Study, volume 44 of *Applied Optimization.* Kluwer Academic, Dordrecht, 2000. (Cited on pages 8 and 188.)

[Tsutsui *et al.* 1999a] Shigeyoshi Tsutsui, Masayuki Yamamura and Takahide Higuchi. *Multi-parent recombination with simplex crossover in real coded genetic algorithms.* In Proceedings of the genetic and evolutionary computation conference, volume 1, pages 657–664, 1999. (Cited on page 60.)

[Tsutsui *et al.* 1999b] Shigeyoshi Tsutsui, Masayuki Yamamura and Takahide Higuchi. *Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms.* In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela and Robert E. Smith, editeurs, Proceedings of the Genetic and Evolutionary Computation Conference, volume 1, pages 657–664, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann. (Cited on page 157.)

[Ulungu *et al.* 1999] EL Ulungu, JFPH Teghem, PH Fortemps and D Tuyttens. *MOSA method: a tool for solving multiobjective combinatorial optimization problems.* Journal of Multi-Criteria Decision Analysis, vol. 8, no. 4, pages 221–236, 1999. (Cited on pages 75 and 193.)

[Valdez *et al.* 2011] Fevrier Valdez, Patricia Melin and Oscar Castillo. *Bio-Inspired Optimization Methods for Minimization of Complex Mathematical Functions.* In Ildar Batyrshin and Grigori Sidorov, editeurs, Advances in Soft Computing, volume 7095 of *Lecture Notes in Computer Science*, pages 131–142. Springer Berlin Heidelberg, 2011. (Cited on page 73.)

[van den Bergh & Engelbrecht 2006] F. van den Bergh and A.P. Engelbrecht. *A study of particle swarm optimization particle trajectories.* Information Sciences, vol. 176, no. 8, pages 937 – 971, 2006. (Cited on page 73.)

[Van Veldhuizen & Lamont 2000] D.A. Van Veldhuizen and G.B. Lamont. *On measuring multiobjective evolutionary algorithm performance.* In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 1, pages 204–211 vol.1, 2000. (Cited on pages 4 and 22.)

[Veldhuizen 1999] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999. (Cited on pages 37 and 191.)

[Vianna & Arroyo 2004] Dalessandro Soares Vianna and José Elias Claudio Arroyo. *A GRASP Algorithm for the Multi-Objective Knapsack Problem.* In SCCC, pages 69–75. IEEE Computer Society, 2004. (Cited on pages 4, 99, 101 and 107.)

[Villegas *et al.* 2010] Juan G. Villegas, Christian Prins, Caroline Prodhon, Andrés L. Medaglia and Nubia Velasco. *GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots* . Engineering Applications of Artificial Intelligence,

vol. 23, no. 5, pages 780 – 794, 2010. <ce:title>Advances in metaheuristics for hard optimization: new trends and case studies</ce:title>. (Cited on pages 48 and 51.)

[Voigt *et al.* 1995] H-M Voigt, Heinz Mühlenbein and Dragan Cvetkovic. *Fuzzy recombination for the breeder genetic algorithm.* In Proc. Sixth Int. Conf. on Genetic Algorithms. Citeseer, 1995. (Cited on page 60.)

[Voss *et al.* 1999] Stefan Voss, Ibrahim H. Osman and Catherine Roucairol, editeurs. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, Norwell, MA, USA, 1999. (Cited on pages 39 and 191.)

[Voudouris & Tsang 1999] C. Voudouris and E. Tsang. *Guided local search.* European Journal of Operational Research, vol. 113, pages 469–499, 1999. (Cited on pages 51 and 192.)

[Voudouris 1997] C. Voudouris. *Guided Local Search for Combinatorial Optimization Problems.* PhD thesis, Department of Computer Science, University of Essex, 1997. (Cited on pages 51 and 192.)

[Voudouris 1998] Christos Voudouris. *Guided Local Search - An Illustrative Example in Function Optimisation.* In In BT Technology Journal, Vol.16, No.3, pages 46–50, 1998. (Cited on page 53.)

[Weise 2009] Thomas Weise. Global Optimization Algorithms - Theory and Application. Self-Published, second édition, 2009. Online available at http://www.it-weise.de/. (Cited on pages xiii and 38.)

[Wright *et al.* 1991] Alden H Wright*et al.* *Genetic algorithms for real parameter optimization.* Foundations of genetic algorithms, vol. 1, pages 205–218, 1991. (Cited on page 60.)

[Zeleny 1982] M. Zeleny. Multiple Criteria Decision Making. McGraw-Hill, New York, 1982. (Cited on pages 8, 24, 25, 32, 188 and 190.)

[Zeng *et al.* 2013a] Rong-Qiang Zeng, Matthieu Basseur and Jin-Kao Hao. *Hypervolume-Based Multi-Objective Path Relinking Algorithm.* In Robin C. Purshouse, Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco and Jane Shaw, editeurs, EMO, volume 7811 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2013. (Cited on page 69.)

[Zeng *et al.* 2013b] Rong-Qiang Zeng, Matthieu Basseur and Jin-Kao Hao. *Solving bi-objective flow shop problem with hybrid path relinking algorithm.*

Appl. Soft Comput., vol. 13, no. 10, pages 4118–4132, 2013. (Cited on page 69.)

[Zhang & Li 2007] Qingfu Zhang and Hui Li. *MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition.* IEEE Trans. Evolutionary Computation, vol. 11, no. 6, pages 712–731, 2007. (Cited on pages iii, v, 4, 16, 79, 92, 93, 96, 98, 107, 109, 121, 138, 161, 162, 164, 188, 194, 196, 204, 206 and 207.)

[Zhang *et al.* 2003] Qingfu Zhang, Jianyong Sun, Edward Tsang and John Ford. *Combination of guided local search and estimation of distribution algorithm for quadratic assignment problems.* In GECCO'2003 Genetic and Evolutionary Computation Conference, pages 42–48, 2003. (Cited on page 53.)

[Zhang *et al.* 2009] Mingming Zhang, Shuguang Zhao and Xu Wang. *Multiobjective evolutionary algorithm based on adaptive discrete differential evolution.* In Proceedings of the Eleventh conference on Congress on Evolutionary Computation, CEC'09, pages 614–621, Piscataway, NJ, USA, 2009. IEEE Press. (Cited on pages 67, 120, 122, 125, 140 and 198.)

[Zimmermann 1991] H.J. Zimmermann. Fuzzy Set Theory and its applications. Kluwer Academic, Dordrecht, 1991. (Cited on page 8.)

[Zitzler & Künzli 2004] Eckart Zitzler and Simon Künzli. *Indicator-based selection in multiobjective search.* In Parallel Problem Solving from Nature-PPSN VIII, pages 832–842. Springer, 2004. (Cited on page 194.)

[Zitzler & Thiele 1999] Eckart Zitzler and Lothar Thiele. *Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach.* IEEE Trans. Evolutionary Computation, vol. 3, no. 4, pages 257–271, 1999. (Cited on pages 4, 16, 20, 21, 78, 79, 81, 109, 146, 188, 190, 193, 194, 198, 200, 203 and 207.)

[Zitzler *et al.* 2000] Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele. *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.* Evolutionary Computation, vol. 8, no. 2, pages 173–195, 2000. (Cited on pages 4, 23, 164, 190, 198 and 207.)

[Zitzler *et al.* 2001] E. Zitzler, M. Laumanns and L. Thiele. *SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization.* In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou and T. Fogarty, editeurs, Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, pages 95–100, Athens,

Greece, 2001. International Center for Numerical Methods in Engineering. (Cited on pages 4, 79, 81, 92, 107, 126, 188 and 194.)

[Zitzler *et al.* 2003] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca and Viviane Grunert da Fonseca. *Performance assessment of multiobjective optimizers: an analysis and review.* IEEE Trans. Evolutionary Computation, vol. 7, no. 2, pages 117–132, 2003. (Cited on pages 4, 23 and 190.)

[Zitzler *et al.* 2004] Eckart Zitzler, Marco Laumanns and Stefan Bleuler. *A tutorial on evolutionary multiobjective optimization.* In Metaheuristics for Multiobjective Optimisation, pages 3–37. Springer, 2004. (Cited on pages 76, 77 and 193.)

[Zitzler *et al.* 2008] Eckart Zitzler, Joshua Knowles and Lothar Thiele. *Quality Assessment of Pareto Set Approximations.* In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen and Roman Slowinski, editeurs, Multiobjective Optimization, volume 5252 of *Lecture Notes in Computer Science*, pages 373–404. Springer Berlin Heidelberg, 2008. (Cited on pages 20 and 189.)