

**The University of Reading**

Department of Computer Science

---

**Local-Search and Hybrid  
Evolutionary Algorithms for  
Pareto Optimization**

Joshua D. Knowles

---

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy  
in Computer Science. Department of Computer Science, University of Reading, Reading,  
RG6 6AY, UK. January 2002.

## **Declaration**

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

## Abstract

In recent years, a gradual increase in the sophistication of multiobjective evolutionary algorithms (MOEAs) for Pareto optimization has been seen, accompanied by an ever-growing list of applications. Despite this trend, however, the proposition that methods based on local search may be a good alternative approach — with the advantages of ease-of-use and lower computational overhead — has not been thoroughly tested. In this thesis we develop a novel, local-search algorithm for Pareto optimization, called PAES, enabling us to test and compare MOEAs against this philosophically different search method.

To help perform this testing, we develop new statistical performance metrics for evaluating collections of approximations to the Pareto set, based on a critical review of currently available methods. Using these metrics, we find that PAES performs well in comparison with popular, modern MOEAs, on a variety of test function and real-world telecommunications problems. These results suggest that local-search-based methods for Pareto optimization do merit further investigation.

Some of the elements of PAES are also more generally applicable for use in the design of other algorithms. In particular, the archiving strategy used by PAES, which incorporates rules to bound the number of solutions stored, and to ensure that they are distributed widely and evenly in objective space, may be used in any multiobjective search algorithm for storing solutions. We demonstrate the generality of the basic PAES procedures by outlining several more sophisticated variants, including multi-start and tabu search algorithms.

We also combine local-search and population-based evolutionary methods together, forming a new memetic algorithm, M-PAES, for Pareto optimization. M-PAES is tested using a number of problems from the literature, and is found to exhibit promising performance compared to other methods. Finally, we use M-PAES to provide a set of benchmark results for some difficult new instances of the multi-criteria minimum spanning tree (mc-MST) problem.

*to Hélène*

# Acknowledgments

Usually, I'm not very good at saying thank you: I tend to say it fast and mumble. However, as this thesis is already over 300 pages, I am not inclined to save space in this section. So, to everyone who has helped me over the last years, here is your "thank you".

First and foremost, I would like to thank my PhD supervisor, David Corne. He has provided me with a free exchange of ideas, constructive criticism, encouragement, advice, and moral support, as well as many opportunities for professional development, and practical and financial help. Above all, I thank him for the many "reality checks" he has offered me over the last years, and for the bizarre sense of humour he has used to deflect much of my negativity.

I thank Martin Oates for providing an excellent and friendly interface with my sponsors, BT, for many "Hungry Horse" dinners and other freebies, and good collaborative work. Martin is definitely one of the most good-humoured and generous people I have met over the last years.

Thanks to Graham Megson, Roger Loader, and Rachel Harrison for additional guidance and Kalyanmoy Deb and Kevin Warwick for agreeing to examine the thesis.

To the department secretaries, in particular, Val Dobree, Charmaine Birchmore, Christina Simmons, and Pat Morrison, thanks for all your help. Thanks also to John Dawson and Glennys Dodge.

Thank you to the technical support staff, especially Ian Bland.

Thanks to Gerard McKee for quietly lending me the use of G33 for the last year. It has been a peaceful place in which to work, and most helpful for my finishing up.

To all those in the evolutionary computation community who have offered me advice, guidance, friendship, or professional opportunities and assistance, especially Eric Bonabeau, Jürgen Branke, Carlos Coello, Paul Darwen, Kalyanmoy Deb, Marco Dorigo, Sevan Ficici, Carlos Fonseca, Hugue Juillé, Natalio Krasnogor, Gary Lamont, Marco Laumanns, Pablo Moscato, Jason Noble, Ricardo Poli, Ian Parmee, Günter Raidl, Günter Rudolph, Jane Shaw,

Brian Turton, David van Veldhuizen, Xin Yao, Eckart Zitzler, and Jesse Zydallis, thank you.

A special thank you also to Richard Watson for many interesting and enlightening conversations, some good beer-drinking sessions, and valuable collaborative work.

I would also like to thank all my ‘university’ friends, particularly Ian Bland and Liz Nickels, Bernard Brooks, Magnus Byne, Oswaldo Cadenas-Medina, Lee Cooper, James Ferryman, Rob Fish, Stephanie Fountain, Tom Grove, Martyn Lewis and Heather Craddock, and Jim Pascoe. You helped make my university life (and my Friday nights) most enjoyable.

And I thank all my ‘non-university’ friends for providing a welcome distraction from work over the last years, although, eventually, most of you did succumb to the awful habit of asking me when I would be finished. Thanks especially to Julian Palmer, Andrew and Donna Wheatley, Barzan Rahman, Paul Meltzer, Matt Tully, Leon Bartlett, Jason Forster and Helen Fretwell, Mark and Kathy Burgess, Daniel and Pauline Martin, Chris Pownall and Alex Cockerill, and Carl Maycock.

Thanks very much, Bernard and Michelle Grislin, for your generosity, concern and warmth.

Finally, I would like to thank my parents and sisters for providing me with continued support, encouragement and love.

# List of Publications<sup>1</sup>

## Patents

- [KC00] Joshua Knowles and David Corne. Optimisation method. European Patent EP00305549.8, 2000. Date filed: 30th June.

## Journals

- [KC00a] Joshua Knowles and David Corne. A new evolutionary approach to the degree-constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4(2):125–134, July 2000.
- [KC00b] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
- [KC01] Joshua D. Knowles and David W. Corne. Enumeration of Pareto optimal multi-criteria spanning trees — a proof of the incorrectness of Zhou and Gen’s proposed algorithm. *European Journal of Operational Research*, 2001. (In press).
- [KOC00] Joshua D. Knowles, Martin J. Oates, and David W. Corne. Advanced multiobjective evolutionary algorithms applied to two problems in telecommunications. *BT Technology Journal*, 18(4):51–65, October 2000.

## Conference Proceedings

- [CJKO01] D. Corne, N. Jerram, J. Knowles, and M. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of GECCO-2001: Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kaufmann, 2001.
- [CK00] David W. Corne and Joshua D. Knowles. The Pareto-envelope based selection algorithm for multiobjective optimization. In *Proceedings of the Sixth International*

---

<sup>1</sup>The publications listed here were completed during the author’s period of registration on the higher degree program. Other publications by the author are not included.

*Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 839–848, Berlin, 2000. Springer-Verlag.

- [KC99] Joshua D. Knowles and David W. Corne. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.
- [KC00a] Joshua D. Knowles and David W. Corne. Heuristics for evolutionary off-line routing in telecommunications networks. In Darrell Whitley *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 574–581, San Francisco, CA, 2000. Morgan Kaufmann.
- [KC00b] Joshua D. Knowles and David W. Corne. M-PAES: A memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*, pages 325–332, Piscataway, NJ, 2000. IEEE Press.
- [KC01a] Joshua D. Knowles and David W. Corne. Benchmark problem generators and results for the multiobjective degree-constrained minimum spanning tree problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 424–431, 2001.
- [KC01b] Joshua D. Knowles and David W. Corne. A comparison of encodings and algorithms for multiobjective minimum spanning tree problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC'01)*, pages 544–551. IEEE Press, 2001.
- [KC09] Joshua Knowles, David Corne, and Martin Oates. A new evolutionary approach to the degree constrained minimum spanning tree problem. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, page 794. Morgan Kaufmann, 1999.
- [KC00] Joshua D. Knowles, David W. Corne, and Martin J. Oates. On the assessment of multiobjective approaches to the adaptive distributed management problem. In *Proceedings of the Sixth International Conference of Parallel Problem Solving From Nature (PPSN VI)*, pages 869–878, Berlin, 2000. Springer-Verlag.
- [KWC01] Joshua D. Knowles, Richard A. Watson, and David W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *Evolutionary Multi-Criterion Optimization : first international conference; proceedings / EMO 2001*, volume 1993 of *LNCS*, pages 269–283. Springer, 2001.

## Edited Book Contributions

- [KC00] Joshua D. Knowles and David W. Corne. Evolving neural networks for cancer radiotherapy. In Lance D. Chambers, editor, *Practical Handbook of Genetic Algorithms: Applications Second Edition*, pages 443–488. Chapman Hall/CRC Press, Boca Raton, Florida, 2000.

## Workshop Proceedings

- [KC99a] J. D. Knowles and D. W. Corne. Local search, multiobjective optimization and the Pareto archived evolution strategy. In Bob McKay, Xin Yao, Ruhul Sarker, Yasuhiro Tsujimura, Akira Namateme, and Mitsuo Gen, editors, *Proceedings of Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 209–216, Ashikaga, Japan, November 1999. University of New South Wales and Ashikaga Institute of Technology.
- [KC99b] Joshua D. Knowles and David W. Corne. Assessing the performance of the Pareto archived evolution strategy. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, pages 123–124, Orlando, Florida, July 1999.
- [KC00a] Joshua D. Knowles and David W. Corne. A comparison of diverse approaches to memetic multiobjective combinatorial optimization. In Annie S. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 103–108, 2000.
- [KC00b] Joshua D. Knowles and David W. Corne. Multiobjective approaches to the adaptive distributed database management problem. In David Corne, editor, *Proceedings of the SAB/PPSN Workshops (Evolutionary Computation in Telecommunications Workshop)*, 2000.
- [KC01] Joshua D. Knowles and David W. Corne. A comparative assessment of memetic, evolutionary, and constructive algorithms on multiobjective  $d$ -MST problems. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program*, pages 162–167, 2001.

# Contents

|  |           |
|--|-----------|
| <b>Dedication</b>                        | <b>i</b>  |
| <b>Acknowledgments</b>                   | <b>ii</b> |
| <b>List of Publications</b>              | <b>iv</b> |
| <b>1 Introduction</b>                    | <b>1</b>  |
| 1.1 Overview . . . . .                   | 1         |
| 1.1.1 Two broad classes . . . . .        | 3         |
| 1.1.2 Baselineing is important . . . . . | 4         |
| 1.1.3 Find all the compromises . . . . . | 4         |
| 1.1.4 A new approach to try . . . . .    | 6         |
| 1.1.5 Hybridize where possible . . . . . | 7         |
| 1.1.6 Contributions . . . . .            | 8         |
| 1.2 Outline of the thesis . . . . .      | 9         |
| <b>2 Approximate Methods of Search</b>   | <b>13</b> |
| 2.1 Overview . . . . .                   | 13        |
| 2.1.1 Combination lock . . . . .         | 14        |
| 2.1.2 Wipeout . . . . .                  | 15        |

|       |   |    |
|-------|---|----|
| 2.1.3 | Super-wipeout                                     | 17 |
| 2.2   | Problems and complexity                           | 20 |
| 2.3   | Search algorithms                                 | 23 |
| 2.3.1 | Heuristics  | 23 |
| 2.3.2 | Neighbourhood search                              | 24 |
| 2.3.3 | Random mutation hillclimbing                      | 24 |
| 2.3.4 | Simulated annealing                               | 25 |
| 2.3.5 | Tabu search                                       | 26 |
| 2.3.6 | Genetic algorithms                                | 27 |
| 2.3.7 | Evolution strategies and evolutionary programming | 29 |
| 2.3.8 | Other metaheuristics                              | 29 |
| 2.4   | Generality  | 30 |
| 2.5   | Problem features and fitness landscapes           | 31 |
| 2.5.1 | Discrete and continuous variables                 | 32 |
| 2.5.2 | Dimensionality                                    | 32 |
| 2.5.3 | Multimodality                                     | 33 |
| 2.5.4 | Discontinuity                                     | 33 |
| 2.5.5 | Deception   | 34 |
| 2.5.6 | Epistasis and linear separability                 | 35 |
| 2.5.7 | Constraints                                       | 36 |
| 2.6   | No-Free-Lunch and its implications                | 36 |
| 2.6.1 | Test functions and GA performance                 | 38 |
| 2.6.2 | Real-world optimization problems                  | 40 |
| 2.7   | Hybrid evolutionary approaches                    | 43 |
| 2.8   | Summary   | 45 |

|   |           |
|---|-----------|
| <b>3 Pareto Optimization</b>  | <b>47</b> |
| 3.1 Introduction and overview . . . . .   | 47        |
| 3.2 Concepts and notation . . . . .   | 49        |
| 3.2.1 Pareto optimization . . . . .   | 50        |
| 3.3 Approximating the efficient set . . . . .                                     | 52        |
| 3.4 General-purpose Pareto optimization schemes . . . . .                         | 53        |
| 3.4.1 Criterion selection . . . . .   | 53        |
| 3.4.2 Pareto selection . . . . .  | 53        |
| 3.4.3 Scalarizing selection . . . . .   | 54        |
| 3.4.4 A two-dimensional classification of methods . . . . .                       | 55        |
| 3.5 Elitism in multiobjective evolutionary algorithms . . . . .                   | 57        |
| 3.6 Obtaining a well-distributed approximation set . . . . .                      | 59        |
| 3.6.1 Niching and diversity maintenance in EAs and MOEAs . . . . .                | 60        |
| 3.6.2 Encouraging diversity in methods based on scalarizing selection . . . . .   | 64        |
| 3.7 Where to go from here . . . . .   | 64        |
| 3.8 Multiobjective problems . . . . .   | 67        |
| 3.9 MOEA test functions and problems . . . . .                                    | 68        |
| 3.10 An introduction to performance assessment and comparison . . . . .           | 69        |
| 3.11 Classification and analyses of nondominated set assessment metrics . . . . . | 72        |
| 3.11.1 The classification scheme . . . . .  | 72        |
| 3.11.2 The $\mathcal{S}$ metric [Zit99] . . . . .                                 | 74        |
| 3.11.3 Error ratio [Vel99] . . . . .  | 75        |
| 3.11.4 Generational distance [Vel99] . . . . .                                    | 78        |
| 3.11.5 Maximum Pareto front error [Vel99] . . . . .                               | 79        |
| 3.11.6 Overall nondominated vector generation (ONVG) [Vel99] . . . . .            | 80        |

|          |  |            |
|----------|--|------------|
| 3.11.7   | Overall nondominated vector generation ratio (ONVGR) [Vel99]                                     | 81         |
| 3.11.8   | Other metrics proposed in [Vel99]  | 82         |
| 3.11.9   | Spacing metric (Deb <i>et al.</i> [DAPM00a])   | 82         |
| 3.11.10  | Spacing metric (Schott [Sch95])  | 83         |
| 3.11.11  | The $\mathcal{C}$ metric   | 84         |
| 3.11.12  | $D1_R$ (Czyzak and Jaszkiewicz)  | 87         |
| 3.11.13  | $R1$ and $R1_R$ (Hansen and Jaszkiewicz)   | 89         |
| 3.11.14  | $R3$ and $R3_R$ (Hansen and Jaszkiewicz)   | 91         |
| 3.11.15  | Summary of analysis  | 91         |
| 3.12     | Assessment methods used in this thesis   | 93         |
| 3.12.1   | Methods based on attainment surface sampling   | 93         |
| 3.12.2   | Methods based on the $\mathcal{S}$ Metric  | 97         |
| <b>4</b> | <b>The Pareto Archived Evolution Strategy (PAES)</b>   | <b>102</b> |
| 4.1      | Requirements for a Pareto hillclimber  | 102        |
| 4.2      | An archiving hillclimber   | 106        |
| 4.3      | Evaluation of archiving strategies   | 107        |
| 4.3.1    | Definitions of terms used in the analysis  | 110        |
| 4.3.2    | Archiving strategy 1: unbounded archive  | 111        |
| 4.3.3    | Archiving strategy 2: for a bounded archive, accept only dominating vectors when archive is full | 113        |
| 4.3.4    | Archiving strategy 3: $\mathcal{S}$ metric archiving   | 115        |
| 4.3.5    | Archiving strategy 4: adaptive grid archiving  | 118        |
| 4.4      | Efficient implementation of an adaptive grid archiving algorithm                                 | 140        |
| 4.5      | Enhancing the acceptance function  | 144        |
| 4.6      | (1+1)-PAES   | 145        |

|          |   |            |
|----------|---|------------|
| 4.7      | The time complexity of (1+1)-PAES . . . . .             | 146        |
| 4.8      | Other variants of PAES . . . . .                        | 148        |
| 4.8.1    | Multi-start PAES . . . . .                              | 148        |
| 4.8.2    | Thresholding and annealing PAES . . . . .               | 150        |
| 4.8.3    | Tabu search PAES . . . . .                              | 151        |
| 4.9      | Summary . . . . .                                       | 153        |
| <b>5</b> | <b>Empirical Testing of PAES</b>                        | <b>156</b> |
| 5.1      | Overview . . . . .                                      | 156        |
| 5.2      | Initial comparative assessment of PAES . . . . .        | 156        |
| 5.2.1    | Test functions $F_1$ – $F_5$ . . . . .                  | 156        |
| 5.2.2    | Test problem $F_6$ . . . . .                            | 159        |
| 5.2.3    | Population-based variants of PAES . . . . .             | 161        |
| 5.2.4    | Rival MOGAs tested . . . . .                            | 162        |
| 5.2.5    | Results and discussion . . . . .                        | 163        |
| 5.3      | Assessment of PAES using Deb's test functions . . . . . | 174        |
| 5.3.1    | Deb's test functions . . . . .                          | 175        |
| 5.3.2    | Experimental results . . . . .                          | 177        |
| 5.3.3    | Summary . . . . .                                       | 192        |
| 5.3.4    | Discussion . . . . .                                    | 194        |
| 5.4      | Summary and conclusions . . . . .                       | 195        |
| <b>6</b> | <b>The Memetic PAES Algorithm (M-PAES)</b>              | <b>197</b> |
| 6.1      | Overview . . . . .                                      | 197        |
| 6.2      | Recent multiobjective memetic algorithms . . . . .      | 198        |
| 6.3      | Pareto memetic algorithm design . . . . .               | 200        |

|          |  |            |
|----------|--|------------|
| 6.4      | Development of M-PAES  | 205        |
| 6.5      | The M-PAES algorithm   | 207        |
| 6.5.1    | Convergence properties of M-PAES                                 | 210        |
| 6.5.2    | Parameter control in M-PAES                                      | 210        |
| 6.6      | Initial study: M-PAES on the multiobjective 0/1 knapsack problem | 211        |
| 6.6.1    | Experimental method  | 211        |
| 6.6.2    | RD-MOGLS   | 213        |
| 6.6.3    | Parameter choices  | 213        |
| 6.6.4    | Results  | 219        |
| 6.6.5    | Summary  | 222        |
| 6.7      | Assessment of M-PAES, (1+1)-PAES, and SPEA on the ADDMP          | 223        |
| 6.7.1    | ADDMP instances  | 223        |
| 6.7.2    | Parameter control and performance assessment                     | 225        |
| 6.7.3    | Results  | 225        |
| 6.7.4    | Concluding remarks   | 227        |
| 6.8      | Summary  | 229        |
| <b>7</b> | <b>Multi-criterion MST (mc-MST) Problems</b>                     | <b>230</b> |
| 7.1      | Introduction   | 230        |
| 7.1.1    | Motivation   | 230        |
| 7.1.2    | The minimum spanning tree problem                                | 231        |
| 7.1.3    | <i>NP</i> -hard MST problems                                     | 232        |
| 7.1.4    | Approaches to the $d$ -MST problem                               | 232        |
| 7.1.5    | The mc-MST problem   | 237        |
| 7.1.6    | Organization of this chapter                                     | 238        |

|          |  |            |
|----------|--|------------|
| 7.2      | The mc-MST problem . . . . .   | 239        |
| 7.3      | Finding the supported efficient solutions . . . . .                  | 240        |
| 7.4      | Representations and operators . . . . .                              | 244        |
| 7.4.1    | The randomized primal method . . . . .                               | 244        |
| 7.4.2    | The Prüfer encoding . . . . .  | 246        |
| 7.4.3    | Direct tree encoding and operators with RPM initialization . . . . . | 246        |
| 7.5      | Benchmark problem generators . . . . .                               | 249        |
| 7.5.1    | Random . . . . .   | 250        |
| 7.5.2    | Correlated and anti-correlated . . . . .                             | 250        |
| 7.5.3    | M-correlated . . . . .   | 251        |
| 7.5.4    | Concave . . . . .  | 254        |
| 7.6      | Problem instances . . . . .  | 257        |
| 7.6.1    | Simple random weight instances . . . . .                             | 257        |
| 7.6.2    | Benchmark problem instances . . . . .                                | 258        |
| 7.7      | Experiments . . . . .  | 259        |
| 7.7.1    | AESSEA . . . . .   | 261        |
| 7.7.2    | Random weight instances . . . . .                                    | 261        |
| 7.7.3    | Benchmark instances with degree constraints . . . . .                | 262        |
| 7.8      | Results . . . . .  | 263        |
| 7.8.1    | Random weight instances . . . . .                                    | 263        |
| 7.8.2    | Benchmark test problem instances . . . . .                           | 265        |
| 7.9      | Conclusion . . . . .   | 268        |
| <b>8</b> | <b>Conclusions</b>   | <b>271</b> |
| 8.1      | Summary . . . . .  | 271        |

|  |            |
|--|------------|
| 8.2 Contributions . . . . .  | 274        |
| 8.2.1 The contribution of PAES . . . . .   | 275        |
| 8.2.2 The adaptive grid archiving strategy . . . . .   | 278        |
| 8.2.3 Metrics for performance assessment . . . . .   | 279        |
| 8.2.4 M-PAES . . . . .   | 279        |
| 8.2.5 MST benchmarks . . . . .   | 280        |
| 8.3 Limitations . . . . .  | 280        |
| 8.3.1 PAES . . . . .   | 280        |
| 8.3.2 M-PAES . . . . .   | 282        |
| 8.3.3 Performance metrics . . . . .  | 282        |
| 8.4 Further work . . . . .   | 283        |
| 8.4.1 Reducing the effect of local optima in search . . . . .  | 283        |
| 8.4.2 Highly multiobjective problems . . . . .   | 284        |
| <b>A Proof that Upper Grid Boundaries Converge</b>   | <b>286</b> |
| <b>B Proof of Theorem 4.4</b>  | <b>289</b> |
| <b>C Derivation of Equation 4.3</b>  | <b>292</b> |
| <b>D Proof of Incorrectness of Zhou and Gen's Proposed Enumeration Algorithm for Pareto Optimal Multi-criterion Spanning Trees</b> | <b>294</b> |
| D.1 Statement of the problem . . . . .   | 294        |
| D.2 Zhou and Gen's proposed enumeration algorithm . . . . .  | 294        |
| D.3 Incorrectness of the proposed approach . . . . .   | 295        |
| <b>Bibliography</b>  | <b>301</b> |

# Chapter 1

## Introduction

### 1.1 Overview

We are all familiar with the activity of searching for something good or ‘right’ from amongst a set of alternatives. To see this, consider trying to do any of the following (more or less) familiar tasks:

- find a good place to have breakfast in an unfamiliar city;
- choose an apartment to live in;
- produce a timetable for all the examinations in a university;
- plan the best route for a car journey;
- make the best choice of play in a game of gin rummy.

Problems like these are ubiquitous not only in our everyday lives, but frequently arise in such diverse disciplines as engineering, architecture, finance, operations research, logistics, and medicine. In tackling these *search problems*, it is clear that while finding a ‘solution’ may not be difficult, finding a good one can be a formidable task. The difficulty of the task may be due to limitations on the time or resources available for considering the alternatives, the conflicting nature of our requirements, or the sheer abundance of alternatives from which to choose.

In approaching any search problem, one hurdle is simply defining the meaning of ‘good’ in the context of the problem, and many search problems require us to find solutions that satisfy or

optimize multiple objectives simultaneously. For example, finding a good timetable for university examinations involves the consideration of numerous constraints and the minimization of several ‘costs’. These costs might include things such as the number of students taking consecutive exams, the number of students taking more than two exams per day, the total duration of the examinations period, the total number of rooms used, the total number of staff needed for invigilation, the overtime that must be paid to staff for working extra hours<sup>1</sup>, and so on. To balance these different objectives, according each the right level of importance, and choosing the optimum or best compromise may be a difficult decision-making task.

Finding a solution ‘in time’ or ‘in budget’ may also be important. In some search problems, we can actually try out alternatives, whereas in others we can simulate trying them, perhaps by estimating their quality in some way, in our mind’s eye. For example, finding an apartment will probably involve visiting some and then imagining living there (and imagining paying the rent), whereas selecting a good breakfast establishment will probably involve actually eating there (and actually paying for the food). In either case, there is a sense in which considering each alternative costs something, so that part of the difficulty in finding suitable solutions arises from restrictions on the time, effort or resources that can be spent in searching. This situation, which is almost universal in nontrivial search tasks, leads to a tradeoff of quality of solutions versus effort or time in searching.

Because restrictions in time or effort can be such an important factor in determining if we are successful in our searches, we tend to conduct them differently depending on our estimates of how important it is to get answers quickly or economically, and how much we can afford to compromise on our goal. But how do we modulate our searches to account for these restrictions? Broadly speaking, when we have plenty of time and resources, we tend to *explore* more alternatives, and search in an open-minded manner. We don’t prejudice the search by looking only in ‘areas’ where we think good solutions are likely to be found. On the other hand, when time or resources are more restricted, we are correspondingly more restricted in our thinking and in our choices. Because we can only afford to look at a few alternatives we try to choose each one more carefully, often by *exploiting* the knowledge we have already gathered in our search. For example, when looking for an apartment, one might look only in a restricted geographical area once some good ones have been found there, reasoning that looking elsewhere is not as likely to be fruitful.

Clearly then, there is a balance to be struck between exploration and exploitation, given a particular search problem, and given restrictions of time or effort. In some problems it will be very hard to exploit the knowledge gathered during the search to help ‘construct’ new

---

<sup>1</sup>This is of course ironic since academics do not get paid overtime.

alternatives. For example, if I am searching for the best play in a game of rummy, there may be little sense in which one play is like another, making it difficult to exploit information gathered from considering or imagining the alternative plays. I may as well try, in my mind's eye, the different alternatives at random<sup>2</sup>. On the other hand, having found a reasonable route for a car journey, finding another one that might be better is more likely to result from changing the reasonable route slightly, than from trying a route which uses roads that we have no information about. Thus the most appropriate method of searching alternatives depends both on the type of problem at hand, and the restriction of time or resources. That is, both *efficacy* and *efficiency* may be important, and choosing a method which strikes the right balance will depend critically on the problem.

### 1.1.1 Two broad classes

This thesis is concerned with *general-purpose optimization algorithms* (GPOAs), which are general methods for solving search problems, like the problems introduced above. Although general-purpose optimization algorithms exist today in many, many different forms (a reflection of their different origins in various fields of study) they all search for solutions in essentially the same way. That is, they try out different alternatives, evaluate their worth (perhaps by estimation) and then repeat this procedure using what has been learnt to guide them. What differentiates the numerous algorithms, then, is the way that they utilize the information from the solutions they have explored in the past, and that they have in some sense “remembered”, in order to find or construct better solutions in the future.

Although there is much variety in these methods, it is helpful to classify GPOAs into two broad classes: single point local-search methods — where a single solution is improved incrementally by making small changes to it; and population-based methods — where a population of solutions are ‘evolved’ in parallel, and improvement results from repeatedly making variations of those solutions in the population that are more highly ‘fit’, and discarding those that are less fit. Hillclimbing, simulated annealing, and tabu search are well-known methods based on single-point local search, while most evolutionary algorithms (EAs) exemplify population-based methods. This thesis will be concerned with the use of these different approaches when applied to multiobjective search problems.

---

<sup>2</sup>To some extent this is a matter of finding the right way to describe or *represent* the alternative plays in my mind (so that they can be classified in a useful way) but this is a nontrivial problem in itself, equal to the search problem.

### 1.1.2 Baselining is important

As discussed above, the relative ‘performance’ of different methods of search (and thus different GPOAs) depends critically upon the search problem under consideration, and the balance between the requirements of efficacy versus efficiency.

For this reason, evolutionary algorithm practitioners have been obliged, historically, to put the performance of EAs to the test, by making comparisons with other search algorithms, particularly those based on single-point local search, such as hillclimbers, and simulated annealing. This is sometimes called ‘baselining’ and it is necessary because one should not take the performance of an EA or any other algorithm for granted. Moreover, numerous test problems and much of EA theory have been developed with the express purpose of analyzing and explaining algorithm performance differences on various problem types.

This practice of comparing and analyzing the performance of different optimization algorithms, seen in much EA literature, is important and useful for a number of reasons. First, it builds up a body of knowledge concerning which algorithms are best suited to which problems. Second, it may help predict — for future unseen problems — which algorithms may perform best, although this goal depends also upon the success or failure of attempts to usefully classify problems via measurable features. And third, the practice sometimes contributes to the understanding of the particular problem on which comparisons are being made, and can facilitate the development of improved approaches; either approaches particularly suited to that problem, or improved algorithms that are more generally applicable. Frequently these new approaches are hybrids of local-search *heuristics* and population-based methods.

In this thesis, we make contributions that should help to further the practice of comparing and analyzing algorithm performance differences. To this end, we propose new local-search algorithms that offer an alternative to EAs, make improvements to methods of measuring and comparing search performance, and develop some test problems. However, in making these contributions, our focus is restricted to a specific domain, called Pareto optimization, described next.

### 1.1.3 Find all the compromises

Pareto optimization relates to the ‘solving’ of *multiobjective* search problems, and at the present time, EAs are a very popular approach to it. More specifically, Pareto optimization is a method of tackling multiobjective problems without the need for weighting the importance of different objectives or normalizing them in any way, or in fact knowing anything about

what solutions are ‘out there’ to be found. In Pareto optimization the goal is to find the set of all ‘best tradeoff’ solutions, the Pareto optimal set. This set has a mathematically well-defined meaning and is independent of the relative importance of the various problem objectives.

When GPOAs — approximate methods — are used, the goal of Pareto optimization is to approximate the Pareto optimal set as closely as possible. That is, the aim is not to find a single, very good solution, as in single-objective optimization, but rather to find a whole set of solutions which offer different compromises. Now, in some sense a population-based EA seems purpose-built for Pareto optimization because its population is a natural structure with which to approximate the Pareto set, and hence little is needed to adapt the method. For this reason, some researchers have claimed that population-based EAs are naturally suited to the Pareto optimization task, with perhaps some implication that other methods may be generally inferior. However, whether or not EAs are easy to apply to this task, general principles in search and optimization suggest that on many problems simpler methods based on single point local-search should provide more effective and efficient search performance; the need to search for different tradeoff solutions and store them can be addressed by other additional mechanisms. Furthermore, local-search methods may be simpler to use and may involve less computational overhead per function evaluation than their evolutionary algorithm counterparts.

Local-search methods can be adapted to the Pareto optimization task in various ways, some of which have already been investigated by other researchers. The main requirement is a change to the acceptance function so that partially ordered solutions can be processed. In addition, a store of solutions must be kept so that a set of nondominated alternatives can be returned. Several articles in the operational research (OR) literature have proposed local searchers for multiobjective optimization, including SA [CJ98, Han97a, Ser94] and tabu search (TS) [GMF97, Han97b] algorithms. These methods use parameterized scalarizing functions to obtain a single evaluation of a solution’s quality from its true vector (multiobjective) cost, thus allowing a standard acceptance function to be used without change. To obtain a whole Pareto front from a single run, these methods must vary the parameters of the scalarizing functions, so that search occurs in different ‘directions’ in the objective function space. This variation of the search direction is generally random, and does not respond to the location (in objective space) of solutions previously found, although some more intelligent methods have also been proposed [CJ98]. Normally, at least one reference point in objective space is required for these scalarizing methods to work, and objectives may have to be scaled and normalized for best performance to be achieved, otherwise the scalarizing functions can favour search in some objectives more than others. Furthermore, poor performance on non-convex

Pareto fronts can occur if weighted-sum aggregation is used as the scalarizing function, as is the case in many of the proposed algorithms.

#### 1.1.4 A new approach to try

The work in this thesis presents a different approach to local-search Pareto optimization. The key contribution is a novel search algorithm, called the Pareto archived evolution strategy (PAES<sup>3</sup>). In its most basic form, called (1+1)-PAES, it represents the Pareto optimization analogue of a single-objective hillclimber. Like a hillclimber, (1+1)-PAES uses single point local search, i.e., a (1+1) selection strategy. However, it uses an acceptance function based on Pareto dominance, avoiding the use of scalarizing functions and the need for estimating reference points in objective space. Because it must return a diverse set of nondominated solutions, PAES also uses an ‘archiving’ strategy for storing and maintaining nondominated solutions as it runs. A number of alternative archiving strategies are considered, and their convergence properties are evaluated. A strategy employing an adaptive region-based crowding mechanism is shown to converge to a well-distributed set of objective vectors under certain conditions. An efficient implementation of the method is devised and its time complexity is analyzed. The analysis of these archiving strategies contributes generally to the question of how solutions should be stored and maintained in all Pareto optimization algorithms. The basic procedures developed for use in (1+1)-PAES also facilitate the design of other, more sophisticated local-search Pareto optimizers. In this thesis, these procedures are used to specify multi-start, population-based, simulated annealing and tabu search variants of PAES, which all use similar archiving and acceptance rules to the basic (1+1)-PAES algorithm.

Armed with PAES, the performance of multiobjective EAs (MOEAs) can be compared against a ‘hillclimber’ for the first time. However, to do this, effective metrics for assessing performance must be used. Due to the nature of Pareto optimization, the output of a single run of an optimizer is a set of solutions (an approximation set), with each solution having a vector of objective values. Thus, even the output of a single run of a multiobjective optimizer is a multidimensional attribute that is difficult to assess. To address this issue, we review several popular assessment metrics from the literature, analyzing them in terms of a number of distinct properties. Our analysis shows that some methods used in the literature are not compatible with relations derived solely from the principles of Pareto optimization [HJ98]. Others are clearly more satisfactory, and these are used as a basis for devising new approaches that can be used to analyze and compare the performance of algorithms over multiple runs. These proposed methods are used to assess the performance of (1+1)-PAES and other algorithms

---

<sup>3</sup>Pronounced “pays”.

in this thesis.

A number of empirical investigations using our proposed metrics reveal that the performance of (1+1)-PAES is competitive with several popular MOEAs from the literature, on a range of well-known test functions, as well as some newer ones. On the more simple functions, (1+1)-PAES can actually outperform some modern MOEAs, given an equal number of function evaluations. Furthermore, timings indicate that (1+1)-PAES is faster than some other MOEAs, and because it requires only a small number of non-critical parameter choices, it is also easier to use. These attributes indicate that PAES may be a useful ‘baseline’ algorithm against which more sophisticated techniques could be compared. On the other hand, some evidence shows that the (1+1)-PAES algorithm *does not* perform as effectively as the strength Pareto EA (SPEA) [Tzb99] when optimizing some multimodal and deceptive problems, indicating that population-based MOEAs may well exhibit more robust behaviour on functions of this type.

### 1.1.5 Hybridize where possible

The PAES algorithm is also used as a starting point to develop a new hybrid MOEA, that combines local and population based search. When practical optimization problems are tackled, general-purpose algorithms may fail to deliver adequate performance. Davis reminds us to: “hybridize where possible” [Dav91a], if we want to improve the results achieved by a standard genetic algorithm (GA). This reflects the fact that although good performance can be often achieved by using the most appropriate general-purpose optimization algorithm, much greater gains can often be made by combining it with specific heuristics or operators that incorporate ‘domain knowledge’. On many difficult, well-studied problems, the best results come from such hybrid approaches where very specific heuristics are combined with well-proven overall strategies.

However, despite the message of Davis, it is generally easier to hybridize problem-specific heuristics with a local-search based method than it is with a genetic algorithm, employing recombination. This is because many heuristics use a form of local improvement, but few use any sort of recombination. Nonetheless, it is true that recombination can be a valuable additional operator to use. In these cases, hybrid genetic algorithms — or memetic algorithms (MAs) — have proved an effective approach. In MAs, there is a local-search phase which is often hybridized with some other heuristic technique for the specific problem, and a recombinative phase that can quickly combine parts of promising solutions together, to form better ones.

The PAES local-search procedure provides, for the first time, the opportunity to construct memetic algorithms for multiobjective optimization, based purely on Pareto selection. We propose a framework for doing this, and specify an instance of the framework, a memetic algorithm called M-PAES, based on the PAES algorithm. M-PAES uses single-point local search phases as a mechanism for improving solutions in a population that is undergoing selection and recombination.

M-PAES is compared with (1+1)-PAES, SPEA [ZT99], and a memetic algorithm that uses scalarizing selection, called RD-MOGLS [Jas99]. We find the performance of M-PAES to be promising on a set of 2, 3, and 4-objective knapsack problems, used previously by Zitzler [ZT99]. On a number of problems based on load-balancing in distributed databases, however, M-PAES does not find better solutions than (1+1)-PAES, indicating, once again the power of simple strategies, on some problems.

In a more extensive study, heuristics are developed specifically for an *NP*-hard application, the multi-criterion minimum spanning tree (mc-MST) problem. These heuristics are applied in (1+1)-PAES and M-PAES and their performance is compared against a more standard MOEA. The mc-MST instances used for testing these algorithms are produced using a new parameterized benchmark problem generator designed by us to make challenging instances. These exhibit a range of features including non-convex Pareto fronts, constraints and deception. Results show that M-PAES is able to build significantly lighter spanning trees than the MOEA, and PAES, in general, across a set of fifteen instances.

### 1.1.6 Contributions

In summary, the contributions of this thesis are:

- A functionally simple, computationally inexpensive, local-search algorithm for Pareto multiobjective optimization, called (1+1)-PAES, which is competitive, on a variety of problems, with many previously proposed evolutionary algorithms of greater computational complexity. This algorithm is useful as a baseline algorithm for benchmarking new approaches, as a local-search procedure for use in a hybrid evolutionary algorithm, and, in its own right, as an efficient search tool for some problems.
- Several variants of (1+1)-PAES, including simulated annealing and tabu search versions.
- Proofs relating to the convergence properties of a number of solution archiving strategies for use in multiobjective search algorithms.

- A framework for a memetic algorithm for Pareto multiobjective optimization, and an instance of the framework, called M-PAES, based on the PAES local search procedure.
- Advances in methods for the statistical comparison of multiobjective optimizers.
- Empirical studies comparing various local-search, hybrid, and population-based approaches on a range of test functions and real-world applications.
- New problem generators for constructing difficult instances of the multi-criterion minimum spanning tree problem, and benchmark results for a suite of these problem instances.

## 1.2 Outline of the thesis

This thesis is divided into eight chapters, beginning with this introduction. Two literature review chapters follow, with the second one including some original contributions relating to multiobjective performance metrics. Chapters 4, 5, 6, and 7 contain the remainder of the original material. Chapter 4 deals with PAES, 5 with empirical testing of PAES, 6 with the development and testing of M-PAES, and 7 with the mc-MST problem. Chapter 8 concludes. In the following, we outline the content of each of these chapters in greater detail.

The review of current literature relevant to this thesis starts in Chapter 2, which considers general-purpose optimization methods. It begins with a discussion of the nature of general problem-solving ‘heuristics’, introducing some of the main themes of the chapter. We indicate how, with some problems, enumeration is the best we can hope to do, whereas for some others, approximate methods — heuristics — can find a good solution much more rapidly. This broad overview of search is followed by an introduction to the theory of *NP*-completeness and its relation to the computational complexity of optimization problems. This theory is behind the need for approximate methods of search and motivates the drive for better general-purpose methods. The latter are considered in the next section, with a description of each of the most influential and popular optimization methods for general-purpose problem-solving. Hillclimbing, simulated annealing, tabu search, genetic algorithms, evolutionary programming and evolution strategies are all considered. We highlight the origins and differences of these methods, paying some attention to their individual strengths and weaknesses. Following this, we attempt to catalog some of the problem features that are currently (partially) understood in optimization, and we relate them to optimization difficulty. We also consider the No Free Lunch (NFL) theorem and what it says about matching algorithms to problems. In the closing sections of the chapter, we review literature describing empirical performance

studies, paying particular attention to EAs and local-search comparisons. From this, we find a great deal of evidence that hillclimbers and simulated annealing outperform standard EAs on various problems. Hybrid approaches are also very successful, we find.

Chapter 3 deals with multiobjective optimization. It begins with a formal definition of the general multiobjective problem, and Pareto optimization is then defined as a special case of it. Further concepts and notation needed in later chapters are also given. We then review some of the main themes in approximate methods for Pareto optimization, including different selection schemes, elitism, and methods for obtaining diverse solutions in the Pareto front. Both evolutionary and previous single-point local-search methods are considered. The chapter finishes with a critical appraisal of methods for measuring the performance of multiobjective optimizers, and analyzes them according to several dimensions. Following this, a number of new performance assessment metrics are proposed for use in this thesis.

Chapter 4 is the first of four chapters focused entirely on original contributions. It outlines the development of PAES: a new Pareto optimizer based on single-point local search. The chapter begins by reviewing the motivation for developing such an algorithm and follows this with an ‘ideal’ specification. The development of the algorithm is then described in stages, detailing the key design choices made. Each of the elements of the final (1+1)-PAES algorithm is explained. The acceptance function and archiving strategy receive close attention. Proofs relating to convergence of the archive of solutions, under different strategies are provided, and the time complexity of the overall algorithm is also considered. Finally, several variants of PAES are specified, including population-based, multi-start, annealing, and tabu-search variants.

Chapter 5 reports on the experimental testing of PAES and its comparison with other modern MOEAs. Two separate studies are considered. In the first one, (1+1)-PAES and some population-based variants are compared with NPGA and NSGA, on four simple test functions from the literature, and two new ones. In this study, (1+1)-PAES is found to provide excellent performance. The final solution quality it achieves is roughly equivalent to NSGA with elitism, over the set of problems, but (1+1)-PAES is far faster and is also easier to use. In the second study, (1+1)-PAES is compared with SPEA on a set of six test functions used by Zitzler *et al.* [ZDT00]. The functions test the algorithms for their ability to cope with different problem features that might cause difficulty to MOEAs. (1+1)-PAES performs relatively well on four of the problems but is not competitive with SPEA on the deceptive trap function  $\mathcal{T}5$ . On the multimodal problem,  $\mathcal{T}4$ , PAES achieves good performance only after the mutation rate is increased. Overall, (1+1)-PAES is found to perform more than adequately as a baseline technique for Pareto optimization, particularly when its low computational overhead is taken

into account.

Chapter 6 relates the development and testing of a memetic algorithm for Pareto optimization, based on the local-search procedures developed for PAES. To begin with a general framework for a memetic algorithm is designed. An instance of the framework, called M-PAES, is then specified, using the PAES local searcher as a subroutine within an elitist, population-based EA. M-PAES is then tested on two combinatorial optimization problems. In the first one, a set of multiobjective knapsack instances with 2, 3 and 4 objectives from [ZDT00] are used. M-PAES is able to obtain better results than a tuned version of SPEA, using the same operators and constraint handling methods in both algorithms. However, M-PAES is rather sensitive to parameters on this problem, we find. The second problem we consider relates to managing the load-balancing in a distributed database. M-PAES does not perform better than (1+1)-PAES on this problem, using the same operators, although it is slightly better than SPEA. This is consistent with some earlier results on this problem that show that it is not very amenable to optimization via recombination.

In Chapter 7, the methods developed in the thesis are used to obtain a set of benchmark results for a number of instances of the multi-criterion minimum spanning tree (mc-MST) problem. The chapter begins with a review of EA literature related to spanning tree problems, including approaches to the degree-constrained MST ( $d$ -MST) and the mc-MST. Two sets of experiments are then performed. The first set focuses on representation issues. In an elitist, population-based MOEA called AESSEA we compare the performance of a Prüfer number encoding with a direct encoding, specialized operators, and the use of a heuristic initialization procedure. The results on a set of random weight problems, demonstrate significant superiority of the direct encoding. As a baseline, we compare these results with an iterated constructive approach, and also enumerate the search space on some of the smaller instances. We find the constructive approach to be much faster than the EA and able to generate equivalent results that closely approximate the true Pareto front. We conclude from this that on these easy problems an EA may not be necessary. In the second set of experiments we devise a suite of more difficult benchmark problems that have an additional constraint on the maximum degree of the vertices in the spanning tree. The benchmark instances also contain problems with correlated edge weight vectors, which affect the shape of the Pareto front. Some instances are also deceptive to constructive approaches. On this suite of problems, (1+1)-PAES, AESSEA and M-PAES are all tested. The latter is found to provide the most robust performance, with no parameter changes required across the set of instances, and outperforms both AESSEA and PAES on most instances. The results presented on these problems provide benchmarks for future methods.

The thesis is concluded in Chapter 8. A summary is given and the main contributions of the thesis are assessed in some depth. The findings of work published by other researchers using our algorithms are also considered. Next, the limitations of the thesis are identified, suggesting where more experiments are needed and where some methods need further development. Finally, a section on further work describes two key directions of interesting further study related to the research in the thesis. Some of this work has already begun and early results are promising.

# Chapter 2

## Approximate Methods of Search

### 2.1 Overview

In this thesis we are concerned with methods for solving certain types of optimization problems. In general, an optimization problem requires us to maximize or minimize some measurable function of one or more variables:

$$y = f(\mathbf{x}) \quad (2.1)$$

subject to  $\mathbf{x} \in X$  where  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  is a *decision vector* and its components are called *decision variables*. Decision vectors are also often referred to as *solutions* or *candidate solutions*. The *search space* which is the set of solutions one is going to search over may be some subset or superset of  $X$ . The function  $f$  is known as the *objective function*. If the goal of the search is maximization then  $f$  is sometimes called a *fitness function* or *utility function* and the value  $y$  assigned to a solution is then its *fitness* or *utility*. Conversely, if the goal is to minimize  $y$  then  $f$  may be called the *cost function* or in the case of constraint satisfaction, the *penalty function*.

Obviously, optimization problems exist in huge variety, and exhibit many different features. Later in this chapter we consider some of the dimensions in which problems can vary, and review how these problem features might affect the search for solutions. Before this deeper review, we briefly examine three examples of optimization problems to introduce some of the main approaches for performing search.

### 2.1.1 Combination lock

Let us first take the simple problem of trying to “crack” a safe or combination lock. Let us imagine that the lock has  $n$  different barrels and  $m$  different numbers on each barrel, and one must find the single combination of numbers that opens the lock. As is normal with any effective combination lock, one receives no information about whether any of the barrels is in the correct position unless simultaneously all of them are, in which case the lock opens. In all other cases the lock simply stays locked. So, we could write the objective function  $f$  for this problem as:

$$y = f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}^* \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

where  $\mathbf{x}^*$  is the correct combination, and we try to maximize the utility  $y$ .

Clearly, there are  $m^n$  different combinations of the lock. One way to crack the lock is simply to exhaustively try every single combination but if  $n$  is large this will take a long time. Using this method, one might be lucky and find the required combination in a few attempts but, in general, the probability of having found the required combination in  $k$  trials is  $k/m^n$ . One would expect, on average, to take  $m^n/2$  trials to find the combination, if one tries each combination in some ordered manner without repeating any combinations, e.g. by starting at the combination 0000...0 and ‘counting’ from there. This kind of approach is called *exhaustive search* or an *enumeration* of the search space.

The problem could also be tackled using a less organized approach. One could try combinations in a random manner, without checking to see if one has already tried the combination before. This approach will take even longer than exhaustive search, in general. The probability of having found the required combination after  $k$  trials is  $1 - (\frac{m^n-1}{m^n})^k$ , assuming that all of the combinations are tried with equal probability. The advantage of this technique (called *random search*) is merely that one does not have to think about how to organize the trials in an order. For some types of problem it is not trivial to organize the trials to avoid repetition, so that random search may be more efficient than an exhaustive search, in the sense that the former may do away with a lot of computation involved in calculating the next solution to try, leading to a faster search overall. Furthermore, for many problems, it is easier to write a computer program for doing random search than for doing exhaustive search, and so in this sense, they may also be more efficient.

### 2.1.2 Wipeout

The combination lock problem is clearly ‘intractable’ for reasonably large instances. In other words, a lock with say 100 barrels, each of 16 numbers, is well beyond the search capability of *any* possible computer, past, present or future (see page 3 of [Whi94]). Let us consider a problem that is similar in form but that is far easier to solve. In the television game-show ‘Wipeout’ [BBC], a player is given a list of items from which she must identify those that are members of a given class. For example, the player might be given the names of twenty different songs, and then asked which six of them were first sung by Frank Sinatra. The player selects exactly six songs from the list and is given feedback (a score) relating the number of songs she has correctly identified. After this she is allowed to repeatedly change her selection but, importantly, feedback is only available when exactly six songs are selected. In the game-show the player must identify all the songs within 30 seconds.

Even if the player does not have any clue as to the Frank Sinatra songs from amongst the distractors, there is a simple strategy that leads to the correct answer in fewer trials than random guessing. The strategy takes advantage of the information given by the ‘current score’ to improve the selection further. The strategy is to make an initial guess at a selection, and then at each subsequent attempt, to swap exactly one item within the selected group with one not in the selected group, according to the following rules:

1. if the score increases or stays the same after the current swap, then on the next trial do a random swap again;
2. if the score decreases after the current swap, then on the next step, reverse the swap just done.

This procedure is repeated until the winning selection of items is found. Let us examine why this is a good strategy for this game. If there are  $n$  items from which  $m$  must be selected then the number of possible solutions is  ${}^n C_m = \frac{n!}{(n-m)!m!}$ , of which only one is correct. Therefore, by enumeration or exhaustive search, the probability of having found the answer after  $k$  trials would be  $k/{}^n C_m$ . By random search, this probability is reduced (for  $k > 1$ ) to  $1 - \left(\frac{{}^n C_m - 1}{{}^n C_m}\right)^k$ . But, using the above swapping strategy, after any pair of trials (after the initial one) the probability of having decreased the number of correct items is 0. So there will be a monotonic increase in the number of correct items, if an even number of trials only are considered. At each trial, the probability of increasing the number of correct items is given by

$$\frac{(m-i)(m-i)}{m(n-m)} \quad (2.3)$$

where  $i$  is the number of items that are already correct. This probability is generally far greater than the probability quoted earlier of guessing all of the items randomly, i.e.  $1/\frac{n!}{(n-m)!m!}$ . So, at every trial there is a relatively high probability of increasing the number of correct items and no chance of irrevocably decreasing them. Since only a small number of increases are needed to reach the goal state of having selected all the correct items, then this strategy will quickly solve the problem.

The strategy just described is an example of local search. In local search, small changes to a solution are tried at each step, and changes that lead to poorer solutions are generally rejected, or accepted (allowed to stand) only with a very low probability, whereas solutions that represent an improvement are maintained and built upon. The particular local-search strategy outlined above is often called *hillclimbing* [MHF94].

Hillclimbing works in the case of Wipeout for several reasons. First, the player receives feedback from the game every time exactly  $m$  items are selected. Second, the feedback leads the player in the right direction, at all times, towards the correct solution. That is, a solution with  $m - 1$  items correct (which scores  $m - 1$ ) is actually very close to the correct answer, in terms of the probability of obtaining the right answer by swapping one item. The feedback reflects the proximity of the player to finding the correct answer. This is in contrast to the combination lock problem where the feedback given was only whether the lock opened or not.

Now, in the game of Wipeout an even better strategy than the one described is available to a player with good memory. The strategy is the same as the one described before except that every time the number of correct items increases, the player should remember the item just added that led to this increase, and in future guesses refrain from ever removing this item. This strategy will lead to the correct solution orders of magnitude faster because as the number of correct items increases, the player actually has fewer possibilities of allowed swaps. This local-search strategy is related to one called *tabu search* [GL97] that will be described in greater detail later. It is also somewhat related to *dynamic programming* [Bel55] methods. The reason this strategy works in the Wipeout game is that correct items are always correct independently of which other items are selected. In other words, correct items are correct in all contexts. This makes this problem *separable*. (We may also say that the problem exhibits no *epistasis*.) Unfortunately, most problems in real-world optimization are not separable. However, most problems are separable to a degree, or there is limited epistasis.

### 2.1.3 Super-wipeout

Let us now consider an imaginary game-show called Super-Wipeout which makes the classification task a little more difficult for the player. In Super-Wipeout  $n$  items are given, and from these exactly  $m$  must be selected, as in Wipeout. However, the players are told that each of the  $n$  items belong to one or more of  $r$  different classes. For example, each item is a song once sung by Frank Sinatra, by Bing Crosby, or by Harry Connick Jnr, or it may have been sung by more than one of them. The aim of the game is to identify all the items belonging to the largest class. In other words if there were 20 songs altogether and 10 were sung by Sinatra, 8 by Crosby and 7 by Connick Jnr, and the player must always select 12 items, then any set of songs containing all 10 of Sinatra's wins the prize. What information is the player initially given? The player is not told what the classes are, or how many items are in the largest class, i.e. she does not know the target value of the number of correct items she must obtain, or from what class these items will be drawn. What feedback is she given as she makes guesses? The number of selected items  $i_r$  that are members of each of the  $r$  classes is calculated. The player is then only told the largest of these  $i_r$  values. For example, if she has selected 12 songs, and 5 were sung by Crosby, 5 by Sinatra, and 2 by Connick Jr then she would receive the feedback score, 5.

Now, this problem can also be tackled by using a local-search strategy but there are now certain difficulties which make the problem harder than Wipeout. First, the problem is not completely separable. An item whose addition increases the score within a particular set of items is not necessarily part of the final solution. So, the problem cannot be solved — as for Wipeout — by holding more and more of the items constant, while searching through a smaller and smaller set. The problem is said to be epistatic, as the contribution of each item is not independent of context.

Second, there are now “local optima” in the problem. This means that it is possible that one can have a selection of items from which it is not possible to increase one's score by any number of single non-detrimental swaps, and yet is not the winning solution. So, if the local search hillclimbing strategy described previously is tried, one can reach a solution from which it is not possible to make any swaps that will improve your score. For example, if all eight Bing Crosby songs were selected then the score given would be, 8. In this situation, in order to get to a higher score, 9 Frank Sinatra songs would have to be selected. This would not be possible by any combination of single swaps without reducing the score at some stage<sup>1</sup>.

Despite the problem of local optima and the problem of epistasis, it is still preferable to use

---

<sup>1</sup>Assuming that Sinatra's ten songs do not include all eight of Crosby's.

a local-search strategy than random guessing of the solution. But in order to find the answer to this problem, one must use a more complicated local-search strategy than hillclimbing to have a good chance of solving it quickly. Possible strategies include:

1. to restart the local search if it is suspected that a local optimum has been reached;
2. to allow swaps to solutions of a slightly worse score;
3. to try each of a number of different swaps from a particular solution  $A$ , remembering the score of each of these trial solutions. Then, to set  $A$  to the solution from among the set of trials with the best score, irrespective of whether it is worse than the previous score of  $A$ ;
4. to gradually increasing the number of swaps that can be tried at the same time to allow jumping across the lower scores.

Each of these gives rise to a well-known local-search technique. 1. is *multi-start hillclimbing* [YI96], 2. is called thresholding, more sophisticated form of which are exemplified by *simulated annealing* [KmV83], 3. is another form of tabu search, and 4. is called *variable neighbourhood search* [RS01]. All of these strategies prevent being stuck forever in a local optimum.

There is another quite different approach to searching for the solution in Super-Wipeout that is not really available to a player, unless they possessed a super memory. Nonetheless, we shall consider this strategy because it introduces some of the concepts described later in this chapter. In this strategy the player tries to keep in mind several solutions at once. Let's imagine that she can do this perfectly for  $P$  solutions. She starts off by making  $P$  completely random and unrelated trial solutions, and remembers the score of each of them. Once again she progresses by making small changes to these initial solutions, but she always works in ‘rounds’ of  $P$  trials, instead of one trial at a time. In her next round of  $P$  trials she tries to improve the original set of solutions, by making single swaps from them, but she allocates *slightly* more of her trials to the solutions that already have a higher score. She does this using a die and some calculations that result in her making random but slightly biased choices as to which of the previous solutions to work from. This means that some of the  $P$  original solutions are not used at all, whereas some might be used two or more times, depending on their relative scores. Throughout the game, she only ever keeps in mind the last round of  $P$  solutions that she tried and their scores, and continues to allocate trials more to solutions that score higher (using her die). Gradually the solutions will score higher, and eventually she might reach the winning solution.

The approach described above is a population-based strategy similar to *evolutionary programming* [FB69] and the set of  $P$  solutions that she keeps in mind is called the *population*. It would work effectively in Super-Wipeout for two reasons. First, it circumvents the problems with epistasis, to some extent. This is because although some of the  $P$  solutions might have high numbers of the Bing Crosby and Harry Connick Jnr. songs in them, others will probably have high numbers of the Frank Sinatra songs. Thus, although some of the evolving solutions will be distracted or deceived into collecting the wrong class of song, some other solutions will not be deceived. As long as only slightly more trials are allocated to high-scoring solutions, then the deceptively good solutions will not take over the population of evolving solutions. Later, the Sinatra solutions can take over. Second, the problem of becoming stuck in local optima — which is related to the epistasis problem — is alleviated. The strategy cannot become completely stuck because there is always a chance that solutions that become worse are maintained in the population long enough for them to get better again. So, even if all the solutions in the population contain 8 Bing Crosby songs, it is possible that one of them by a series of swaps that initially reduce its score, eventually becomes 9 Sinatras, and scores 9, overcoming the local optimum and allowing the optimum of 10 Sinatras to be subsequently reached.

Clearly, the population-based approach would be inefficient for the *original* Wipeout game because the tabu search/dynamic programming procedure outlined previously would work much more quickly. But for Super-Wipeout, the evolutionary programming approach would be more effective.

Let us consider the population-based approach further. One might have noticed that good solutions to Super-Wipeout are those that contain large numbers of items from the same class. Poor solutions are those that contain a more even mixture of items from all the classes. Consider two solutions,  $a$  and  $b$  both with a moderate score, each containing some items from the same class,  $FS$ . Some of the items from  $FS$  are in both  $a$  and  $b$ , and some are in just one of them. If parts of  $a$  and  $b$  could be mixed up together in a single solution then the number of items in  $FS$  could become large, resulting in a high score for this solution. In a sense, the parts of  $a$  and  $b$  that are items in the class  $FS$  are ‘building blocks’ that can be put together to make a good solution. Unfortunately, the player does not know which parts of  $a$  and  $b$  are from the same class (for if she did she would surely make a solution with all of these items in it). Instead, she could try combining them ‘blindly’. To do this she could first place all the items that occur in both  $a$  and  $b$  in the new one,  $c$ , and then finish making  $c$  by adding in random items from  $a$  or  $b$  until  $c$  has exactly  $m$  items in it. This strategy ensures that no item in  $c$  occurs twice, and ensures that common features of the two ‘parent’ solutions are preserved in the ‘offspring’ solution.

But how can she hope to combine the good parts of  $a$  and  $b$  together effectively using this random ‘recombination’ of items? The answer is that she cannot hope to combine them together effectively by luck alone, but by using her strategy of repeatedly working from the solutions in the population that have the better scores and recombining these, she biases the mixing up of the solutions in her favour. Sets of items that occur together in a solution — the building blocks — that most frequently lead to good solutions tend to stay in the population (because they are in good solutions that are generally kept), whereas sets of items that appear together more often in poor solutions become less frequent in the population. So, as her search progresses, the player is more likely to combine together good building blocks that work together to form good solutions.

This last search strategy is an example of another population-based approach called a *genetic algorithm* [Hol75]. It differs from the classic form of evolutionary programming in its use of *recombination* of solutions, but is otherwise similar.

## 2.2 Problems and complexity

In the previous section we saw three different kinds of problem, and a number of different search strategies for tackling them. The first problem was so hard, the only strategy for finding the optimal solution was to search every feasible solution, or to guess solutions randomly and hope to find the answer. The second problem could be solved very quickly by taking advantage of the fact that each element in the optimal solution could be found separately in a small number of trials, and these elements could be simply combined to give the optimal solution. The last problem lay some way in between the first two in its difficulty. This problem could not be separated like the second problem, into easy sub-problems, but several strategies could be used that, we argued, would get us to the solution faster than searching randomly or exhaustively searching every solution.

But, given a new problem, how can we tell whether it is difficult or easy? And what exactly is meant by difficult or easy? These questions are dealt with in a field of mathematics called algorithmic complexity. Algorithmic complexity classifies problems as easy or hard based on the computational complexity of the simplest algorithm that can guarantee solving them. In turn, the computational complexity of an algorithm is merely the number of steps it needs to solve the problem, expressed as a function of the size of an *instance* of the problem.

Problems can be roughly classified into those that are tractable and those that are *intractable*. Intractable problems include those that are formally undecidable [Min67], such as solving

Diophantine equations, for which there exists a proof that no constructive procedure (or algorithm) exists for solving all instances. But intractable problems also include problems which are formally decidable but which nonetheless cannot be solved because the amount of computation time needed to solve them — or their computational *complexity* — is prohibitively large even for “reasonably”-sized instances. More formally, this can be expressed by stating that an intractable problem is one for which there is no known algorithm that operates in a number of steps that is a polynomial function of the input length of the problem (problem size), and which solves *all* instances of the problem. Conversely, a tractable problem, or a *problem in the class P*, is any problem for which *all* instances can be solved using an algorithm that takes a number of steps that is some polynomial of the problem size. For short, we can say that the tractable problems can be solved in *polynomial time*. The intractable problems cannot be solved in polynomial time.

The decidable but intractable problems can be further classified into those for which it is at least possible to *check* a solution in a number of steps that is some polynomial function of the problem size. These problems, together with the tractable problems, form a class that can be solved using a theoretical (imaginary) machine called a nondeterministic Turing machine (NDTM), in a polynomial number of steps. A nondeterministic Turing machine can pursue an unbounded number of ‘guesses’ at the solution to a problem, but its time complexity (or number of steps to reach a solution) is calculated as the minimum time required to check that a guess is a solution to the problem. The class of problems that can be solved by a NDTM in polynomial time is called the *NP* class. Obviously, the class *NP* contains the class *P*, i.e.,  $P \subseteq NP$ . It remains a very important question of mathematics whether, in fact,  $P = NP$ , or whether its contrary  $P \neq NP$  can be proved.

Although to date no-one has managed to prove that  $P \neq NP$ , there is certainly a large body of problems which are considered intractable (i.e. no known polynomial time algorithm exists for solving them), and that are members of the class *NP*. In fact, many of these problems are *very* usual, everyday problems that we would like to solve, including many problems related to partitioning sets of numbers, network design, storage and retrieval of data, games and puzzles, logic, scheduling, program optimization, and many others [GJ79]. The most difficult members of the class *NP* are called *NP*-complete problems. A defining property of the *NP*-complete class is that all problems in *NP* can be ‘transformed’ into any *NP*-complete problem, using a polynomial-time algorithm. Here, “transformed” implies that solving the *NP*-complete problem would result in the solution of the original problem. This property means that if a polynomial-time algorithm is found for *any* problem of the *NP*-complete class, all problems in *NP* can be solved in polynomial time. However, until such a monumental event occurs, proving that a problem is *NP*-complete shows that there

is no known polynomial-time method for solving it, and that probably time should not be wasted looking for such an algorithm.

Note, however, that *optimization problems* are not generally in the class *NP*, and therefore cannot be proved *NP*-complete. This is because it is generally not possible to check whether one has an optimal solution to an optimization problem (equivalent to a solution to a general problem) in a number of steps that is a polynomial function of problem input size. In many cases, the only way to check whether one has an optimal solution is to compare it with all the other feasible solutions in  $X$ . If the size of this set is exponential in the input size of the problem then clearly this cannot be done in polynomial time.

Fortunately, any optimization problem can be restated as a closely related problem, called a *decision problem*. A decision problem has only two possible solutions, either the answer “yes” or the answer “no”. If the yes answer is verifiable in polynomial time by a deterministic Turing machine then the problem is in the class *NP*. Transforming an optimization problem into a decision problem can (usually) be done by choosing a bound  $B$  and simply asking if there exists a solution that has a cost of *at most*  $B$ <sup>2</sup>. Now, if the decision problem equivalent of an optimization problem can be shown to be *NP*-complete then the original optimization problem *must be* at least as hard. This is the case since if it were not true, then the easier optimization problem could be solved first, and then given any bound  $B$  one could check in polynomial time whether the answer was “no” or “yes”, by simply comparing  $B$  to the known optimum, and solving the decision problem, thereby contradicting our original assumption that the optimization problem was easier. Thus, in general, optimization problems, although not in *NP* themselves, can also be “provably hard” — at least as hard as the *NP*-complete problems. These optimization problems (together with other problems at least as hard as the *NP*-complete class) form a class of problems called the *NP*-hard problems.

It should be clear from the discussion above that *NP*-hard optimization problems cannot be solved in polynomial time using any known algorithm, where ‘solved’ is used in the strict sense of ‘solved to optimality’. Thus, many large instances of these problems will take a prohibitively large amount of time to solve, and as the size increases will become truly intractable. For these problems, where large instances do arise in the real-world, we must approach the problem accepting that solving it is not possible for all instances, and instead endeavour to find feasible solutions that are only good approximations to the optimal solution, but in a number of steps that is bounded by a polynomial of problem size. An algorithm that is used to find such approximate solutions is called an *approximate method*, and often relies on some heuristic, which helps to find “good” solutions from the set of feasible solutions.

---

<sup>2</sup>Assuming a minimization problem.

## 2.3 Search algorithms

### 2.3.1 Heuristics

A heuristic is a rule of thumb used to help solve problems where no exact procedure for doing so exists [Pea84]. Best first search is an example of a heuristic used in searching decision trees and other tree-like data structures [Pea84].

A heuristic approach usually involves employing some piece of knowledge about the structure of the *specific* problem being undertaken, to devise a strategy for solving it. As an example, let us consider a knapsack problem [MT90]. In this type of problem, one must select, from a number of items, which ones to place into a container (a knapsack). Each item  $i$  has a size or weight  $w_i$  and a value or profit  $p_i$ , associated with it. The knapsack has a capacity that limits the total weight of items that can be placed in it. The aim of the problem is to select the set of items that maximizes the profit without exceeding the knapsack's capacity. A good heuristic strategy for this problem is to sort the items to be placed in the knapsack according to their profit to weight ratio (PWR), placing them in the knapsack in this order, beginning with the item with the highest PWR, until the knapsack cannot accept the next item. To complete the strategy, continue adding items in order from the sorted list, missing out any items that cannot fit into the knapsack, until the remaining capacity of the knapsack is less than the weight of the lightest item. It is clear that this heuristic is very specific to the knapsack problem, and although it cannot guarantee optimal solutions it does work very well. It is often true that when one has knowledge of a problem, then a very good heuristic can be devised. However, sometimes one doesn't have any strong insight into how a problem might be solved, or strategies could be devised but they may be overly-complicated to implement. In these cases, it is sometimes best to use a more general heuristic, often called a *metaheuristic*. Metaheuristics are sometimes also called *black-box optimization algorithms* [WM97] or simply, general-purpose optimization algorithms.

So, a metaheuristic approach is a strategy that can be generally applied to solving problems. In optimization problems, where we try to minimize (or maximize) an objective function of a decision vector over some decision space, it is often the case that small changes in the decision vector will lead to small changes in the objective function value. In other words, there is often 'smoothness' in the search landscape. This smoothness can be exploited because a corollary of it is that good decision vectors will tend to be near other good decision vectors, and bad ones near other bad ones. Therefore, a good strategy — or metaheuristic — might be to 'walk' through the search landscape by taking small steps (small changes in the decision vector) always in the 'direction' that reduces the cost. Although this strategy can fail for different

reasons, it will often be a better approach than simply trying random decision vectors, one after the other, if the landscape is smooth. This very general metaheuristic strategy is called *neighbourhood search* or local search.

### 2.3.2 Neighbourhood search

The neighbourhood of a solution is defined in [AL97] as follows:

**Definition 2.1** *Let  $(X, f)$  be an instance of a combinatorial optimization problem. A neighbourhood function is a mapping  $\mathcal{N} : X \rightarrow 2^X$ , which defines for each solution  $i \in X$  a set  $\mathcal{N}(i) \subseteq X$  of solutions that are in some sense close to  $i$ . The set  $\mathcal{N}(i)$  is the neighbourhood of solution  $i$ , and each  $j \in \mathcal{N}(i)$  is a neighbour of  $i$ . We shall assume that  $i \in \mathcal{N}(i)$  for all  $i \in X$ . Roughly speaking, a local-search algorithm starts off with an initial solution and then continually tries to find better solutions by searching neighbourhoods.*

### 2.3.3 Random mutation hillclimbing

One of the simplest local-search algorithm is *random mutation hillclimbing* (RMHC), as described in [MHF94]. In RMHC, an initial solution is first generated and evaluated, and this becomes the *current* solution. Then, at each iteration, a copy of the current solution is made, and a random mutation is applied to the copy, producing a new *candidate* solution. (The random mutation is a small change akin to the neighbourhood function defined above.) The candidate solution is then evaluated. If it is not worse than the current solution then it becomes the current solution. If it is worse then it is discarded. The algorithm may be stopped when a specified number of evaluations have been carried out, or when there has been no improvement in the evaluation of the current solution over a specified number of iterations.

Random mutation hillclimbing exemplifies the advantages of a general metaheuristic strategy. It is applicable to any (single-objective) optimization problem whatever; the problem need not be a differentiable function of real parameters. It does not require any knowledge of the problem structure to work either<sup>3</sup>; only the objective function is needed, in order to evaluate each solution that is generated.

If the RMHC algorithm is allowed to run indefinitely then the value of the current solution will converge to a solution that is *locally optimal*, that is, there is no solution in the neighbourhood

---

<sup>3</sup>A neighbourhood function must be defined but this can be designed in some straightforward manner.

of the current solution with a better evaluation. The locally optimal solution that is reached depends upon the neighbourhood function that is being used, the random initial solution, and the random mutations applied at each step. The solution reached may not be *globally optimal*. That is, there may be solutions in the search space, but not in the neighbourhood of the current solution, that have a better evaluation than the current solution. It is not possible to reach these solutions because in order to do so, the RMHC algorithm would have to allow the current solution to be replaced by a solution with a poorer evaluation, so that a different neighbourhood could be explored. We saw this situation in the example Super-Wipeout game considered earlier. This exemplifies the disadvantage of a basic local-search strategy like hillclimbing: it generally converges to a local optimum rather than a global optimum.

One way to partially alleviate the problem of convergence to a local optimum is to use *multi-start* hillclimbing [YI96]. Whereas the stopping criterion in RMHC may be defined in terms of the number of iterations in which no improvement in the evaluation of the current solution is found, in multi-start RMHC, the same criterion is used to define a re-start of the algorithm from a new, random initial solution. If the algorithm is allowed to re-start indefinitely according to this criterion, then it will find a global optimum with probability 1, on all optimization problems [AL97]. This is clear since it will eventually search all neighbourhoods in the search space. However, the length of time needed to do this will, in general, exceed that needed for a deterministic enumeration of the whole search space.

### 2.3.4 Simulated annealing

*Simulated annealing* (SA) is one of the most highly regarded, well-understood, and widely applied local-search algorithms. It is quite similar to RMHC described above but, essentially, it allows the current solution to be replaced by a candidate solution having a worse evaluation, under the control of a randomized scheme. This crucial difference with RMHC means that simulated annealing is able to search for a global optimum, and under certain conditions it converges to a globally optimal solution with probability 1.

In simulated annealing, the probability function for *accepting* the candidate solution  $j$  from the current solution  $i$  (assuming a minimization problem) is:

$$P_{c_k}\{\text{accept } j\} = \begin{cases} 1 & \text{if } f(j) \leq f(i), \\ \exp\left(\frac{f(i)-f(j)}{c_k}\right) & \text{if } f(j) > f(i) \end{cases} \quad (2.4)$$

where  $c_k \in \mathbb{R}^+$  is a control parameter, which is some function of the iteration  $k$  of the simu-

lated annealing algorithm. In SA, the value of  $c_k$  is set initially high, and is gradually lowered, eventually to zero, so that initially transitions to highly inferior solutions are frequently accepted, but later these transitions become extremely unlikely. The regime for controlling  $c_k$  is called the *cooling schedule*, and it specifies:

- an initial value of the control parameter  $c_0$ ,
- a decrement function for lowering the value of the control parameter,
- a final value of the control parameter specified by a stop criterion,
- a finite number of transitions at each value of the control parameter.

The simulated annealing algorithm was first proposed as a search algorithm in 1983 by Kirkpatrick *et al.* [KmV83], based on an algorithm put forward by Metropolis *et al.* [MRR<sup>+</sup>53] 30 years earlier, for simulating the cooling of materials in a heat bath - a process known as annealing. Aarts, Korst and van Laarhoven [AKvL97] have given proofs that the SA algorithm converges to the global optimum with probability 1 provided that the sequence of trials (or Markov chains) approximate a stationary distribution. However, this requires that an exponential number of trials are performed, even to approximate an optimal solution arbitrarily closely, and for some problems (e.g. the TSP) it requires more computation than a complete enumeration of the space. In practice, however, simulated annealing has been practically applied in a large range of applications, including routing, graph partitioning, the travelling salesman problem, VLSI design, and job-shop scheduling, according to [AKvL97].

### 2.3.5 Tabu search

Tabu search (TS) [GL97] is another example of a neighbourhood search algorithm that, like simulated annealing, is capable of avoiding being trapped in local optima. However, the operation of TS is quite different than SA's. At each iteration in TS a *subset* of the neighbours of the current solution are considered, and the *best* of these is chosen. This contrasts to SA which chooses a neighbouring solution at random and then accepts or rejects it on the basis of a probabilistic function. The subset of neighbouring solutions considered at each step is made up of all the solutions in the neighbourhood minus some set of solutions which are considered *tabu*. The tabu solutions (tabu list) are usually solutions or moves that would bring the search back to a solution that has already (recently or frequently) been visited. Thus the tabu list inhibits the search from cycling. The tabu list is a form of short term memory that guides the search away from areas that have already been seen, but it can

also be overridden if a solution that is tabu has some property that makes it particularly appealing (e.g. it has the best evaluation yet encountered in the search). TS also allows for the evaluation function, used to choose the best solution in the reduced neighbourhood under consideration, to be changed so that the search can be guided towards or away from certain areas. Specifically, the search can be guided *towards* solutions that are in some way similar to the current solution (*intensification*), or *away* from solutions similar to the current solution (*diversification*).

TS has been used in a large variety of applications, many of which have been summarized in [GL97]. Hertz *et al.* [HTdW97] also describe the application of TS to problems including the quadratic assignment problem, graph colouring, vehicle routing and course scheduling. TS thus represents a flexible metaheuristic technique that can easily incorporate domain knowledge and heuristics to provide efficient search.

### 2.3.6 Genetic algorithms

Genetic algorithms (GAs), like the other metaheuristics described above, use an iterative approach based on generating and evaluating solutions, one after the other. However, they differ in that rather than searching, at each step, the neighbourhood of a *single* current solution, “GAs use a collection (or *population*) of solutions, from which, using selective breeding and recombination strategies, better and better solutions can be produced.” [Ree96].

In a standard or simple GA, a population of *chromosomes*, representing the decision vectors, through some encoding of them, are initially generated randomly. Each chromosome is then decoded and evaluated according to the objective function. Following this a phase of selection occurs, in which a number of parent chromosomes are probabilistically chosen to generate offspring, with greater chance of selection being given to the chromosomes with higher fitness (a measure of the relative quality of chromosomes in the population). Offspring are generated from the selected chromosomes by applying mutation and recombination operators to them. The mutation operator makes a small random change to a single solution, whereas recombination takes the genetic material from two or more chromosomes, called parents, and forms one or more offspring chromosomes. The offspring replace the parent chromosomes, becoming the next generation. With each new generation, the process is repeated from the decoding and evaluation phase through to replacement. Usually, the algorithm is run for a fixed number of generations, although many other stopping criteria are also possible.

Genetic algorithms were initially introduced and investigated by Holland [Hol75] as a model of adaptation. In [Hol75], several arguments are developed to explain how a genetic algorithm

can conduct complex and robust search by implicitly sampling a large number of partitions of a search space. The general idea is that a chromosome can be viewed, not simply as a sample of the search space in one point, but rather as a sample of the quality of numerous different partitions of the search space. For example, any binary chromosome beginning with a 1 is a sample of all of the chromosomes in the space that begin with 1. This can be stated in another way by using the terminology of schema or hyperplane partitions. A schema is represented by a string constructed with 0, 1 and \* where 0 and 1 represent fixed allele values and \* is the don't care symbol. A schema is said to match a chromosome (and conversely the chromosome is in the schema partition) if the chromosome matches the schema at all of the fixed values. So, using this terminology any binary string beginning with 1 matches the schema 1\*\*\*\*.... Clearly, any chromosome is a sample of *many* different partitions (i.e. matches many different schema), in fact  $2^L$  of them for a binary chromosome of length  $L$  [Rad97]. Therefore, implicitly, a chromosome's evaluation betrays partial information about the average evaluation of all other chromosomes in the same partitions as it. Holland's theory suggests that through the application of biased selection, recombination and reproduction, the frequency of schema represented in the population will increasingly reflect the average quality of these schema in the entire search space, as more and more samples are effectively taken. In other words, schema with higher than average observed payoffs are allocated exponentially more trials over time while schema with lower than average observed payoffs are allocated exponentially fewer trials. And since the combination of highly fit schema within a single solution will generally result in solutions of high quality, the genetic algorithm's search will result in high quality solutions.

However, although the schema theorem developed by Holland is not itself disputed, its predictive power is very limited [Whi94]. In reality, stochastic sampling errors in the selection mechanism, and disruption caused by recombination and mutation operators all conspire to making the genetic algorithm very hard to analyse theoretically. The simpler *building block hypothesis* which is often used to explain the dynamics of genetic algorithms, which states that "genetic algorithms seek near-optimal performance through the juxtaposition of short, low-order, high performance schemata, or building blocks" has been criticized by Grefenstette [Gre93], because it can lead to serious misinterpretations if it is taken as an operational theory of GAs. Even the use of a binary representation, originally thought to maximize the effectiveness of the hyperplane sampling is now not used very often in practical applications [Rad97], and newer texts on the subject, such as [Mic96], encourage the use of more direct representation of parameters. These recommendations follow those of Davis [Dav91a] who argues that using specialized mutation and recombination operators often improves the search for solutions to real problems.

### 2.3.7 Evolution strategies and evolutionary programming

Evolution strategies (ESs) [Rec65, Rud97] “...emphasize the behavioural link between parents and offspring, or between reproductive populations, rather than the genetic link” [Fog94]. That is, the solution parameters or decision vector are operated on directly by the ES operators, rather than through some encoding as in the original GA. The mutation operator used in ESs acts on all the parameters at once, changing the current values to new ones following a Gaussian distribution with zero mean difference and a standard deviation that is controlled by various *strategy parameters* which may be incorporated as part of the solution vector itself, subject to evolutionary change.

ESs may use a single point search, in which case they are termed (1+1)-ESs, or a population-based search similar to a GA. The latter can be either of two deterministic selection strategies, called  $(\mu, \lambda)$  and  $(\mu + \lambda)$  selection. In the former, the  $\mu$  parents of one generation are replaced by the best  $\mu$  from the  $\lambda$  offspring. In the latter,  $\mu$  parents and  $\lambda$  offspring compete together, so the best  $\mu$  from the  $\mu + \lambda$  become the next generation of parents. Although evolution strategies did not originally include operators for recombining solutions in the generation of offspring, a variety of such operators were later added, and reported on by Schwefel in 1987 [Fog94].

Evolutionary programming (EP) has many features in common with ESs. It too uses mutation operators that act directly on the solution parameters, rather than through a genetic encoding of them, and it does not (generally) employ recombination. EP was initially proposed by Fogel [Fog64] where it was applied to the task of evolving finite state machines that learned to predict the next number in a sequence of numbers presented to the machine one at a time. The evolution of two-player game strategies, again using finite state machines, was also investigated in the early development of EP [FB69].

Various selection strategies and mutation types may be employed in EP but generally EP uses a constant population size greater than one. Continuous parameter optimization and the inclusion of self-adaptive parameters similar to those used in ESs came later [Por97].

### 2.3.8 Other metaheuristics

In the sections above a brief overview of some of the main search metaheuristics in use today has been given. The list is by no means comprehensive but most of the key ideas in metaheuristic search are contained in the strategies described above: generating solutions randomly to start with; generating solutions in the neighbourhood of other solutions; proba-

bilistically or deterministically accepting solutions based on their evaluation; recombining two or more solutions to generate new ones; memorizing solutions already visited; and directing the search to particular areas of the search space. Other metaheuristics use essentially the same ideas although they may emphasize one or more of them differently, while yet others combine these ideas with constructive heuristics for generating solutions.

Ant colony optimization [DBT00] is perhaps the most original of the other metaheuristics, in that it uses a *constructive* approach to build its solutions incrementally, by making local choices as to the components of the solution to be included. At the same time, the whole process is iterative, in that the local choices made by an ‘ant’ as it builds a solution, depend upon previous constructions made by other ants in previous iterations. Another combination of constructive and iterative approaches is exemplified by the greedy randomized adaptive search procedures (GRASPs) [FR95]. However, here the constructive part of the algorithm is just as the (re-)initialization phase in a multi-start neighbourhood search strategy.

Scatter search, path relinking, differential evolution, and particle swarm optimization metaheuristics [CDG99] are all population-based evolution algorithms that use specialized operators that are particularly beneficial for exploring real-parameter spaces. These methods all emphasize the use of memory, learning and selection to different extents, reflecting their different origins, but the main contribution of these modern techniques are the novel ways in which new solutions are generated by combining, and learning from, previous ones.

## 2.4 Generality

Most of the metaheuristics reviewed above are supposed to be *general-purpose* optimization algorithms. That is, they are intended for use on a wide range of optimization problems rather than any specific one. In truth, however, they each have their strengths and weaknesses. Even in the simple problems we introduced in the first section of this chapter we saw how different search strategies have varying performance and appropriateness on different problems. However, much research in general-purpose optimization, particularly evolutionary computation, is concerned with designing algorithms that offer high performance on as many different problems of practical importance as possible. Sharpe [Sha00] has dubbed this phenomenon: “the search for the holy-grail algorithm”. However, in order to convince us that her new algorithm is a holy-grail ( $\mathcal{HG}$ ), or is at least generally competent, a researcher must demonstrate its performance on problems that exhibit different problem features, known or believed to exist in real-world problems, and known or believed to cause difficulties to different search strategies. In the next section we consider some of these different problem features,

and discuss how they might affect different search strategies.

## 2.5 Problem features and fitness landscapes

Practical optimization problems that we may genuinely be interested in solving — so-called real-world problems — come in many, many different forms. Whether our aim is to design and demonstrate an algorithm that performs well on a large portion of these problems, or to establish which problems a particular algorithm is well-suited to, and which it is not, it is essential to understand the different features exhibited by real-world problems. We are particularly interested in those features that cause the most difficulty for search, or on which it seems that a particular technique would fail or do especially well. In general, understanding problem features may help us progress in our ability to solve optimization problems in two different respects:

1. By testing our algorithms on *artificial test functions* that contain a problem feature in isolation and allow that feature to be controlled, we may understand which techniques are effective on problems with particular features, and we may be better equipped to design better techniques for tackling problems with these features
2. Given any particular *real-world* problem, measuring the presence of absence of these features may help us to choose the best available technique to use.

Roughly speaking, different problems can be described according to the degree to which they exhibit these features:

- Discrete (combinatorial or Boolean) vs. continuous variables;
- Dimensionality;
- Multimodality;
- Discontinuity i.e. non-differentiable vs. differentiable;
- Epistasis vs. linear separability;
- Deception;
- Unconstrained vs. linear constraints vs. non-linear constraints.

The above list represents some of the more accessible and well-investigated ‘dimensions’ of problem type, but is by no means an exhaustive list. In the following we say a little about each of these problem features.

### 2.5.1 Discrete and continuous variables

Much of heuristic search is concerned with problems whose solutions can be expressed exactly using a finite length string of integer parameters. Such problems are called discrete or combinatorial optimization problems, and include all those detailed in [GJ79]. They also include test problems such as the *NK* landscapes [Kau89] and Royal Road functions [MHF94].

In addition to combinatorial problems, metaheuristic search may also be used to optimize mathematical functions of one or more continuous parameters. These are usually tackled by choosing a finite precision with which to express the parameter(s). The parameter value(s) may then be represented using chromosomes in which the allele value of each gene represents the value of a parameter directly (to some precision), or the parameter value(s) may be encoded, for example in binary, on the chromosome using multiple genes.

It is clear that local-search algorithms can easily be adapted for both continuous and discrete spaces by defining the neighbourhood operator appropriately. GAs were originally thought to work best with binary representation of variables, be they continuous or discrete. But recombination of binary strings that are representations of continuous variables does not preserve or recombine what is likely to be, in the building block sense, the relevant information. Evolution strategies and evolutionary programming typically operate directly on the continuous decision variables, and thus their operators are particularly suited to these problems. Others have now developed similar operators for use in GAs. e.g. [Dav91b, DA95]. Optimization strategies such as scatter search, path relinking, and differential evolution, are particularly suited to continuous variable problems because they emphasize the production of new solutions by a linear combination of others [CDG99].

### 2.5.2 Dimensionality

Dimensionality refers to the number of dimensions of the parameter space,  $X$ . Test function suites used for assessing EAs and other techniques should contain problems of high dimensionality because these are more representative of real world problems according to Bäck and Michalewicz [BM97]. Low dimensional problems can often be solved more efficiently by traditional optimization methods such as mathematical programming.

### 2.5.3 Multimodality

Most real-world optimization problems of interest are multimodal, that is, they contain more than one optimum. Sometimes the optima in a multimodal landscape may be of different levels or of the same level. If they are all of the same level then they are all global optima. Finding one of them is usually sufficient to solve the optimization problem exactly, thus multimodality can potentially make a problem easy, as many points are easier to search for than one.

If the optima are of different levels, then some are not global optima. These ‘local’ optima are generally regarded as a nuisance, particularly for local-search algorithms such as hillclimbers because they can become stuck in them, unable to escape to any point of better evaluation. However, as we have seen in our discussions above, there are many strategies for escaping from local optima, although they generally slow down progress compared to a straightforward hillclimbing technique.

Genetic algorithms (and other population-based techniques) are often touted as being particularly suited to searching multimodal landscapes. However, [LP98] shows that when the problem becomes very strongly multimodal EAs may be little better than random search. Furthermore, research comparing methods on the *NK* landscapes [Kau89] suggests that GAs are certainly not peerless on multimodal problems [MF98].

In some applications, it is required or desirable to find multiple optima, particularly if multiple global optima exist. Genetic algorithms do have the advantage on problems of this type, as multiple optima can be found from a single algorithm run if niching techniques are used [GDH92].

Bäck and Michalewicz [BM97] suggest that test function suites should contain a few unimodal functions in order to test the speed with which the search method approaches the objective (efficiency). On these problems, one would expect hillclimbing algorithms to exhibit particularly efficient search.

### 2.5.4 Discontinuity

Heuristic search can clearly deal with discontinuous objective functions because it does not rely on gradient information to find the direction of search. This makes it suitable for combinatorial problems which are always non-differentiable. When a mathematical function is continuous it is possible that gradient methods such as Newton’s method ([BC81], page 666) may be faster than “blind” methods. However, gradient-based methods do not usually fare

well on high dimensional and/or multimodal search landscapes.

Discontinuity is not usually regarded as a relevant dimension of problem difficulty for general-purpose search methods even though local search does rely on the extent to which points within a neighbourhood have similar evaluation. Instead, the related concept of ‘ruggedness’ is used, which is generally subsumed into the multimodality dimension.

### 2.5.5 Deception

The notion of deception in problems has been an important concept in understanding what makes a problem difficult for a genetic algorithm, albeit it is a notion fraught with controversy.

According to Deb [Deb97], “Deceptive functions are those in which low order building blocks do not combine to form higher-order building blocks: instead they form building blocks for a suboptimal solution.”

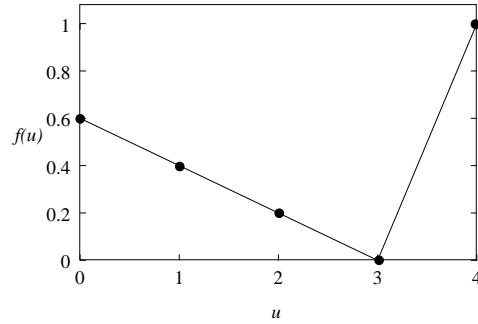


Figure 2.1: A four-bit trap function as a function of unitation  $u$ .

For example, the deceptive trap functions [DG92] are a common deceptive test problem, defined in terms of unitation  $u$  (the number of 1s) on a binary string, as follows:

$$f(u) = \begin{cases} \frac{a}{z}(z - u) & \text{if } u \leq z \\ \frac{b}{l-z}(u - z) & \text{otherwise} \end{cases} \quad (2.5)$$

where  $a$  is the value of the deceptive optimum,  $b$  is the value of the global optimum,  $l$  is the length of the string and  $z$  is a parameter controlling where the slope in the function changes when  $f(u)$  is plotted against  $u$ . Figure 2.1 shows a deceptive trap function with  $a = 0.6$ ,  $b = 1.0$  and  $z = 3$  (from [Deb97]). In this function, the solution with three 1s is at the basin of attraction for the global optimum solution (four 1s). On the other hand, all solutions with fewer than four 1s (15 of them) are attracted towards the deceptive optimum (four 0s).

Thus, the basin of attraction for the deceptive optimum is much larger than for the global optimum.

Das and Whitley [DW91] suggest that only deceptive problems are hard for a GA. However, Grefenstette [Gre93] points out that deception is not necessary for a problem to be hard, citing the needle-in-a-haystack problem as a straightforward counter-example. Watson *et al.* [WHP98] have also provided a test problem, H-IFF, that is very difficult for a standard GA, without possessing any deception. Instead, H-IFF exhibits strong epistasis at different hierarchical levels, making it difficult for the GA to maintain and assemble building blocks.

The deceptive trap functions are sometimes cited as an example of a problem that can be tackled well by a GA (albeit one with niching, or some method of searching specifically for building blocks up to a particular order) but which is very difficult for local-search methods to solve. However, an order  $n$  deceptive trap function is completely separable (see next section) at order  $n$ , so it could also be solved more efficiently using some other technique, provided the position of building blocks is known. Knowing the identity and position of building blocks is, of course, at least half the problem. However, demonstrations of GAs solving deceptive problems often rely on building blocks being ‘tightly linked’ a form of *a priori* knowledge about where they are. In fact, GAs using one or two point crossover can solve deceptive trap functions of low order only if the traps — the building blocks — are tightly linked on the chromosome. If they are bit re-ordered then the GA has much greater difficulty solving the problem.

### 2.5.6 Epistasis and linear separability

Epistasis is a measure of the degree of interaction between parameters in an objective function. If a problem has no epistasis then all of the parameters can be independently optimized, so that the number of points that must be visited is very small compared to the whole search space. If the parameters in a problem can be split into groups in such a way that, taking each group separately, the parameter values within that group which give the best evaluation, with the values of all other parameters held constant, are the same as those in the global optimum, then the problem is linearly separable. On the other hand, if in a problem, the contributions of all parameters depend upon all others then the problem has unbounded epistasis and is not linearly-separable. Such a problem is generally difficult to search using an EA or any other general-purpose technique. For this reason, epistasis and epistasis variance have been used as predictors of problem difficulty. It has also been suggested by some that real-world problems exhibit bounded epistasis and this makes it possible to search them efficiently using EAs and other metaheuristics.

The property of epistasis is closely linked with multimodality and deception considered earlier. The presence of epistasis leads to multimodality: Kauffman's *NK* landscapes are models of tunable rugged landscapes built up from maps of how the parameters interact; higher parameter interaction leads to greater ruggedness. Deception relies on epistasis because it relies on the fact that unless a significant portion of the elements of the optimal solution are discovered at once by a search algorithm, then these elements will not be attractive. Thus these groups of elements are epistatically linked.

### 2.5.7 Constraints

Constraints are virtually ubiquitous in real world optimization problems, both discrete and continuous, so we should expect that good general-purpose search algorithms can deal with constraints. Bäck and Michalewicz [BM97] suggest that test function suites for assessing the ability of EAs to deal with constraints should ideally contain functions with

- different numbers and types of constraint (linear and non-linear),
- the optimum located in different places, particularly on the constraint boundary,
- and, different relatively sized feasible and infeasible regions,

in order to emulate the full range of real-world constrained problems.

It can be argued that population based evolutionary techniques are better suited to constrained optimization problems than local-search methods because the latter can become trapped in suboptimal feasible regions, unable to traverse an infeasible region in order to approach the optimum. However, there are many different approaches to dealing with constraints, including penalty functions, decoders, repair mechanisms, and others, that can allow both population-based and local-search methods to work in constrained search spaces.

## 2.6 No-Free-Lunch and its implications

The no-free-lunch theorem (NFL) [WM97] states that all search algorithms have identical performance when their performance is averaged over all possible search spaces. This result proves the futility of trying to devise a truly general-purpose search algorithm. However, it does not imply that a particular algorithm cannot be better over a restricted set or class of

problems<sup>4</sup>. The class or set of problems that most researchers are interested in is the set of real-world problems (RWPs). The NFL does not specifically rule out the possibility that one algorithm could be best over the set of RWPs. Thus, it is suggested in [Sha00](page 80) that some researchers still believe that it is possible that some algorithms perform better than others, when performance is averaged over all possible real-world problems, and that consequently some algorithm or group of algorithms must be ‘best’, over this set. In Sharpe’s notation, such an algorithm — a holy grail for real world problems — is denoted by  $\mathcal{H}G_{RWP}$ . Of course, the set of real-world problems is not a well-defined class so neither a proof nor a refutation of the possibility of an  $\mathcal{H}G_{RWP}$  is currently available.

However, even if NFL does not presently have anything to say about the RWPs, and even if there were such a group of algorithms that *on average* were best over the set of RWPs, it does not mean that these algorithms would be the best choice on all new real-world problems. Obviously, when we encounter a new problem we *do* usually come to it with *some* knowledge of its structure before any evaluations of the search space are made. Therefore we can hope to apply roughly a good approach to it, better than blindly applying our putative  $\mathcal{H}G_{RWP}$ . In some sense we are outside the NFL because we can make predictions about the search space without making any evaluations of it. In fact, in real-world problems we almost always do not approach a completely black box.

So, at present, and probably for at least some time into the future, the best possible strategy for approaching a new problem is to look at it, make some judgments about the features it will possess and then try out some different algorithms, perhaps being guided by some measurements of the search landscape etc., until we find an approach that works well enough on it. This process cannot all be automated, at present, and never can be if it is the case that we are able to make some useful judgments without actually performing any objective function evaluations. Our job then, is to try to choose the best algorithm for the problem. This matching of algorithm to problem *is* worthwhile doing.

From the discussion about problem features above, and what has been said about the NFL, it should be clear that — for the time being at least — we should reasonably expect that we will have to choose the most appropriate approach, be it hillclimbing, simulated annealing, a GA, TS, or some hybrid algorithm, given a problem and what we know or expect about its features. Any talk of there presently being a single algorithm which is so good that it should always be applied to problems (even of restricted classes) should not be believed. Schwefel makes this point very clearly when he writes [Sch97]:

---

<sup>4</sup>In fact, on the contrary, if one algorithm  $A$  is shown to perform worse than another  $B$  on one problem  $\pi$ , by the NFL one can state that  $A$  is better than  $B$  on average over the set of all problems excluding  $\pi$ .

First, there will always remain a dichotomy between efficiency and general applicability, between reliability and effort of problem-solving, especially optimum-seeking algorithms. Any specific knowledge about the situation at hand may be used to specify an adequate specific solution algorithm, the optimal situation being that one knows the solution in advance. On the other hand, there cannot exist one method that solves all problems effectively as well as efficiently. These goals are contradictory.

### 2.6.1 Test functions and GA performance

In the late eighties and early nineties, much work was carried out on trying to identify problem features that would make a problem easy or difficult for a genetic algorithm to solve as compared to local-search algorithms, particularly hillclimbing. Amongst the features put forward were deception, multimodality, crosstalk, isolation, and noise [HG95] as well as exponentially long paths of monotonically increasing evaluation [HGD92].

In order to investigate the building block hypothesis [Hol75], Mitchell, Forrest and Holland put forward a set of test functions called the Royal Roads with identifiable building blocks or low order schema that were non-overlapping on the chromosome. These low-order schema could be combined to form higher order schema which were rewarded with a larger evaluation. By studying how a genetic algorithm processed these separate and identifiable schema, Mitchell *et al.* hoped to test the “most prevalent “folk theorem” of GAs — that they will outperform hillclimbing and other common search and optimization techniques on a wide spectrum of difficult problems, because crossover allows the powerful combination of partial solutions” [MHF94]. However, although the Royal Roads were designed to have just the sort of building-block structure ideally suited for GAs, Mitchell *et al.* found that a simple random mutation hillclimber was a factor of 10 faster than a GA with crossover at solving the these functions. The explanations given for the poor performance of the GA on this problem underlined the fact that selection pressure, mutation rates, population size, and crossover all need to be very carefully controlled in a standard GA in order for it to perform better than a hillclimber on the simple Royal Road functions.

The results of Mitchell *et al.* followed prior work by Tanese [Tan89] in which test functions based on Walsh polynomials [Wal23, Bet80] were used to compare the performance of genetic algorithms and a multi-start steepest-ascent hillclimber. Tanese found that hillclimbing outperformed a genetic algorithm on all the Tanese functions; functions of length 32 bits and varying Walsh polynomial order of between 4 and 20. In an analysis of the results of Tanese, Forrest and Mitchell admitted that the Tanese functions did not exhibit deception

— previously thought to be the main cause of poor GA performance. In fact, Forrest and Mitchell found that further causes of poor GA performance had to be postulated. Grefenstette [Gre93] also analysed the theory of deception as put forward by Goldberg, Liepins and Vose, and others, and considered that it was not an adequate explanation of the poor performance of GAs, since it was neither necessary nor sufficient for a problem to be difficult for a GA.

Wilson [Wil91] and Louis [LR93] had more success in finding problems for which a simple genetic algorithm outperforms a stochastic hillclimber. In the former, a population-based steepest ascent hillclimber does not perform as well as a genetic algorithm with crossover on a GA-easy problem, that is, one in which the low order schemata associated with the global optimum have higher static average fitness than the competing schemata in their partitions. However, the work of Forrest and Mitchell indicates that steepest ascent hillclimbing can exhibit poorer performance than random mutation hillclimbing because it does not share the ability of the latter to explore plateaus. With this fact taken into account the conclusion of Wilson is weakened.

Louis constructed a problem that was maximally deceptive for a hillclimber, with a structure tailor-made for one point crossover. He showed that on this problem a GA indeed outperforms stochastic hillclimbers and populations of stochastic hillclimbers. However, the structure of the problem put forward by Louis is particularly contrived. Louis does not consider adding noise to this function or re-mapping some bit values. The problem therefore looks unconvincing as a model of any real-world problem.

More convincing demonstrations of good GA performance compared to hillclimbers were given by Deb and Horn [HGD92] on functions with exponentially long paths for a hillclimber to the global optimum. They showed that a GA with crossover is orders of magnitude faster than algorithms (hillclimbers) that only search a small neighbourhood with high probability, and a larger neighbourhood with very small probability.

A comparison of methods was also carried out by Keane [Kea96], on a Royal Road function described in [MFH91] and the fifty dimensional bump problem introduced by Keane. SA is compared with GA, EP, and ES. Keane demonstrates that the GA is clearly best on these problems, and is robust to changing its parameters. He argues that a modern GA with niching and a form of restricted mating, as used in his experiments, would outperform even hand-tuned versions of the EP, ES and SA algorithms on the test functions considered.

In two papers by Watson *et al.* and Watson and Pollack [WHP98, WP00], some of the problems of the Royal Roads as functions for demonstrating the power of GAs to identify and

recombine building blocks, are overcome in a new test problem, H-IFF, which they propose. H-IFF combines the features of building block interdependency (i.e. partial-separability) and competing optima, within a clearly defined hierarchical building block structure like that in the Royal Roads. The resulting functions exhibit fractal multimodality, making them difficult for hillclimbing algorithms to solve. Watson *et al.* demonstrate that a simple GA, although it cannot solve the problem, significantly outperforms random mutation hillclimbing. When the simple GA is augmented by a resource-based niching strategy it can solve the problem to optimality, demonstrating that in order to solve H-IFF, it is necessary to maintain and combine diverse building blocks. When the bits of the H-IFF problem are re-mapped (or shuffled), however, a GA cannot solve the problem to optimality. This demonstrates that the GA requires tight genetic linkage in a problem if crossover is to be a useful tool for combining building blocks. The conclusion of Watson *et al.*'s early work are that GAs can discover and combine building blocks provided diversity is maintained, and provided there is tight genetic linkage. Furthermore, given these conditions, a GA can clearly outperform hillclimbing on H-IFF-type multimodal functions. The requirement of diversity maintenance is in agreement with the findings of Keane discussed above.

Merz and Freisleben [MF98] compared GA, genetic local search (GLS) [IMT97], and multi-start local search (LS) on  $NK$  landscapes, with a range of  $K$  and  $N$  values with  $N$  going up to 1024. They find that for large  $K$  and  $N$  a multi-start LS algorithm outperforms both 1-point and uniform crossover GAs. However, their key findings were that the genetic local-search algorithms outperform both the GAs and LS algorithms at all sizes of  $N$  and  $K$ , and scale particularly well. These findings on these test functions provide strong evidence of the power of hybridizing local search and GA strategies. Much further evidence of this is provided in papers discussed below.

Overall, much of the work in the GA literature on test functions serves to show that simple GAs can exhibit poor behaviour in many function domains, and often require additional features in order for them to outperform simpler techniques like hillclimbing and simulated annealing.

### 2.6.2 Real-world optimization problems

Although the test function studies reported in the previous section are interesting and useful because specific problem features and effects can be isolated and investigated, many researchers believe that real-world problems often exhibit different features that are not readily captured by test functions. They advocate the use of real-world problems for testing and comparing algorithms, arguing that test functions are artificial landscapes that do not truly

reflect the nature of problems that we would actually like to tackle.

A very large number of studies in which metaheuristics have been compared on real-world problems have been published, and we consider just a sample of these here, summarizing some of the key findings of them. We note also that the term real-world problem does not always imply that the instances used in these studies have actually been taken from problems in the real world, or that they are of the same or similar size to problems of actual interest. In some cases they are much simpler. However, the problems tackled do reflect real problems in the sense of their general structure, if not their size, number of constraints, or other details.

Bramlette and Bouchard [BB91] compare GA, SA and hillclimbing on an aircraft design problem. Their study provides evidence that considerable tuning of the GA approaches was necessary for them to reach a level of performance achieved by SA. It is not clear from the study how much effort was put into tuning the SA cooling schedule. Interestingly, the short study finds that the best result for the GA approach is when it is hybridized with the stochastic hillclimber, and iterated with multiple random re-starts.

In [JW94], four real-world problems for which GA approaches had previously been proposed in the literature are tackled using stochastic hillclimbing. The problems are the maximum cut problem, the 11-multiplexer problem (tackled using genetic programming originally), the multiprocessor document allocation problem, and the job-shop scheduling problem. In each case, a stochastic hillclimber (capable of exploring plateaus in the landscape because it accepts all non-worse neighbouring solutions, rather than just improving solutions), using (in general) a similar encoding and mutation operator as proposed for the GA was tested, using runs of the same length in terms of function evaluations as performed by the GA. In all cases the stochastic hillclimber is found to outperform the GA in terms of mean values achieved over 100 runs. For the last problem, the authors use a multiple start strategy where each run of the hillclimber is divided into 5 mini-runs (and the best solution found returned), but the authors claim that this strategy was chosen arbitrarily and may be improved if tuned further. The authors stress that the aim of their study is to encourage GA practitioners to ‘baseline’ their approaches against simpler techniques — not to try to argue that hillclimbing is superior to evolutionary algorithms. Indeed, in the last set of experiments where the encoding and operators used in the hillclimber *were* different from the GA they compared against, they develop a new GA using this improved encoding and operator pairing that does outperform the hillclimber.

The point of view expressed in [JW94], that researchers should baseline their approaches against simpler techniques, is a view that is shared by us, and central to the work in this thesis. However, we focus on work in the multiobjective EA field, where no baseline algorithms

have been put forward, and comparisons with fundamentally different techniques, are rare.

Duvivier *et al.* [DPT96] also compare multi-start hillclimbing to an EA on the job-shop scheduling problem using a direct encoding of job order. Their algorithm study also includes a fairly simple tabu search implementation and a hybrid between the EA and the hillclimber. The findings of this study echo those of [JW94], in that an EA is outperformed by hillclimbing using the same encoding, mutation operators, and number of evaluations. These findings lead the authors to question the value of recombination on this problem, particularly under the encoding used, but they also emphasize that the poor EA results on the scheduling problem differed greatly from previous results where an EA was compared to hillclimbers on a number partitioning problem. The results of hybridizing the EA and hillclimbing algorithms on the scheduling problems were found to be, in general, better than the pure EA and similar to the multi-start hillclimber.

Yagiura and Ibraki [YI96] compare various metaheuristics on single machine scheduling problems. They report that a GA is found to be robust to the type of neighbourhood operator used but that its performance is poor compared to the other algorithms considered. They find that multi-start local search is the best simple algorithm with few parameters to control and good performance. Overall, however, simulated annealing and a genetic local-search algorithm are found to give the best performance over long computational runs. The performance of tabu search, GRASP, and iterated local search, all depended strongly on the neighbourhood defined for the problem, but achieved better results than the GA for the more effective of the two neighbourhoods explored in the study.

Mann and Smith [MS96] compared SA and GA approaches on a number of telecommunications traffic routing problems. A careful set of experiments were carried out to select the parameters of each algorithm before final results were collected. Both algorithms were run until they had converged as judged by the rate of improvement having fallen below some defined level. The findings indicated that the SA and GA algorithms reached very similar final evaluation levels on the range of problems, but SA converged in approximately 10 times fewer evaluations. This finding, combined with the fact that the SA algorithm could be accelerated further by using “delta evaluation” of solutions led Mann and Smith to conclude that SA would be the favoured approach on this application.

In a paper by Lahtinen *et al.* [LMST96], a GA, SA, and two greedy local-search algorithms were compared on a spanning tree problem in which an optimal branching must be determined to minimize the cost of the tree while meeting certain capacity constraints. The experiments carried out allowed for the greedy search algorithms to be run multiple times per single GA run to allow for the fact that they converged in far fewer evaluations. Graphs were then

plotted of the best solution found against total evaluations carried out for each algorithm. The SA and greedy algorithms all outperformed GA on all three problem sizes tried. One of the two greedy approaches was particularly fast and effective, but SA reached the best evaluation in all of the experiments, though with slightly slower initial convergence speed. Furthermore, SA was found to increase in relative performance compared with the GA as problem size increased. Further similar comparisons in [LMS<sup>+</sup>98] show that while a GA compares favourably with the other approaches on a frequency assignment problem, in terms of final evaluation achieved, it requires of the order of 10 times as many evaluations as SA to achieve this result, and 20–40 times as many as a greedy local search approach which only achieves very slightly inferior solutions.

Oates [Oat00] compares the performance of various EAs with SA and HC on 4 different instances of the adaptive distributed database management problem [OC00]. Three different evaluation limits were investigated in each of the approaches taken. The SA algorithm was used with an appropriate geometric cooling schedule for each of the three evaluation limits. The EAs investigated used 4 different mutation rates and 3 different population sizes giving 12 in all, and were otherwise similar. Oates found that no single algorithm outperformed the others over the set of problems and evaluation limits. With the problems possessing a larger search space, the performance advantage of the EAs was seen to diminish with respect to hillclimbing and simulated annealing. Patterns in the EA results suggest that a low mutation rate and large population size give good performance on the larger problem spaces but these rules of thumb are not particularly accurate. Oates concludes that SA is a good default technique while the EAs clearly give better performance when “...simple consideration is given to parameter choice...”, but it is clear from the study that much more effort was expended on tuning the EAs than on trying different cooling schedules for SA. Thus, once again we see that EAs do not offer a clear advantage over local-search algorithms, particularly when the effort to tune them is taken into account.

## 2.7 Hybrid evolutionary approaches

In the real-world applications studies considered above, genetic algorithms were often compared with other heuristic optimization techniques. On some of the problems, GAs clearly did not perform as well as other approaches. This result simply underlines the fact that GAs are by no means a ‘special’ approach, both efficient and effective on all problems. Rather, the GA’s strength lies in the large domain of problems to which it can be applied with minimal problem-specific knowledge [Gol89](Chapter 8).

Nonetheless, the genetic algorithm framework also permits the use of natural data structures, problem-specific operators, repair mechanisms, penalty functions, decoders and the like, to make a much more problem-specific algorithm. Michalewicz calls the resulting algorithm, an evolution program [Mic96]. Such tailoring of GA encodings and operators to the problem at hand is the minimum requisite in any serious attempt to obtain a useful optimization algorithm based on a GA, for most real-world problems.

A truer hybridization is described in [Dav91a] where Davis proposes a method for making a hybrid GA for any problem where a good optimization algorithm is already known (the *current algorithm*). This involves first using the current algorithm's encoding in the hybrid GA. In addition, elements of the current algorithm can be used to generate initial solution vectors, and/or to make "successive transformations" of, or search, solution vectors, and/or to decode solution vectors appropriately. In Davis' scheme, the GA's own genetic operators should then also be adapted to the specialist encoding, whilst maintaining their usual roles in genetic search. Davis maintains that this scheme will almost never fail to improve on the current algorithm's performance.

The model of Davis, particularly where the current algorithm is used for initialization and to actually make "successive transformations" of the solution vector is more akin to the notion of a memetic algorithm as described in [Mos99] and [MF99], than is simply the exploitation of problem-specific knowledge as used in Michalewicz's evolution programs [Mic96]. Both formulations fall under what we consider to be a hybrid evolutionary approach, for the purposes of this section. Memetic algorithms, for our purposes, mean evolutionary approaches that include some iterative local search element to improve the solutions found by recombination and/or mutation.

Memetic algorithms, albeit under different names, have been remarkably successful on a wide range of *NP*-hard problems including TSP, graph colouring, set covering and many others. For an extensive list with references see [Mos99]. The theory behind the approach is not well-founded, however [Mos99]. Nonetheless, some interesting and potentially general results have been found. Sharpe [Sha00] empirically demonstrates that searching a landscape with a hybrid approach which mixes hillclimbing, recombination and mutation performs moderately well on each of three different landscapes which are respectively designed to be easy for hillclimbing, easy for recombination, and easy for mutation. The hybrid does not perform as *efficiently* as the appropriate pure strategy but its effectiveness is high, suggesting that a hybrid merely possessing the appropriate operator(s) will be effective on a given problem. Sharpe's results also demonstrate the converse, that is, that a hybrid algorithm is only as good as its operators are appropriate. Hybrids without mutation, on a mutation-easy problem do

not fare well, for example.

Finding what the appropriate operators are for a given problem landscape is the subject of [MF99]. Merz and Freisleben show that by analyzing properties of fitness landscapes, appropriate representations and operators can be designed. However, the authors also note that in many applications, different instances of the same problem have different landscape properties which may makes the use of landscape metrics for predicting what operators to use an unreliable approach. Despite this caveat, Merz and Freisleben do report that MAs using the appropriate operators are the most effective algorithms on a number of *NP*-hard problems.

Using hybrid algorithms which incorporate local search may improve efficiency too. For example, Smith *et al.* [SDRW00] note that in some applications, the evaluation of solutions can be significantly accelerated if ‘delta evaluation’ is used. In local search approaches, where each new solution differs only slightly from its predecessor, the fitness of the new solution can be calculated by making a slight adjustment to the fitness of the predecessor solution to reflect the small change made to the solution. This saves significant computation time compared to doing a complete evaluation. This means that local search approaches have an advantage over GA approaches on these applications, because GAs cannot typically make use of delta evaluation; the crossover operator is too disruptive for it to be worthwhile. In the fixed channel assignment problem, simulated annealing approaches are much more efficient than GAs for this reason. However, in [SDRW00], a hybrid GA which overcomes this problem, to some extent, is proposed. It uses a crossover operator which tries to improve the better of a pair of parents by swapping in individual genes from the inferior parent. This process is done using a greedy search through the genes, using delta evaluation at each step. Smith *et al.* obtained a large efficiency improvement using this hybrid approach compared to a standard GA. On medium-sized problems the hybrid approach was found to be competitive with SA. On large problems, however, SA remained more effective and efficient.

## 2.8 Summary

In this chapter we have seen that, despite a great deal of research in general-purpose optimization, there is no such thing as a truly general-purpose algorithm that performs better than others on all (real-world) problems, or that can even guarantee better *average* performance than *any* other method over all problems. Moreover, it is not easy to predict which algorithm will perform most effectively and/or efficiently on any given problem, or given instance of a problem. Much theory and empirical investigation has been directed to classifying

and quantifying various dimensions of problem features, and has tried to match these with predictions of algorithm performance. However, to a great extent, it is still not possible to make accurate predictions as to which algorithm(s) will perform best on a given problem (instance). We have seen that on many problems, local-search based methods perform better than population-based evolutionary methods, and that on a great number of problems, hybrid methods also perform well. With this context in mind, we will later propose both local-search and hybrid approaches for Pareto optimization, as alternatives to population-based MOEAs, which are the dominant approach, today. However, before these contributions are presented, a review of the current state-of-the-art in Pareto optimization is given in the next chapter.

# Chapter 3

## Pareto Optimization

This chapter begins with a brief overview of multiobjective optimization and explains its relationship to the more specific domain of Pareto optimization. Next, the general Pareto optimization problem is formally defined, and some key concepts and notation are introduced. A number of different approaches to general-purpose Pareto optimization are then reviewed, and the themes of elitism and diversity control are considered. We also review the literature on methods for testing, assessing and comparing the performance of multiobjective optimizers, and critically analyze a number of current approaches. Finally, we propose some new performance metrics that will be useful in assessing the algorithms presented in this thesis.

### 3.1 Introduction and overview

When one thinks of optimization in general, or about well-known optimization problems, one usually thinks about problems of minimizing or maximizing a *single* quantity or objective. Indeed, with many optimization problems, one begins with the implicit assumption that all candidate solutions can be ranked unambiguously<sup>1</sup> according to their cost or utility. The goal of the optimization process is then well-defined: One must find the highest ranked solution(s) possible.

But in real-world applications, problems with a single, well-defined objective to optimize tend to be the exception rather than the rule. In finance, operational research, medicine, engineering, design, planning, scheduling, timetabling and many other domains, the norm is for problems with multiple and conflicting criteria. Often, problems with a single objective

---

<sup>1</sup>Albeit the evaluation of solutions may be subject to noise or uncertainty.

may express what is most important or fundamental about a task in these domains, and they are a mathematical nicety, greatly simplifying the problem, but they are not a faithful model of the real world.

Unfortunately, in a problem with multiple objectives, it is generally impossible to obtain a total-ordering (a ranking) of all of the alternative solutions, without invoking further rules or assumptions. This means that ‘pure’ optimization, in which an unambiguously best solution is sought, may not be possible. This problem of ranking solutions arises whenever we must compare two solutions that offer a different compromise of the different criteria — with one scoring better on one criterion, and the other scoring better on another criterion. In this situation, the decision as to which solution is actually better may become somewhat subjective, or must rely on additional information, such as the ‘importance’ of each criterion. In any case, the solutions cannot be ranked (totally ordered) by their evaluation alone. Thus, in a broad sense, *multiobjective optimization* (MOO) really entails two entirely different tasks: search and *decision-making*. Search is needed to find solutions, and decision-making is needed for ranking them.

The study of methods for making choices between solutions that offer a different compromise of criteria is a scientific and mathematical discipline in itself, separate from search, called multicriteria decision-making (MCDM). MCDM essentially entails methods for *scalarizing* the vector evaluation of a solution, so that a total ordering of solutions can be obtained, from which the ‘best’ one can be chosen. Scalarizing methods in turn involve techniques for equalizing the ranges of different criteria, and for mathematically modeling the ‘preferences’ that (human) expert decision-makers (DMs) have, when faced with making compromise choices between solutions. For a concise but extensive overview of methods for performing MCDM, see [Mie01].

In this thesis, we do not concern ourselves with methods for performing multiobjective optimization in the broad sense alluded to above. Rather, we are concerned only with what is sometimes called *vector optimization* [Mie01], or more commonly, *Pareto optimization*. This is a pure optimization process, not involving any decision-making. In the next section we define Pareto optimization formally, but here we note that it is a purely mathematical concept which follows directly as a consequence of the definite, total ordering of solutions that exists in terms of each single objective taken separately, and the fact that we can always decide which of two candidate solutions  $X$  and  $Y$  is better, if  $X$  is better than  $Y$  on one or more objectives, and  $X$  is worse than  $Y$  on none.

## 3.2 Concepts and notation

**Definition 3.1** **Multiobjective optimization** is the process of finding one or more vectors of decision variables that simultaneously satisfy all feasibility constraints and optimize a vector objective function that maps the decision variables to two or more performance criteria or objectives.

**Definition 3.2** A **decision vector** (also **solution**)  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ , is a vector of decision variables (also **parameters**), representing the numerical qualities for which values must be found in an optimization problem. The variables may be integer, real, or a mixture. The set of all decision vectors for a given optimization problem is called the **decision space** (also **parameter space**), and is denoted by  $X$ .

**Definition 3.3** The set of decision vectors (solutions) that satisfy all feasibility constraints is called the **feasible set**, and is denoted  $X_f$ .

**Definition 3.4** The vector **objective function**  $\mathbf{f}$  maps the decision vectors from the decision space into a  $K$ -dimensional **objective space** (also **criterion space**)  $Z \subset \mathbb{R}^K$ ,  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ ,  $\mathbf{f} = (f_1, f_2, \dots, f_K)$ ,  $\mathbf{z} \in Z$ ,  $\mathbf{x} \in X$ , where  $\mathbf{z}$  may be called the **objective vector**, the **criterion vector**, or, simply, the **point**.

**Definition 3.5** The image of  $X_f$  in objective space is called the **feasible region** in the objective space and is denoted by  $Z_f$ .

**Definition 3.6 (Multiobjective Optimization — mathematical form)**

$$\text{"Minimize" } \mathbf{z} = \mathbf{f}(\mathbf{x}) \quad (3.1)$$

$$\text{Subject to } \mathbf{x} \in X_f. \quad (3.2)$$

Notice that the definition is in the minimization form. Of course, in general we may have to either maximize all the objective functions, minimize all the objective functions, or minimize some functions and maximize others. However, any objective function can be always be converted from the minimization form to the maximization form, and vice versa since:

$$\max(f_i(\mathbf{x})) = -\min(-f_i(\mathbf{x})) \quad (3.3)$$

and

$$\min(f_i(\mathbf{x})) = -\max(-f_i(\mathbf{x})) \quad (3.4)$$

so that the optimization problem can always be stated as a pure minimization (or pure maximization) problem.

In any case, the definition given above is not fully specified. The term ‘Minimize’ appears in quotes to remind us that the exact meaning of the (vector) minimization must be specified before optimization can be performed.

In order to find the minimum (or set of minima) of a set of (objective) vectors, it must be possible to order them. To do this one must select a type of binary relation that can be used to form a (partial) order between all objective vectors in  $Z$ . Many different order relations can be used, including lexicographic order, component-wise order, and max-order. In this thesis, only component-wise order will be considered, which forms the basis of *Pareto optimization*.

**Definition 3.7** *The component-wise order relation  $<$  is defined as:  $\mathbf{z}^1 < \mathbf{z}^2 \Rightarrow z_i^1 \leq z_i^2, i = 1..K \wedge \mathbf{z}^1 \neq \mathbf{z}^2$ .*

**Definition 3.8** *The weak component-wise order relation  $\leq$  is defined as:  $\mathbf{z}^1 \leq \mathbf{z}^2 \Rightarrow z_i^1 \leq z_i^2, i = 1..K$ .*

### 3.2.1 Pareto optimization

**Definition 3.9** *A solution  $\mathbf{x}^* \in X_f$  is called **Pareto optimal** if there is no  $\mathbf{x} \in X_f$  such that  $\mathbf{f}(\mathbf{x}) < \mathbf{f}(\mathbf{x}^*)$ . If  $\mathbf{x}^*$  is Pareto optimal,  $\mathbf{z}^* = \mathbf{f}(\mathbf{x}^*)$  is called **efficient**. If  $\mathbf{x}^1, \mathbf{x}^2 \in X_f$  and  $\mathbf{f}(\mathbf{x}^1) < \mathbf{f}(\mathbf{x}^2)$  we say  $\mathbf{x}^1$  **dominates**  $\mathbf{x}^2$  and  $\mathbf{z}^1 = \mathbf{f}(\mathbf{x}^1)$  **dominates**  $\mathbf{z}^2 = \mathbf{f}(\mathbf{x}^2)$ . The set of all Pareto optimal solutions  $\mathbf{x}^* \in X_f$  is  $X^*$ , the **Pareto set**. The set of all efficient points  $\mathbf{z}^* = \mathbf{f}(\mathbf{x}^*) \in Z$ , where  $\mathbf{x}^* \in X^*$ , is  $Z^*$ , the **efficient set** or the **Pareto front**.*

**Definition 3.10** *For any two decision vectors  $\mathbf{x}^1$  and  $\mathbf{x}^2$ ,*

$$\mathbf{x}^1 \prec \mathbf{x}^2 \quad (\mathbf{x}^1 \text{ dominates } \mathbf{x}^2) \quad \text{iff} \quad \mathbf{f}(\mathbf{x}^1) < \mathbf{f}(\mathbf{x}^2) \quad (3.5)$$

$$\mathbf{x}^1 \preceq \mathbf{x}^2 \quad (\mathbf{x}^1 \text{ weakly dominates } \mathbf{x}^2) \quad \text{iff} \quad \mathbf{f}(\mathbf{x}^1) \leq \mathbf{f}(\mathbf{x}^2) \quad (3.6)$$

$$\mathbf{x}^1 \sim \mathbf{x}^2 \quad (\mathbf{x}^1 \text{ is incomparable to } \mathbf{x}^2) \quad \text{iff} \quad \mathbf{f}(\mathbf{x}^1) \not\leq \mathbf{f}(\mathbf{x}^2) \wedge \mathbf{f}(\mathbf{x}^2) \not\leq \mathbf{f}(\mathbf{x}^1). \quad (3.7)$$

**Definition 3.11** *A solution  $\mathbf{x} \in X_f$  is said to be **nondominated** with respect to a set*

$X_a \subseteq X_f$  iff

$$\nexists \mathbf{x}_a \in X_a, \mathbf{x}_a \prec \mathbf{x}. \quad (3.8)$$

In some circumstances it may be clear from the context which set  $X_a$  is meant, in which case it can be omitted. Often, the set  $X_a$  is a set of solutions that has been found by a multiobjective optimization approximation algorithm.

**Definition 3.12** Let  $X_a \subseteq X_f$  and  $Z_a = \mathbf{f}(X_a)$ . Let the function  $\text{ND}(X_a)$  return the set of nondominated solutions from  $X_a$ :

$$\text{ND}(X_a) = \{\mathbf{x}_a \in X_a \mid \mathbf{x}_a \text{ is nondominated with respect to } X_a\}. \quad (3.9)$$

Similarly we define the function  $\text{ND}(Z_a)$  as follows:

$$\text{ND}(Z_a) = \{\mathbf{z}^i \in Z_a \mid \nexists \mathbf{z}^j \in Z_a \text{ such that } \mathbf{z}^j < \mathbf{z}^i\}. \quad (3.10)$$

The set  $\text{ND}(X_a)$  is the **nondominated set** with respect to  $X_a$ ; the corresponding set of objective vectors  $\text{ND}(Z_a)$  is the **nondominated front** with respect to  $Z_a$ .

**Definition 3.13** We define the following relations between a vector  $\mathbf{z}^a$  and a nondominated front  $Z_{nd} = \text{ND}(Z_{nd})$ :

$$\mathbf{z}^a < Z_{nd} \iff \exists \mathbf{z} \in Z_{nd}, \mathbf{z}^a < \mathbf{z}, \quad (3.11)$$

$$\begin{aligned} Z_{nd} < \mathbf{z}^a &\iff \exists \mathbf{z} \in Z_{nd}, \mathbf{z} < \mathbf{z}^a, \text{ or in other words} \\ Z_{nd} < \mathbf{z}^a &\iff \text{ND}(\{\mathbf{z}^a\} \cup Z_{nd}) = Z_{nd} \wedge \mathbf{z}^a \notin Z_{nd}, \end{aligned} \quad (3.12)$$

$$\mathbf{z}^a \sim Z_{nd} \iff \mathbf{z}^a \not\prec Z_{nd} \wedge Z_{nd} \not\prec \mathbf{z}^a \wedge \mathbf{z}^a \notin Z_{nd}. \quad (3.13)$$

**Definition 3.14** Similarly, we define the equivalent relations between a solution  $\mathbf{x}^a$  and a nondominated set  $X_{nd} = \text{ND}(X_{nd})$ :

$$\mathbf{x}^a \prec X_{nd} \iff \exists \mathbf{x} \in X_{nd}, \mathbf{x}^a \prec \mathbf{x}, \quad (3.14)$$

$$\begin{aligned} X_{nd} \prec \mathbf{x}^a &\iff \exists \mathbf{x} \in X_{nd}, \mathbf{x} \prec \mathbf{x}^a, \text{ or in other words} \\ X_{nd} \prec \mathbf{x}^a &\iff \text{ND}(\{\mathbf{x}^a\} \cup X_{nd}) = X_{nd} \wedge \mathbf{x}^a \notin X_{nd}, \end{aligned} \quad (3.15)$$

$$\mathbf{x}^a \sim X_{nd} \iff \mathbf{x}^a \not\prec X_{nd} \wedge X_{nd} \not\prec \mathbf{x}^a \wedge \mathbf{x}^a \notin X_{nd}. \quad (3.16)$$

**Note 3.1** The notation and concepts presented here are not unique in the literature. Unfortunately, there has not been a common standard in describing these concepts, adopted by

*all. However, in this thesis, the notation, wording and meanings defined above will be used wherever possible. But, in describing other authors' work we may adopt their notation temporarily in order to make direct quotation of their work possible. In this case, a footnote acknowledging the change in notation will be included.*

### 3.3 Approximating the efficient set

In Pareto optimization, the optimization goal is to find the global Pareto set of solutions, or alternatively, the efficient set of points. The latter is usually sufficient because we would not normally require more than one solution having identical attributes. In any case, given any reasonably large *NP*-hard problem, this optimization goal cannot be reliably achieved using an exact algorithm. Thus, as in scalar optimization, the problem will be to find solutions that approximate this ideal, in a reasonable amount of time. But, it is not so clear how to judge the quality of an approximation to the optimization goal in Pareto optimization, as this involves measuring the quality of a *set* of solutions (or points) and comparing this with the desired, global Pareto set (or efficient set). One means of judging the quality of an approximation to the efficient set might be to measure the number of true efficient points found, and compare this with the number in the efficient set. Of course, this assumes that the efficient set is known, which it may not be. However, even if it is known, this measure of quality cannot differentiate between two different sets where neither contains a single efficient point. Considering this, we might argue that we need to measure the distance of the discovered points from the efficient set, in objective space. But doing this is not trivial because it will necessarily involve combining measures in completely different objectives. Furthermore, even if the distance is zero by this measure (on average over the solutions found, say) it does not differentiate between different numbers of solutions being discovered, or how they are distributed along the Pareto front. Thus, we can see that distance from the Pareto front does not alone capture the quality of an approximation to the efficient set: some means of measuring the distribution of nondominated points found is also required. From this, we can see that Pareto optimization has at least two separate dimensions of success: distance of points from the Pareto front; and distribution along it. Thus, algorithms for approximating the Pareto front should aim to address *both* these issues.

## 3.4 General-purpose Pareto optimization schemes

In Pareto optimization, only a partial ordering of solutions is available, resulting in three alternative methods for carrying out selection in an EA or other search metaheuristic [Hor97b]:

1. Consider just one objective in isolation, each time a solution is evaluated for selection. This method is known as criterion selection.
2. Utilize the partial ordering directly to perform selection. This is Pareto selection.
3. Scalarize the different objectives, using a parameterized function, and vary the parameters such that a diverse set of solutions may be found. This is called scalarizing selection.

### 3.4.1 Criterion selection

The first, pioneering work in the evolutionary multiobjective optimization (EMOO) field was Schaffer's vector evaluated genetic algorithm (VEGA) [Sch84]. Its selection scheme worked by building up the intermediate population in fractions. Each fraction of the intermediate population was selected from the current generation using a different component of the objective vector to assign fitness. VEGA works well, although it has a tendency to favour extreme solutions to the detriment of solutions that represent a compromise of the different objective attributes, particularly when the shape of the Pareto front is non-convex. Two other EAs used criterion selection: a tournament selection method based on comparing pairs of individuals on one chosen objective was proposed by Fourman in 1985 [Fou85], and, in 1991, Kursawe devised a method based on deleting the worst-performing fraction of the population according to one objective at a time [Kur91]. Both methods are discussed in more detail in [FF95]. Criterion selection has not gained much popularity since these early algorithms because other methods of selection seemed to exhibit better behaviour. However, some new methods based on different forms of criterion selection have been recently devised [GETA99, SB00].

### 3.4.2 Pareto selection

Selection mechanisms based on Pareto dominance relationships have been the most prevalent in the population-based algorithms used in the EMOO community. In 1989, Goldberg [Gol89] first suggested an elegant method of ranking a population of solutions, based on their mutual dominance relations. His nondominated sorting method effectively 'peeled off' Pareto fronts

one at a time from the population, assigning maximum fitness to all those solutions in the first layer, and progressively less fitness to successive layers. The nondominated sorting method was implemented in an algorithm, NSGA, by Srinivas and Deb [SD94] in 1994. The NSGA has since been one of the most popular and successful methods in EMOO and fuelled interest in the field. Similar methods of selection, based on Pareto ordering of solutions, have been devised by Fonseca and Fleming [FF93], Horn and Nafpliotis [HN93], and others.

These population-based algorithms all exploit the knowledge in the whole (or a sample) of the population in order to perform selection. This is achieved by performing many dominance comparisons between solutions in order to compute fitness. The advantage of such an approach is that the current store of solutions is being used to drive the search towards the Pareto front in all directions simultaneously. This is because newly generated solutions are judged by their dominance relationships with all other solutions. Commonly, speciation methods are also used to ensure that an even and diverse spread of points across the internal Pareto front is achieved, further increasing the efficiency and efficacy of these methods. The disadvantage of this approach is the high computational cost of performing so many comparisons, particularly when large populations are involved, or there are many objectives. However, this criticism is often dismissed using arguments that, in most real-world applications, the cost of comparisons is irrelevant compared to the time expended in evaluating solutions.

In single-point search methods, there has been little use of Pareto selection. This is because only two solutions, the current and candidate (mutant) solution, are compared at each step. Frequently, pairs of solutions ( $\mathbf{x}, \mathbf{x}' \in X$ ) will be *nondominated* with respect to each other. That is, neither is better than the other on all objectives. Thus, it is not possible to judge accurately which of the current and mutant solutions is better, and much of the selection pressure may be lost.

### 3.4.3 Scalarizing selection

The advantage of using scalarizing functions, in which the vector of objectives is mapped to a single objective value, is that standard selection or acceptance mechanisms can then be used without modifications. This has led to the great popularity of the approach in the operations research and multi-criteria decision making (MCDM) communities. Adaptations of both tabu search [GF98, GMF96, Han97b], and simulated annealing [CJ98, Ser94, UTFT99] have used these methods. Most of the methods store the internally efficient solutions found but they are not used further in the search. In most of the algorithms, a random weighting of the scalarizing function is simply chosen at each step, and the new solution generated is accepted

or rejected according to its utility on this measure compared with the current solution. This method does not take advantage of the information from the previously found solutions, nor does it direct the search in any particular direction. These factors seem to suggest that such an approach would be inefficient compared to the latest elitist Pareto MOEAs, but there is little empirical evidence of this fact, to date.

Some researchers argue that the use of such scalarizing vectors more naturally allows the preferences of the decision maker to be used to guide the direction(s) of the search towards the region(s) of interest (see for example [CJ98]). This may be true, although good methods for directing the search towards desirable regions of the moving Pareto front within Pareto ranking methods are now available [Deb99b, PCWB00]. Also, the use of randomly selected utility functions in the absence of such preference information, as used in many of the MCDM algorithms, seems wholly unsatisfactory.

Examples of the use of scalarizing functions are few in population-based Pareto optimization approaches. Hajela and Lin [HL92] proposed a GA using weighted-sum aggregation of objectives. In their method, the parameters of the weight vector were encoded on the chromosome, thus cleverly providing an implicit mechanism for maintaining diversity in objective space. Later, Bentley and Wakefield [BW97] put forward and tested a number of different weighting schemes for providing a sub-set of Pareto-optimal solutions. But these methods have not caught on, and in the case of [HL92], there is now some evidence that this approach is less efficient than some pure Pareto EAs [ZDT00]. More recently, however, memetic algorithms for multiobjective optimization put forward by Ishibuchi and Murata [IM96] and Jaszkiewicz [Jas98] has also used linear scalarizing functions.

#### 3.4.4 A two-dimensional classification of methods

If we consider the choice between the different strategies of selection outlined above on the one hand, and the choice between local-search, population-based, or hybrid (memetic) approaches on the other, we can obtain a broad, two-dimensional classification of general-purpose Pareto optimizers. Table 3.1 presents such a classification for some of the methods reviewed in this chapter.

From Table 3.1, it is possible to gain an appreciation of the relationships between different methods of Pareto optimization. The popularity of Pareto-based EA approaches, represented in the top left element of the table, is clearly visible. These methods have been successfully applied to many problems in the EMOO literature. However, very little comparison has been carried out between them and methods in other regions of the table. Thus, nothing is really

### 3.4 General-purpose Pareto optimization schemes

|                             | Pareto Selection  | Scalarizing Selection  | Criterion Selection                              |
|-----------------------------|---|--|--|
| Population-based approaches | MOGA [FF93]<br>NPGA [HN93]<br>NSGA [SD93]<br>SPEA [ZT98b]<br>PESA [CK00]<br>NSGA-II [DAPM00b] | GA with weights on chromosome [HL92]<br>Weighting schemes for GAs [BW97]   | VEGA [Sch85]<br>ES [Kur91]<br>Parallel ES [SB00] |
| Memetic approaches          | <b>M-PAES</b> [KC00c]   | MOGLS [IM96]<br>RD-MOGLS [Jas98]   | -  |
| Local-search approaches     | <b>(1+1)-PAES</b> [KC99b]<br><b>Other PAES variants presented in this thesis</b>              | Multiobjective SA [Ser94]<br>A TS procedure [GMF96]<br>MOTS [Han97b]<br>Pareto SA [CJ98]<br>MOSA method [UTFT99] | -  |

Table 3.1: Some of the general-purpose methods for Pareto optimization, classified by selection mechanism and overall search strategy.

known about the relative performance of the Pareto EAs and the scalarizing local-search approaches, for example.

Scalarizing selection and criterion selection are less popular in EAs than Pareto-based approaches and there is some evidence that Pareto EAs are more effective and efficient than EAs based on these other selection strategies, on some problems at least [ZT98b].

Scalarizing selection has been used in a number of local-search and memetic approaches, mainly proposed by researchers from the operational research (OR) community, and have been applied to various combinatorial optimization problems. However, criterion selection has not, to our knowledge, been used in anything other than EAs, to date.

The methods proposed in the following chapters of this thesis, namely PAES and M-PAES, complete the table. They use Pareto selection in a local-search and memetic approach, respectively, and to our knowledge are unique in doing so. They are methods whose performance can be directly compared with today's Pareto-based EAs, which should contribute to greater understanding as to the (types of) problems where local-search and hybrid approaches might be effective. In future work, these algorithms may also be compared with the local-search scalarizing selection methods of the OR community.

### 3.5 Elitism in multiobjective evolutionary algorithms

Elitism is the retention of good parents in the population from one generation to the next, to allow them to take part in selection and reproduction more than once and across generations. Elitism in (single-objective) GAs was investigated as far back as 1975 by Kenneth De Jong [DeJ75] according to Eshelman [Esh97]. Evolution strategies used what would now be called an elitist strategy from their beginning (1971) [JFS97], although in the terminology of ESs it is called a plus selection strategy; non-elitist comma selection strategies came slightly later in 1974-1975 [JFS97].

The concept of elitism may be simple but there are many different instantiations of it, and all of them have slightly different goals and effects. In generational GAs, elitism can be incorporated by selecting deterministically the best individual(s) in the population and placing a copy (copies) into the next generation. Then the remainder of the next generation population are produced in the normal way using some stochastic selection scheme, mating, and reproduction. In steady-state GAs [Whi89, Sys89], the concept of “a generation” is softened because only one mating is allowed per cycle. Thus, almost all of the population members last for multiple “generations”. In some respects, this means that steady-stage GAs are elitist: they have the property, as in generational elitist GAs, that the best fitness in the population monotonically increases over time. But, in another sense, steady-state GAs are not strictly elitist because they do not bias the reproductive selection towards the best parents, instead they bias the replacement selection to the worst parents, ensuring that the best (and many other population members) will last for many cycles.

The first multiobjective evolutionary algorithms employing elitism seem to have appeared at approximately the same time as MOGA, NSGA, and NPGA were put forward, around 1993-4, judging by Horn’s comprehensive review of the MOEA field [Hor97a], although Kursawe [Kur91] proposed a (1+1)-ES for vector optimization in 1991. In Horn’s terminology, “Pareto elitist selection”, means dividing the population into two ranks: the dominated and the nondominated, and biasing the reproductive selection towards the nondominated individuals. According to [Hor97a], Belegundu *et al.* [BMSC94] use selection from the nondominated individuals only, whereas Tamaki *et al.* [TMA95] use a mixed strategy based on copying the nondominated individuals into the next generation, and using criterion selection to make up the remaining individuals. Takada *et al.* [TYK96] apply mutation and recombination first to generate an intermediate population and then make the new population by selecting only nondominated individuals from among the old and intermediate populations.

In other elitist MOEAs, the strategy of elitism is combined with the maintenance of an

‘external population’ of solutions that are nondominated among all those found so far. Horn [Hor97a] considers such an external population is essential to any MOEA implementation, and this view is echoed by Veldhuizen [Vel99]. However, there are many ways to incorporate elitism with the external population. Cieniawski *et al.* [CER95] seem to be the earliest exponents of using the external population as part of an elitist strategy, rather than simply as a repository of discovered solutions [Hor97a]. In their MOEAs, members of the external population are re-injected into the population, replacing randomly selected individuals. Zitzler [Zit99] notes that some slightly later schemes [AL96, ALG96, MIT96, TS97] control the amount of elitism by selecting just the best  $k$  individuals from the current population to survive to the next generation.

Unfortunately, many of these early elitist MOEA schemes were largely unread by researchers in the EMOO field, at least until Horn’s review in 1997 and the theses of Zitzler and Veldhuizen in 1999. Thus there was little real investigation of the effects of these different strategies until very recently. The first elitist MOEA paper to be published in the more mainstream evolutionary computation literature did not appear until 1998 [PM98]. In this paper, Parks and Miller describe a MOEA that maintains an ‘archive’ of nondominated solutions, similar to an external nondominated set, but limited in size. They only archive members of the main population into the archive if they are sufficiently dissimilar from any already stored. Reproductive selection takes parents from both the main population and the archive. Parks and Miller investigate the effects of different degrees of selection from each, and also different strategies for selecting from amongst the archive, including how long individuals have remained there.

Recently, some theoretical justification for the use of elitism in MOEAs was given in a convergence proof by Rudolph [Rud98a], although Hanne [Han99] notes that selection strategies where global convergence cannot be proved may work better in practice than those where convergence can be proved.

The most well-known elitist MOEA, SPEA [ZT98a] developed just prior to much of the work in this thesis, has been shown to be very effective on a range of test problems [ZDT00] and combinatorial optimization problems [ZT99], compared with other non-elitist MOEAs. Zitzler also formulated a general framework (or unified model) for elitism in MOEAs in his PhD thesis [Zit99], that has since been extended in [LZT00b].

Even more recently, Deb *et al.* put forward an elitist and more efficient version of Srinivas and Deb’s NSGA [DAPM00b]. Comparison of the new NSGA-II against PAES and SPEA on five test functions seems to show that the new algorithm performs well in terms of both distance from the true Pareto front and also the distribution of nondominated vectors found. However,

the results are questionable in one respect. Deb uses a real-valued mutation operator in NSGA-II and does not extend this to either PAES or SPEA. This could explain why NSGA-II gets closer to the Pareto front than PAES or SPEA.

The elitist models of SPEA, NSGA-II, and those presented in this thesis — PAES and M-PAES — as well as PESA [CK00] have been used to tackle a variety of problem domains and it is now more generally accepted that elitism in multiobjective search is at least as important as in scalar optimization. With this advance, the problem of controlling elitism in MOEAs is now coming under closer scrutiny. In [DG00], Deb *et al.* investigates a method in which a set number of solutions are placed in each equivalence class at each generation so as to control the elitism. This is shown to improve the convergence properties of NSGA-II on some constrained test functions. And using the unified model developed by Zitzler, Laumanns *et al.* [LZT00a] investigate various (elitist) selection schemes, diversity maintenance techniques, and mutation strengths. They find that mutation strength and elitism are strongly linked: elitist MOEAs work well with a higher mutation strength, whereas non-elitist MOEAs only work effectively with a relatively lower rate.

### 3.6 Obtaining a well-distributed approximation set

In section 3.4 we identified a number of different ways to undertake Pareto optimization. In particular, different methods were broadly classified based on whether search was population-based or based on local moves, or a hybrid of the two, and by the type of selection strategy used. These different approaches to the Pareto optimization problem should all aim to obtain a ‘well-distributed’ approximation to the true efficient set. However, the different features of the approaches mean that for some of them, obtaining diverse points in the nondominated front does not require any additional mechanisms explicitly for this purpose. For example, in a strategy where a standard (single-objective) GA is run multiple times using a different weighted scalarization of the objectives, additional techniques for diversity maintenance may not be required at all. Similarly, local-search methods based on scalarizing selection do not need niching mechanisms because the use of diverse scalarizing vectors already encourages finding diverse points in the objective space. Indeed, some researchers in the MCDM field regard it as a weakness of Pareto selection-based MOEAs that explicit niching is required to make them work [Jas98]. Nonetheless, in the field of evolutionary multiobjective optimization, Pareto selection genetic algorithms employing some form of diversity maintenance technique have been by far the most popular algorithms in the literature [Vel99]. Thus, in the following we focus mainly on the important issue of niching methods for use in Pareto

MOEAs, before giving a brief discussion about diversity mechanisms used in other Pareto optimization methods.

#### 3.6.1 Niching and diversity maintenance in EAs and MOEAs

In natural evolutionary ecosystems, competition does not exist primarily between different species; it exists within environmental “niches”, which are believed to be the cause of the different species forming, and thus the cause of the diversity of life within the ecosystem. The different niches can be viewed as different local optima in a multimodal fitness landscape, and the different species evolve around these different peaks. Thus researchers in EAs interested in multimodal optimization, view speciation as a valid means of maintaining diverse solutions so that multiple optima can be explored.

The need for some form of diversity promotion or maintenance in EAs is related to the general exploitation/exploration tradeoff that must be balanced in all search and learning systems. Selection in EAs “drives the population toward a uniform distribution of  $N$  copies of the most highly fit individual” [Hor97b]. This can lead to premature convergence to a suboptimal solution, so that diversity maintenance is also important even when only one (optimal) solution is desired. Maintaining diversity in the face of selection pressure helps to avoid genetic drift [AM94], where the population converges towards a less fit region from multiple similarly fit sampled regions, through stochastic sampling effects (errors) in the selection procedure.

Many mechanisms for promoting speciation and the maintenance of diversity have been proposed and studied in the literature. Before describing these, we can identify several features or dimensions of quality of the different approaches:

1. Time complexity — particularly in terms of the population size.
2. Selection pressure and the exploration/exploitation tradeoff.
3. What the measure of diversity is: genotypic vs phenotypic in EAs, and different tradeoffs in objective space in MOEAs.
4. Accuracy and stability — how closely the method approaches the desired number of solutions on each optimum (usually related to the fitness of the optimum) and how stably it maintains these numbers as selection continues i.e. in the steady state.
5. Robustness to optima of different sizes and shapes, and to optima distributed non-uniformly in the search space.

6. Parameterization/self-adaptiveness. How much *a priori* knowledge is needed about the size, position and number of optima in the solution space in order to apply the technique, how many parameters need to be set, and how much robustness is there if parameters are not set accurately.

The schemes of *crowding* and its close relative, *preselection*, are two of the oldest methods of maintaining population diversity under selection pressure in EAs, dating back to 1975 [DeJ75] and 1970 [Cav70], respectively [MG92]. Crowding is used in EAs that employ a generation gap, i.e., only a fraction of the population reproduces and dies in each generation. (Both steady-state EAs and generational EAs are just special (extreme) cases of EAs with a generation gap.) In a generation gap EA, after reproductive selection and reproduction, parents from the previous generation must be selected for replacement by the newly generated offspring. In crowding, the idea is that offspring should replace parents that are most similar to them, to maintain diversity. This is achieved by sampling the parent population and finding the closest parent from the sample to die, for each offspring generated. Originally, the similarity between individuals was measured as the Hamming distance between them - which can be viewed as maintaining genetic (bit-wise) diversity, which Mahfoud and Goldberg regard as of questionable value [MG92]. Crowding requires a fairly large degree of computational overhead to measure the Hamming difference between many pairs of individuals. The amount of computation is controlled to some extent by a parameter, the crowding factor  $CF$ , controlling the size of the sample used for choosing parents to replace. However, low values of  $CF$  cause stochastic sampling errors and instability [MG92].

Preselection uses a similar idea to crowding but has a much lower computational overhead. The idea is that children should replace their parents, since one could estimate that a parent would be one of the members of the population most similar to the new offspring. This can be instantiated in a number of different ways. According to [MG92], one of the best methods of preselection is for the offspring to replace the parent with the lowest fitness if it is fitter.

Better than either of these methods, though, is a kind of combination of them put forward by Mahfoud [Mah95], called *deterministic crowding* (DC). In DC, all members of the population are paired up randomly to take part in recombinative reproduction, i.e., there is no reproductive selection pressure. Each pair of parents then produces two offspring. Each offspring then competes deterministically against one of its parents. Which of the two possible ways of pairing up the offspring with the parents to execute this competition is chosen, is determined by minimizing the sum of the absolute differences between parents and offspring. Mahfoud recommends normalized phenotypic distance as the measure of difference, where available. Deterministic crowding has several advantages: it is of low computational cost; it has high

accuracy as measured by the number of peaks it can maintain, and the number of replacement errors it makes; it has no niching-specific parameters, and it adapts itself to the number of peaks in a multi-modal landscape. On the negative side, DC, has low selection pressure compared to some other methods and because it has no parameters, this cannot be adjusted, except by changing the population size. In some applications this may be a disadvantage. Given the history of crowding and the advances made by Mahfoud, it is surprising that it is “rather seldomly implemented in MOEAs” [Zit99]. This may be due to the greater theory and exposure related to fitness *sharing*, considered next.

Fitness sharing is a mechanism whereby the reproductive fitness allocated to an individual in a population is reduced proportionately to the number of other individuals that share the same region of the fitness landscape. In other words, fitness is a resource which is shared amongst the individuals competing in the same niche.

In explicit fitness sharing [GR87], the original fitness allocated to an individual is scaled using a sharing function:

$$F'(i) = \frac{F(i)}{\sum_{j=1}^N sh(d(i, j))} \quad (3.17)$$

where  $F'(i)$  is the adjusted fitness of individual  $i$ ,  $F(i)$  its original fitness,  $N$  the population size, and  $sh(d(i, j))$  is the share value given by:

$$sh(d(i, j)) = \begin{cases} 1 - \left(\frac{d(i, j)}{\sigma_{sh}}\right)^{\alpha_{sh}} & \text{if } d(i, j) < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases} \quad (3.18)$$

The share value is a function of  $d(i, j)$ , the ‘distance’ between  $i$  and  $j$ , and  $\sigma_{sh}$  the niche radius, a parameter setting the radius over which one individual can affect the fitness of another. The parameter  $\alpha_{sh}$  affects the shape of the sharing function. It is often set to one, yielding the ‘triangular’ sharing function.

Sharing may be *genotypic*, where the  $d$  function is a suitable distance function (e.g. the Hamming distance) between two strings, or *phenotypic* where  $d$  should measure some meaningful property of the phenotype. In multiobjective optimization, it may be more appropriate to somehow measure the distance in the multidimensional objective space, i.e. based on where individuals map to in objective space. This will encourage an even distribution of points along the Pareto front.

Explicit fitness sharing has been used widely in multiobjective EAs. The NSGA [SD93] uses phenotypic fitness sharing, whereas Horn and Napfliotis’ NPGA [HN93] is usually used with sharing in the objective space. Horn and Napfliotis explain how to size the niche radius

in the NPGA based on knowledge of the best and worst values in the search space of each objective [HN93]. However, the lower bound calculation is stated wrongly since Horn says that the “minimum length [area] Pareto frontier is the hyperplane connecting these extremes.” This is clearly not true. The minimum area of the Pareto frontier is zero, since in the limit one lone solution may dominate all others in the space. Small Pareto frontiers tend to occur in problems where there is a strong positive correlation between the values of the different objectives. Thus, these calculations for sizing the niche radius in objective space may not work well in the case of positively correlated objectives. In actual fact, one needs to know the extreme values of the Pareto front in order to size the niche radius correctly.

The problems with explicit fitness sharing include its  $O(N^2)$  time complexity, and the difficulty of setting the niche radius correctly. Several new explicit fitness sharing schemes have been proposed which may reduce problems with the latter by using an adaptive niche radius, including [MJ96, GW00]. However, these have not been used in MOEAs to date.

Fitness sharing may also be ‘resource based’, where reproductive fitness is allocated in proportion to the amount of resources (or credits) an individual obtains. The resources available for individuals to collect are finite so that individuals qualifying for a given resource must share it between them, based on their relative ‘rights’ to it. Resource based sharing has been recently used successfully in a MOEA called the evolutionary local selection algorithm (ELSA) [MDS00]. This algorithm is particularly effective at covering the full extent of the Pareto front.

The strength Pareto evolutionary algorithm (SPEA) [ZT98b] is the first MOEA to employ clustering for diversity maintenance. The external population in the SPEA is limited in size and must be periodically reduced. To perform this reduction a standard clustering algorithm is used, which clusters the members of the external population in the objective space, and removes individuals from the densest clusters. Clustering may have the advantage over sharing because it is more flexible and adaptive, requiring less knowledge of the size and shape of the Pareto front.

There are a few other approaches to promoting and maintaining diversity in a population which have seen little application in multiobjective optimization. Spatially structured populations and/or island models, where mating is restricted to individuals that are close to one another geographically, can encourage the formation of niches. Mating may also be restricted to individuals that are phenotypically similar [SD93]. This has been used recently in the RD-MOGLS algorithm [Jas98] to solve multiobjective travelling salesman problems. Finally, multiploidy [CCR96, Kur91, YA94], where the loss of genetic diversity is prevented by storing information which may not be expressed, has not been investigated at all in MOEAs.

### 3.6.2 Encouraging diversity in methods based on scalarizing selection

In Pareto optimization approaches based on scalarizing selection, diversity of the discovered objective vectors is encouraged by varying the parameters of the scalarizing function. In some methods, such as CHESS and MOSA method [UTFT99, Bor00], search is carried out using multiple runs, with each run using a particular parameterization of the scalarizing function. In other methods, such as MOGLS [IM96], the weight vector is varied *during* a run. In all of these approaches, however, the direction of search (specified by the scalarizing function's parameters) does not respond to the evolving distribution in objective space, of the discovered nondominated points. Thus, it is quite possible that some regions of the objective space will become much more densely populated than others.

Two approaches that use a reactive specification of search directions, to encourage a more even distribution of points, are the tabu search procedure of Gandibleux *et al.* [GMF96], and the Pareto simulated annealing algorithm put forward by Czyzak and Jaszkiewicz [CJ98]. In the former, the scalarizing weights are updated after each iteration of the tabu search. The weights given to objectives where a large improvement has occurred in the last ‘move’ are reduced, whereas the weights given to objectives where the improvement is small or no improvement has occurred are given a larger weight in the next iteration.

In Czyzak and Jaszkiewicz’ Pareto SA algorithm, each member of a population uses a simulated annealing-style acceptance function to search in its neighbourhood. Each member has its own scalarizing weight vector to specify the direction in which it is trying to improve. Initially, the weight vectors are set randomly. However, the weights of an individual  $\mathbf{x}$  are updated at each iteration, in order to increase the probability of moving it away from its closest, nondominated neighbour in the population,  $\mathbf{x}'$ . This is achieved by increasing the weights of the objectives on which  $\mathbf{x}$  is better than  $\mathbf{x}'$ , and decreasing the weights on which  $\mathbf{x}$  is worse than  $\mathbf{x}'$ . This will tend to evenly distribute the vectors in each equivalence class.

## 3.7 Where to go from here

In the previous sections we have reviewed some of the key steps in the development of modern MOEAs, and considered the important elements of these algorithms. We have seen how the early algorithms employed different selection methods for dealing with the multiple attribute values, and how the successes and failures of these early schemes affected the direction of the field. Pareto selection quickly became easily the most prevalent of selection methods in the evolutionary camp, with the most popular schemes proposed by Srinivas and Deb,

Fonseca and Fleming, Horn and Nafpliotis, and most recently, Zitzler and Thiele. In the MCDM community, however, parameterized scalarizing selection has been favoured, probably because scalarizing methods are routinely used in the decision-making process to convert the DM's preferences into a single value of utility. Using their expertise with scalarizing methods, it was simple to convert well-understood local search algorithms such as SA to work with multiple attribute values. Thus, although within the evolutionary camp, Pareto selection is most popular, and believed by some to be superior to other selection schemes, this view is not implicitly accepted by the MCDM community. It remains an important avenue of future research for the two communities to take a closer look at these selection methods, and to compare their performance on a variety of problem types.

Shortly after the first MOEA schemes were proposed, methods for niching became important to the mainstream evolutionary field for multimodal optimization. It was quickly appreciated that these methods could also be vital for multiobjective optimization, and they were incorporated into the MOEAs put forward in the early nineties. Explicit fitness sharing has been the most prominent method for niching in MOEAs but it suffers from two problems: it can be difficult to set the parameters and it is computationally expensive. Further work should focus on methods more suited to the particular needs of niching in objective space; adapting automatically to the size of the Pareto front and minimizing the number of comparisons necessary between solutions.

The incorporation and control of elitism has been the most important issue concerning the EMOO field in more recent years. We have seen how elitism in multiobjective search is not merely a question of keeping the best solution(s) from the current population; it is more concerned with the interaction between the current nondominated solutions, and the reproductive population. Most of the new MOEAs maintain nondominated solutions in an external store and this may be used to re-insert solutions into the population, or they may take part in selection. A number of different elitist schemes have been investigated but the issue of controlling the size of the external store has not been investigated very thoroughly to date. Zitzler's SPEA uses a computationally expensive clustering algorithm to reduce the size of the external population periodically to allow a bounded population of nondominated solutions to take part in reproduction. The RD-MOGLS algorithm of Jaszkiewicz, which has demonstrated good performance on the multiobjective knapsack problems used by Zitzler [Jas00], works by using a large queue to store the nondominated solutions found, and then works with a small selection of these to search in a particular direction. The computation time needed to search through this queue to select the mini-population may become very large, however, if the queue is large. If the queue is smaller it may not be a good representation of the current Pareto front. Since it now seems that MOEAs will use such external stores in real

applications, it is time that the computational issues of maintaining this store, and keeping it down to a reasonable size, whilst ensuring it represents a diverse, well-distributed set of solutions in the current nondominated front were investigated.

Whilst the issues of selection, niching, and efficient storage of nondominated solutions are all important areas which need continued research, a more important issue may be to investigate under what circumstances current MOEAs are really more effective, efficient, robust, and easy to use than other methods. MOEAs have not been compared, historically, with either random search or simple hillclimbing methods, on real search spaces of interest. Bucking this trend, Zitzler did compare SPEA and several other MOEAs to random search in his recent investigations of the Deb test functions. This is encouraging, particularly as the results did clearly indicate that all of the MOEAs outperformed random search on this test suite. This kind of verification should also be repeated on other test functions and also real-world problems, to extend this result to a broader range of domains and problem features. However, hillclimbing algorithms may be more formidable opponents to EAs than random search is, as we have seen in single-objective optimization. If the evidence of single-objective optimization is anything to go by, hillclimbers may be effective even in unexpected domains, and certainly wherever the search space is not strongly epistatic, noisy, or deceptive. This would suggest that MOEAs should really be compared against some sort of hillclimber in the future.

Comparisons of this type may be useful in attaining several important goals for the future development of EMOO:

- to learn which existing problems can be tackled most effectively using current population-based MOEAs;
- to learn to predict which problem features tend to suggest the use of current MOEAs is appropriate, and which suggest simpler techniques may be more effective or efficient;
- to stimulate the design of better test functions which can differentiate between different algorithm strengths and weaknesses;
- to investigate whether the generally different topology of multiobjective landscapes may affect the relative merits of local and population-based search;
- and to facilitate the design of more diverse evolutionary algorithms based on local search, or hybrids with a local search element.

A Pareto multiobjective analogue to a single-objective hillclimber, against which MOEAs could be compared, would only usefully serve these goals if it met certain design criteria, however. Overall, hillclimbers are simple; computationally, they do little work between function

evaluations, they use only minimal domain knowledge (for the design of the neighbourhood operator), they can be applied with little effort to a wide range of applications, and they have very few parameters to set or adjust. Despite their simplicity they give good search performance on many problems of interest (better than random search). These attributes of a hillclimber should be maintained, as much as possible, in a Pareto hillclimber.

Good search performance in a multiobjective optimizer necessarily means that a single algorithm run would return a diverse, well-distributed nondominated set of solutions. One method for achieving this in a Pareto hillclimber may be to use parameterized scalarizing selection in a standard hillclimber, and to store all the solutions encountered during an algorithm run. This would be simple to implement, and computationally inexpensive but using scalarizing selection may not meet the design criteria for other reasons. When using scalarization, the objectives have to be appropriately scaled, and their ranges equalized prior to starting the algorithm. If linear scalarizing is used then concave regions of the Pareto front cannot be found. On the other hand, if Tchebycheff utility functions are used then it is necessary to know some ideal point. These considerations mean that it would be better to specify the use of Pareto dominance in the acceptance function.

### 3.8 Multiobjective problems

As in scalar optimization, multiobjective optimization problems can be broadly divided into three categories: discrete or combinatorial optimization problems; continuous problems; and those that are a mix of discrete and continuous variables.

A very thorough and extensive review and classification of problems and methods for multiobjective combinatorial optimization (MOCO) is carried out in [EG00], where twenty different multiobjective combinatorial problem structure types are identified and over 350 papers are cited. References to fifteen design and engineering problems are given in [Deb99a], including those of a continuous and mixed nature. Coello considers eight design/engineering problems in his thesis [CCH98], seven of which are continuous and one of which is combinatorial in nature. As well as these real-world problems, there also exist a number of test functions for multiobjective optimization. These are discussed in more detail below.

### 3.9 MOEA test functions and problems

MOEA performance has been demonstrated, historically, on several, quite simple test functions, which were proposed along with the seminal algorithms of the field, including those of Schaffer [Sch85] and Fonseca and Fleming [FF94]. More recently, the way in which these functions have been used for testing MOEAs has been brought into question [VL99, Deb99a]. Veldhuizen noted in 1999 that of the thirty test functions which had been used in the literature, all but three used at most two decision variables and most were two-objective problems [Vel99]. He concluded that: “...MOEA performance claims/comparisons based on these functions may not be meaningful.” Veldhuizen also noted that the rationale of the test functions used was often unstated and unclear. To improve this situation, he classified the existing test functions along a number of important dimensions, allowing researchers to use these functions more intelligently [Vel99]. In addition, a test suite consisting of a number of these functions was proposed that, when taken together, exhibit a range of different characteristics and “form a coherent basis for MOEA comparisons”. Veldhuizen discretized these problems to a certain level of resolution and solved them through exhaustive search using powerful supercomputers. Thus for these problems, it is now possible to compare results directly with the true Pareto set and Pareto front, up to this resolution.

Deb’s article [Deb99a], approaches the test suite design problem from the other direction from Veldhuizen’s. Rather than classifying available functions in terms of their characteristics, Deb proposes that certain problem features should be identified first, and test functions which isolate these features should then be designed. He notes that few studies have tried to identify those specific problem features which might cause difficulty to a MOEA, and argues that test problems based on such features may enable better comparison of MOEAs with other methods, better understanding of MOEA working principles, and facilitate improvements to current algorithms, as good test suites have in single-objective optimization. The article identifies problem features which may cause difficulties in three areas: convergence towards the Pareto front, maintenance of diversity, and constraint-handling. A generic two-objective optimization problem is then proposed which allows the identified problem features to be introduced in a controlled manner. A sample of problems which may form a basic test suite is also given.

The test function suites of Deb and Veldhuizen begin to address the needs of the MOEA community to test their algorithms effectively. However, both of them also note the limitations of the use of purely numerical function test suites, and encourage more use of combinatorial and real-world applications for testing MOEAs.

### 3.10 An introduction to performance assessment and comparison

Assessing and comparing the performance of heuristic search methods is always difficult because performance is a multidimensional attribute: solution quality; computational effort; and robustness may all be important [BGK<sup>+</sup>95]. However, performance assessment in MOEAs is even more difficult because there are even more dimensions of quality to measure, as a result of the multidimensional nature of Pareto fronts. This is due to two separate factors. First, the quality of individual solutions themselves are now represented by a vector, rather than a scalar value. Second, (at least in Pareto optimization) the quality of solution found is not represented by one single best solution, but rather by a set of nondominated solutions. Thus it is now more difficult to represent solution quality as a simple variable that can be plotted against time, for example, and to measure and report how robust it is to changes in the problem, algorithm parameters, or other factors. Thus, the ideals that are put forward in [BGK<sup>+</sup>95] are far more difficult to achieve in the multiobjective optimization case.

Due to these difficulties it has been common in much of the multiobjective EA literature for researchers to indicate the performance of their algorithm by simply plotting the nondominated solutions (to a 2-objective problem, at least) found upon termination of the algorithm. This is remarked upon by Veldhuizen [VL00]: “Comparative results are then ‘clearly’ shown in graphical form indicating which algorithm performed better, often implying the new MOEA’s returned  $PF_{known}$  is a better representation of  $PF_{true}$ .<sup>2</sup> However, such an approach is not acceptable on its own (although it can be useful in conjunction with the reporting of other results) because it tells us nothing about the robustness of the approach over multiple independent runs (i.e. it has no statistical significance) and furthermore it does not relate the solution quality to the computational time, or the convergence of the algorithm(s).

Several techniques developed more recently do seek to provide results that can summarize and make inferences from the data collected from several independent runs of an algorithm, and that can be used to measure the rate of convergence of a multiobjective algorithm. These methods rely on being able to reduce the very high dimensionality of solution ‘quality’ as represented by a *single* nondominated set of objective vectors, into a much smaller dimensional metric. Several methods for achieving the latter have now been proposed and they work in a variety of different ways: some of them rely on knowing the (entire) true Pareto front; others give an absolute value or values that summarize the quality of a nondominated set, without reference to any other reference set; while still other techniques can be used only to compare

<sup>2</sup>In our notation,  $PF_{known}$  is denoted  $ND(Z)$  where  $Z$  is any approximation of the efficient set. And  $PF_{true}$  is  $Z^*$ , the efficient set.

two or more nondominated sets, or algorithms, against each other.

Of course, reducing the dimensionality of the data represented by a set of nondominated objective vectors into a meaningful measure of quality requires us to first identify desirable global aspects of such a set. Zitzler *et al.* [ZDT00] suggest three goals of Pareto multiobjective search that can be identified and measured:

- The distance of the resulting nondominated set to the Pareto-optimal front should be minimized.
- A good (in most cases uniform) distribution of the solutions found is desirable. The assessment of this criterion might be based on a certain distance metric.
- The extent of the obtained nondominated front should be maximized, i.e., for each objective, a wide range of values should be covered by the nondominated solutions.

Although these three criteria might not seem controversial, since these three aspects do indeed describe desirable outcomes of a multiobjective search, it *is* questionable whether they are completely general, and whether they are a minimal set. For example, if the true Pareto front consists of just one point in objective space then item three is not appropriate. If, on the other hand, the points on the true Pareto front are not uniformly distributed, then an ‘approximation’ that contains nearly all of the points in the true Pareto front will not comply with item two, in the list. Thus, although the above list does serve as an intuitive guide to the goals of Pareto search, it is not completely general. A more general (and economic) statement of the goals of Pareto optimization would be to expand on the first point only, defining what is meant by the distance of one set from the other. This is likely to lead to more useful metrics than the set given above, which may lead to misleading metrics.

More recently, Hansen and Jaszkiewicz have written a detailed report [HJ98] on the subject of evaluating approximations to the true Pareto front. In it they define a number of outperformance relations that express the relationship between two sets of internally nondominated objective vectors,  $A$  and  $B$ :

**Definition 3.15 (Weak outperformance)**  $A \text{ OW } B \iff \text{ND}(A \cup B) = A \text{ and } A \neq B$ .  
*In other words, approximation  $A$  weakly outperforms approximation  $B$  if all points in  $B$  are ‘covered’ by those in  $A$  (where ‘covered’ means is equal to or dominates) and there is at least one point in  $A$  that is not contained in  $B$ .*

**Definition 3.16 (Strong outperformance)**  $A O_S B \iff \text{ND}(A \cup B) = A \text{ and } B \setminus \text{ND}(A \cup B) \neq \emptyset$ . In other words, approximation  $A$  strongly outperforms approximation  $B$  if all points in  $B$  are covered by those in  $A$  and at least one point in  $B$  is dominated by a point in  $A$ .

**Definition 3.17 (Complete outperformance)**  $A O_C B \iff \text{ND}(A \cup B) = A \text{ and } B \cap \text{ND}(A \cup B) = \emptyset$ . In other words, Approximation  $A$  completely outperforms  $B$  if each point in  $B$  is dominated by a point in  $A$ .

Notice that  $A O_C B \Rightarrow A O_S B \Rightarrow A O_W B$ . In other words, complete outperformance is the strongest and weak outperformance is the weakest of the relations. This can also be written as  $O_C \subset O_S \subset O_W$ .

These relations are valuable descriptors of the relationships between approximations to  $Z^*$  because they are compatible with, and only depend upon, the standard dominance relation. They do not of themselves constitute metrics of performance, however, and they cannot be used to determine which of two sets is better in the often encountered case where each set contains points that are not covered by the other set<sup>3</sup>. Nonetheless, we can use these relations to assess the usefulness of other quantitative assessment metrics: any metric which is not compatible with these relations cannot be relied upon to provide evaluations that are compatible with the notion of Pareto dominance.

Hansen and Jaszkiewicz formally defined compatibility and weak compatibility with an outperformance relation, as follows:

**Definition 3.18 (Weak compatibility)** A comparison metric  $R$  is weakly compatible with an outperformance relation  $O$  if for each pair of nondominated sets  $A$  and  $B$ , such that  $A O B$ ,  $R$  will evaluate approximation  $A$  as not being worse than  $B$ .

**Definition 3.19 (Compatibility)** A comparison metric  $R$  is compatible with an outperformance relation  $O$  if for each pair of nondominated sets  $A$  and  $B$ , such that  $A O B$ ,  $R$  will evaluate approximation  $A$  as being better than  $B$ .

---

<sup>3</sup>Pairs of sets like this are said to be incomparable according to the weak outperformance relation. This implies they are also incomparable according to  $O_S$  and  $O_C$  since  $O_C \subset O_S \subset O_W$ .

## 3.11 Classification and analyses of nondominated set assessment metrics

### 3.11.1 The classification scheme

In this section we classify and critically analyse some of the key metrics proposed in the literature for assessing approximations to the efficient set. Specifically, we discuss the purpose and approach of each metric, its compatibility with the outperformance relations discussed above, and several other important factors affecting its usefulness. The results of the analysis are summarized in Table 3.2.

We discuss each of the metrics under four headings:

- Purpose and approach
- Pareto compatibility
- Advantages
- Disadvantages and caveats.

Under the first heading we describe the intended purpose of the metric (e.g. to measure the diversity in the Pareto front, or to measure distribution etc.), and the way in which the metric actually compares two approximation sets  $A$  and  $B$ . There are several alternative approaches to the latter which we briefly describe next.

A metric may compare the two approximations directly using a scalar measure  $R(A, B)$  which describes how much better  $A$  is than  $B$ , and  $R(B, A)$ , vice versa. If  $R(A, B) = c - R(B, A)$  for some constant  $c$  for all pairs of nondominated sets  $A, B$  then  $R$  is ‘symmetric’. Metrics of this comparative type, such as the R1 metric of Hansen and Jaszkiewicz [HJ98], we will call, ‘direct comparative’ metrics.

An alternative approach for comparing two approximations is to use a reference set, perhaps the efficient set, and compare both approximations against this reference set, and then compare the results. These metrics we will call ‘reference’ metrics. Clearly, any direct comparative metric can also be used as reference metric by specifying a particular reference set. However, the converse is not true because some reference metrics, like Veldhuizen’s Error Ratio are defined specifically for a particular reference set (the efficient set).

A third alternative approach is to measure some property of each set that is not dependent

on the other, or any other reference set of points, and compare the results of these two measurements. These metrics, like Schott's spacing metric, we will call 'independent'.

Another important feature of a metric discussed under our first heading is whether or not it induces a complete ordering of all possible nondominated sets. A complete ordering of nondominated sets ensures transitivity, so that when nondominated sets  $A$ ,  $B$ , and  $C$  are compared, if  $A$  beats  $B$  and  $B$  beats  $C$  then it is always true that  $A$  beats  $C$ . Often, direct comparative metrics do not induce a complete ordering of sets so that the relations between different approximation sets may be intransitive. In this case it is sometimes advisable to use these metrics with reference sets (i.e. as reference metrics) to ensure transitivity. Transitivity is not generally a problem with independent metrics as they all induce a complete ordering of nondominated sets, provided they return a single figure of merit only.

Under the first heading we also note if the metric is a cardinal measure (based on counting the number of vectors in some set) or non-cardinal.

The Pareto compatibility section of each analysis is concerned with judging if the metric give scores or judgments that are compatible with the outperformance relations between nondominated sets. The aim of this section is to indicate if a metric can be misleading, giving scores for nondominated sets that do not accurately reflect their relative worth in a Pareto sense. To judge the degree to which a metric is Pareto compatible, its compatibility with the three outperformance relations  $O_W$ ,  $O_S$ , and  $O_C$ , proposed by Hansen and Jaszkiewicz is determined. Recall that a metric can be *weakly compatible* or *compatible* with an outperformance relation.

The hardest relation to be (weakly) compatible with is  $O_W$ , and the easiest is  $O_C$ . We note that compatibility with  $O_W$  is necessary and sufficient for ensuring **monotony** and sufficient but not necessary for ensuring **relativity**, defined as follows.

**(weak) monotony** Given a nondominated set  $A$ , the addition of a nondominated point always improves (never degrades) the set's evaluation.

**(weak) relativity** The evaluation of the global Pareto front is uniquely (non-uniquely) optimal, i.e., all other nondominated sets have a strictly inferior (non-superior) evaluation.

Weak compatibility with  $O_W$  is sufficient for the weak versions to be exhibited.

The Pareto compatibility section also includes some discussion and figures to illustrate examples where the metric fails to perform in a desirable manner.

The final two sections summarize the advantages and disadvantages of the metric by consid-

ering its compatibility with the outperformance relations and the following additional factors:

- the computational cost of the metric;
- whether it is scaling independent (is the ordering of approximations affected if one objective is scaled relative to the others?);
- whether it relies on knowledge of the true efficient set or any other reference point;
- whether its purpose is well-defined;
- and whether it can differentiate between different levels of complete outperformance. This means that given three approximation sets  $A$ ,  $B$ ,  $C$  with  $A \text{ } O_C \text{ } B$  and  $B \text{ } O_C \text{ } C$ , would the metric give a different evaluation if  $A$  and  $B$  were compared than if  $A$  and  $C$  were compared?

#### 3.11.2 The $\mathcal{S}$ metric [Zit99]

A definition of the  $\mathcal{S}$  metric and detailed description of how to calculate it is given in section 3.12.2.

##### Purpose and approach

The purpose of the  $\mathcal{S}$  metric is to evaluate the overall quality of a nondominated set. To achieve this it computes the size of the region dominated by the nondominated set. It is an independent metric (although a single reference point must be chosen to bound the dominated region), so it induces a complete ordering of nondominated sets. It is a non-cardinal measure.

##### Pareto compatibility

It is compatible with  $O_W$  provided that the upper boundary of the dominated region is set so that all feasible nondominated sets are evaluated positive.

##### Advantages

There are many advantages to this metric. It is compatible with the outperformance relations. It is an independent metric, giving a single figure of merit that all nondominated sets can be judged by. It differentiates between different degrees of outperformance of two sets, where

one completely outperforms the other. It is scaling independent. There is no need to know the Pareto set or other reference points to use it. Its meaning is intuitive.

### Disadvantages and caveats

Strictly, it requires defining some upper boundary of the region within which all feasible points will lie. The choice of this boundary does affect the ordering of nondominated sets (see Figure 3.1), and it is difficult to justify any particular choice regarding this. The  $S$  metric has a very large computational overhead,  $O(n^{k+1})$ , which makes it completely unusable for large numbers of objectives or large nondominated sets. However, in most applications in the literature at present, only two or three objectives problems are usually tackled. It multiplies ‘apples’ by ‘oranges’, that is, different objectives together. It can be argued that this does not matter, however, as the metric is scaling independent anyway, and the units are irrelevant.

#### 3.11.3 Error ratio [Vel99]

$$\frac{\sum_{i=1}^n e_i}{n} \quad (3.19)$$

where  $n$  is the number of vectors in the approximation set;  $e_i = 0$  if vector  $i$  is in  $Z^*$  and 1 otherwise. Lower values of the error ratio represent better nondominated sets.

### Purpose and approach

The error ratio measures the ratio, amongst the approximation set ( $PF_{known}$  in Veldhuizen’s terminology), of those vectors that are in the true Pareto front, to those not in the true Pareto front. It is a reference metric using the true Pareto front as reference set. It induces a total ordering of nondominated sets. It is a cardinal measure.

### Pareto compatibility

The error ratio is only weakly compatible with  $O_C$ . It is not weakly compatible with  $O_S$  or  $O_W$  e.g. if an algorithm finds two nondominated vectors, one in  $PF_{true}$ , and the other far away from the true front then its error ratio is 0.5. If it finds one hundred solutions, 99 of which are very close to  $PF_{true}$  (and perhaps distributed evenly along it over a wide range in the objectives), and one (as before) which is in  $PF_{true}$ , then its error ratio will be 0.99. Clearly the second set of points is far better, but the first one scores a much lower error ratio.

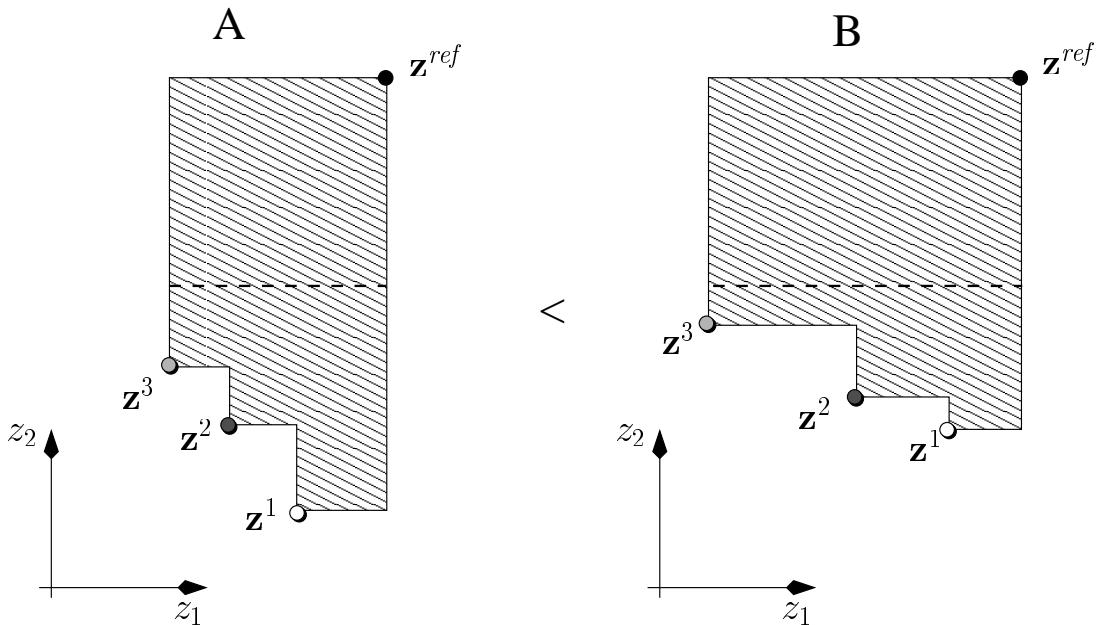
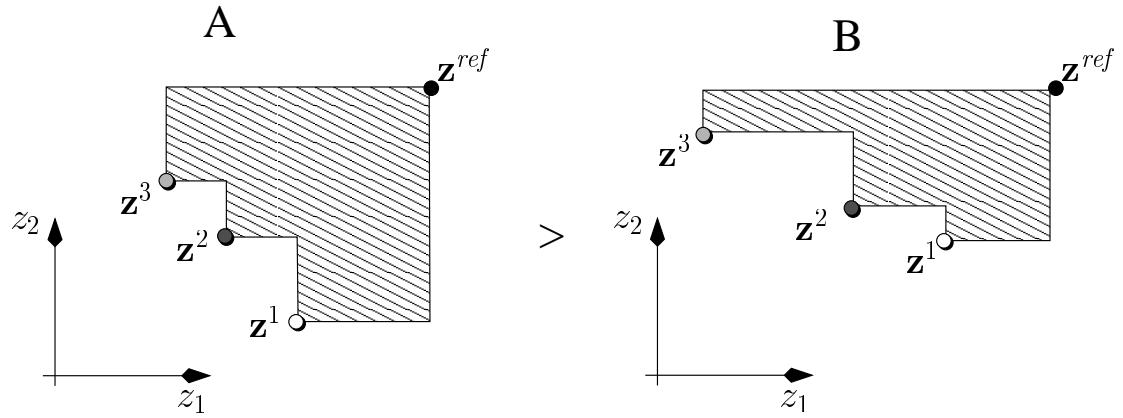


Figure 3.1: The relative value of the  $\mathcal{S}$  metric depends upon a rather arbitrary choice of where the reference point is chosen. In the upper half of the figure two nondominated point sets are shown,  $A$  and  $B$ . With the chosen reference point  $\mathcal{S}(A) > \mathcal{S}(B)$ . But in the lower half the same point sets have a different ordering in  $\mathcal{S}$ . The reference point has been moved to a larger value in objective 2 and a smaller value in objective 1, and consequently  $\mathcal{S}(A) < \mathcal{S}(B)$ .

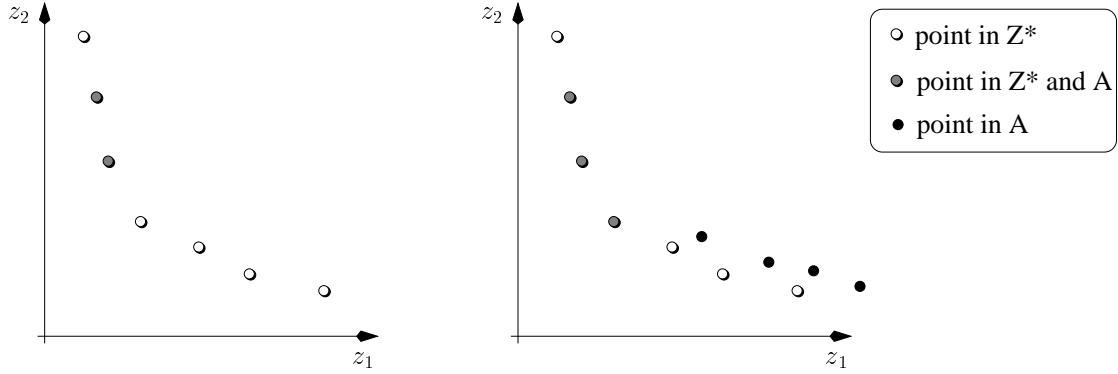


Figure 3.2: The nondominated set  $A$  on the left has an error ratio of zero. The set  $A$  on the right has an error ratio of  $4/7$ , but the set on the right is *clearly* better, independent of any preferences. This illustrates the non-monotonicity of the error ratio metric. Similarly, the set  $A$  on the left has a generational distance of zero, whereas the set  $A$  on the right has a generational distance  $> 0$ , which indicates that generational distance is not monotonic either.

It can perform almost in a completely contrary manner to the monotony property, because given a nondominated set  $A$  with one or more Pareto optimal points in it, then as more nondominated but non-Pareto optimal points are added, the error ratio goes up (which represents a decrease in quality according to this metric).

It does not exhibit relativity either because any subset of the Pareto front achieves the error ratio, 0, the best evaluation. However, it does exhibit weak relativity because the Pareto front itself is evaluated not worse than any other set.

### Advantages

It is easy to understand and easy to calculate. It is scaling independent. For test problems it can be used as a quick and rough means of assessing progress towards  $Z^*$ .

### Disadvantages and caveats

There is a requirement of knowing  $Z^*$ . It is incompatible with the outperformance relations. It cannot differentiate between different levels of complete outperformance, so long as neither has a point in  $Z^*$ .

### 3.11.4 Generational distance [Vel99]

$$\frac{\sqrt{\sum_{i=1}^n d_i^2}}{n} \quad (3.20)$$

where  $n$  is the number of vectors in the approximation set, and  $d_i$  is the distance in objective space between vector  $i$  and the *nearest* member of  $Z^*$ . Lower values of generational distance represent better nondominated sets.

#### Purpose and approach

The purpose of the metric is to measure the general or overall progress towards the true Pareto front. It is a reference metric using the true Pareto front as reference set. It induces a total ordering on the set of approximations. It is a non-cardinal measure.

#### Pareto compatibility

It is not weakly compatible with  $O_W$ . It is compatible with  $O_S$ . It does not exhibit the property of **weak monotony**. For example, according to this metric, it is better to have one vector close to the Pareto front, than to have the same vector and some other ones, in addition, as long as they are not closer on average to  $PF_{true}$  than the first one. This is clearly not true, in general. It does exhibit **weak relativity** because any subset of  $Z^*$  has the best evaluation.

#### Advantages

For a constant size of nondominated set, the metric's compatibility with  $O_S$  is satisfactory. It is relatively cheap to calculate.

#### Disadvantages and caveats

Because it is not compatible with  $O_W$  it cannot be used confidently for nondominated sets that are changing in cardinality. Even a population in an evolutionary algorithm will typically have one or more solutions in different dominance ranks (equivalence classes), therefore the cardinality of the rank one set is constantly changing. Therefore, the addition of a non-dominated point into the rank one set, within a constant population size, can *degrade* the

evaluation of the set, according to this metric. In addition, the metric cannot reliably differentiate between different levels of complete outperformance. Knowledge of  $Z^*$  is required for its use. The distance metric will either add or multiply different objectives together, introducing scaling and normalization issues that cannot be properly resolved without reference to additional preference information.

### 3.11.5 Maximum Pareto front error [Vel99]

$$\max_j (\min_i |f_1^i(\vec{x}) - f_1^j(\vec{x})|^p + |f_2^i(\vec{x}) - f_2^j(\vec{x})|^p)^{1/p} \quad (3.21)$$

where  $i = 1, \dots, n_1$  and  $j = 1, \dots, n_2$  index vectors in the approximation set and  $Z^*$  respectively, and  $p = 2$ . Lower values represent better nondominated sets.

#### Purpose and approach

This metric measures the largest distance between any vector in the approximation set and the corresponding closest vector in  $Z^*$ . It is a reference metric using the true Pareto front as a reference set. It induces a complete ordering on the set of approximations. It is a non-cardinal measure.

#### Pareto compatibility

It is not weakly compatible with any outperformance relation. It does not exhibit **weak monotony**. It is better, according to this metric, to find one solution close to the Pareto front than to find ten solutions, nine of which are Pareto optimal, and one which is some distance away from the Pareto front. This does not correspond to our intuitions about the quality of a nondominated set. It exhibits **weak relativity** because any subset of the true Pareto front has the best evaluation.

#### Advantages

It is cheap to compute. It provides information about whether any points found are far from the true front.

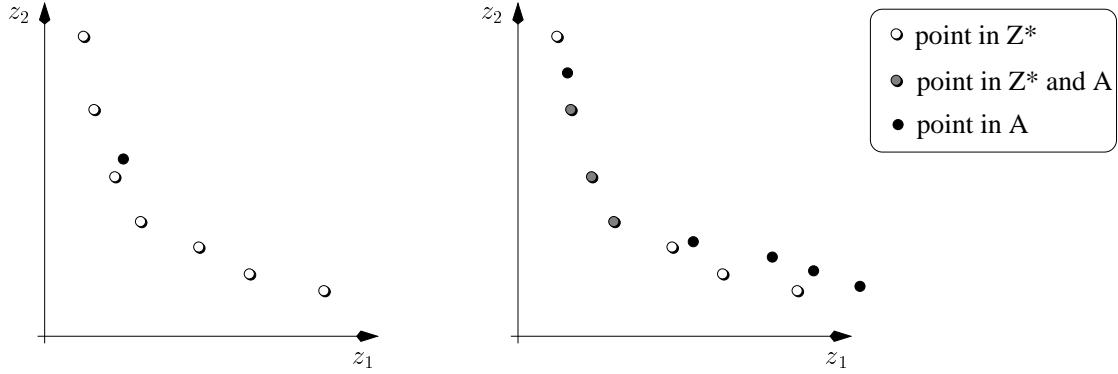


Figure 3.3: The nondominated set  $A$  on the left has a smaller Maximum Pareto Front Error than the set  $A$  on the right. But the set on the right completely outperforms the set on the left.

### Disadvantages and caveats

Even if a nondominated set has a very low Maximum Pareto Front Error it does not make it a good front. It certainly doesn't make it a better front than another one with a much larger error. Therefore much care is needed in using this measure. As with the other distance metrics, different objectives must be combined to get a single figure of merit. This will involve scaling and normalization issues. The true Pareto front must also be known.

#### 3.11.6 Overall nondominated vector generation (ONVG) [Vel99]

$$ONVG \triangleq |PF_{known}| \quad (3.22)$$

where  $PF_{known}$  is  $ND(A)$  for approximation set  $A$  in our notation.

### Purpose and approach

This metric measures the size of the approximation set. It is an independent metric, it induces a complete ordering on the set of approximations, and it is a cardinal measure.

### Pareto compatibility

It is not weakly compatible with any outperformance relation. It does not exhibit either weak monotony or weak relativity.

### Advantages

It is easy to compute. It is scaling independent. There are a few special pathological cases where this metric can be used to gauge the quality of a nondominated set, for example, if the entire search space contains only nondominated points.

### Disadvantages and caveats

See Pareto compatibility. In general, and on its own, this metric cannot be trusted to gauge anything which corresponds to measuring any aspect of nondominated set quality.

#### 3.11.7 Overall nondominated vector generation ratio (ONVGR) [Vel99]

$$ONVG \triangleq \frac{|PF_{known}|}{|PF_{true}|}. \quad (3.23)$$

### Purpose and approach

This metric measures the ratio of the size of the approximation set found to the number that actually exist in (the discretized) Pareto front. Larger ratios correspond to better evaluations. Notice that the ratio has no upper bound. According to Veldhuizen [Vel99] this metric: “gives some feeling for the number of nondominated vectors found versus how many exist to be found.” It is a reference metric using the true Pareto front as reference set. It induces a complete ordering on the set of approximations and it is a cardinal measure.

### Pareto compatibility

The metric is not weakly compatible with any outperformance relation. It does not exhibit weak monotony or weak relativity.

### Advantages

It is easy to compute. It is scaling independent.

### Disadvantages and caveats

See Pareto compatibility. The true (discretized) Pareto front is needed. The metric is useful in the pathological case when the whole search space is the Pareto front. Other than in this case, it is difficult to see how this measure corresponds to measuring any useful aspect of nondominated set quality.

#### 3.11.8 Other metrics proposed in [Vel99]

The Generational Nondominated Vector Generation (GNVG) and the Nondominated Vector Addition metrics specified in [Vel99] are based on the two metrics immediately above. Therefore they exhibit similar flaws: they are not Pareto compatible and do not exhibit either weak relativity or weak monotony.

#### 3.11.9 Spacing metric (Deb *et al.* [DAPM00a])

$$\Delta = \sum_{i=1}^{|PF_{known}|} \frac{|d_i - \bar{d}|}{|PF_{known}|} \quad (3.24)$$

where  $d_i$  is the Euclidean distance between two consecutive vectors in the nondominated front of the approximation set, and  $\bar{d}$  is the average of these distances.

### Purpose and approach

The purpose of this metric is to gauge how evenly the points in the approximation set are distributed in the objective space. It is an independent metric, it induces a complete ordering on the set of approximations, and it is a non-cardinal measure.

### Pareto compatibility

It is not weakly compatible with any outperformance relation. It does not exhibit weak monotony or weak relativity. It is quite possible that the true Pareto front has a non-uniform distribution of points.

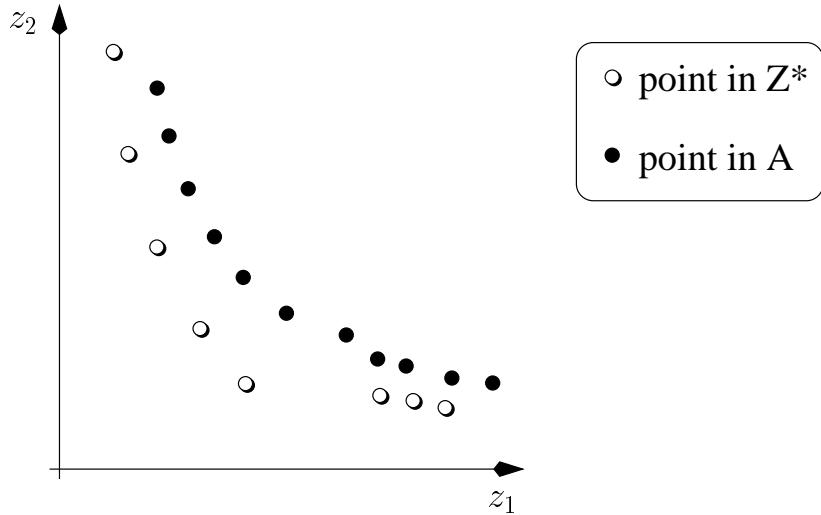


Figure 3.4: The true efficient set  $Z^*$  has a worse score on the spacing metrics of Schott and Deb *et al.* than the set  $A$ . This illustrates the non-relativity of these metrics.

### Advantages

If used in conjunction with other metrics (as it is designed to be), it may provide information about the distribution of vectors obtained. It has low computational overhead.

### Disadvantages and caveats

This metric is only suitable for two-dimensional objective spaces because it is not clear how ‘consecutive’ would be defined in the case of more than two objectives. It suffers from normalization and scaling issues, as with other metrics that combine objectives. In [DAPM00a], the boundary vectors in the true Pareto front are added into the approximation set, to ensure that the calculation takes into account the distribution of solutions in the entire region of the true PF. However, these may not be available. The metric’s incompatibility with the outperformance relations and the properties of monotony and relativity make it an unreliable means of making judgments about the overall quality of a nondominated set.

#### 3.11.10 Spacing metric (Schott [Sch95])

$$\sqrt{\frac{1}{n-1} \sum_{i+1} n(\bar{d} - d_i)^2} \quad (3.25)$$

where  $d_i = \min_j (|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|)$ ,  $i, j = 1..n$ ,  $\bar{d}$  is the mean of all  $d_i$  and  $n = |PF_{known}|$ .

### Purpose and approach

The purpose of this metric is to gauge how evenly the points in the approximation set are distributed in the objective space. It is an independent metric, it induces a complete ordering on the set of approximations, and it is a cardinal measure.

### Pareto compatibility

The metric is not even weakly compatible with  $O_W$ . It exhibits neither monotony nor relativity. This is the case because the true efficient set may not be well-spaced.

### Advantages

If used in conjunction with other metrics (as it is designed to be), it may provide information about the distribution of vectors obtained. It has low computational overhead. The metric can be generalized to more than two dimensions by extending the definition of  $d_i$  to cover more objectives.

### Disadvantages and caveats

The definition of  $d_i$  used at present does not specify the use of normalized distances, which may be problematic. The metric's incompatibility with the outperformance relations and the fact that it exhibits neither monotony nor relativity make it an unreliable means of making judgments about the overall quality of a nondominated set.

#### 3.11.11 The $\mathcal{C}$ metric

Let  $A, B \subseteq X$  be two sets of decision vectors. The function  $\mathcal{C}$  maps the ordered pair  $(A, B)$  to the interval  $[0, 1]$ :

$$\mathcal{C}(A, B) \triangleq \frac{|\{b \in B \mid \exists a \in A, a \preceq b\}|}{|B|}. \quad (3.26)$$

The value  $\mathcal{C}(A, B) = 1$  means that all decision vectors in  $B$  are weakly dominated by  $A$ . The opposite,  $\mathcal{C}(A, B) = 0$ , represents the situation when none of the points in  $B$  is weakly dominated by  $A$ . Note that always both orderings have to be considered, since  $\mathcal{C}(A, B)$  is not necessarily equal to  $1 - \mathcal{C}(B, A)$ .

### Purpose and approach

The purpose of this metric is to compare two nondominated sets for overall quality. It is a direct comparative approach giving a single figure of merit<sup>4</sup> that is not symmetric. It is a cardinal measure. It is difficult to establish whether the metric induces a complete ordering on the set of approximations because it is not clear how the pair of  $\mathcal{C}$  values should be interpreted together.

### Pareto compatibility

Since the  $\mathcal{C}$  metric gives two values when comparing sets  $A$  and  $B$ ,  $\mathcal{C}(A, B)$  and  $\mathcal{C}(B, A)$ , it is more difficult to analyze whether it is compatible with the outperformance relations. Its compatibility will depend on how we interpret or combine the two outputs of the metric.

Given two sets  $A$  and  $A'$  such that  $A \subset A'$  and  $\text{ND}(A') = A'$  i.e.  $A'$  is a mutually nondominated set, then  $\mathcal{C}(A, A') < 1$  and  $\mathcal{C}(A', A) = 1$  so the  $\mathcal{C}$  metric evaluates  $A'$  better than  $A$  if we take it that in general a set  $C$  is evaluated better than a set  $D$  according to the  $\mathcal{C}$  metric if  $\mathcal{C}(C, D) = 1$  and  $\mathcal{C}(D, C) < 1$ . Accepting this convention, the  $\mathcal{C}$  metric is compatible with the weak outperformance relation.

Of course, the  $\mathcal{C}$  metric can also be used with reference sets. Consider now the sets  $A$  and  $A'$  as just defined, and a set  $R$  where  $R$  is a reference set, with  $\text{ND}(R) = R$  and  $A' \subseteq R$ . Then,  $\mathcal{C}(R, A) = \mathcal{C}(R, A') = 1$  and  $\mathcal{C}(A, R) < \mathcal{C}(A', R)$ . Now for compatibility with  $O_W$  we wish  $A$  to be evaluated worse than  $A'$ , so we may simply make the convention for general sets  $C, D, R$  with  $D \subseteq R$ , that if  $\mathcal{C}(R, C) = \mathcal{C}(R, D) = 1$  and  $\mathcal{C}(C, R) < \mathcal{C}(D, R)$  then we say  $C$  is evaluated ‘worse’ than  $D$ . Then all sets  $C, D$  where  $D O_W C$  will be correctly evaluated via a reference set  $R$  provided  $D \subseteq R$ .

Consider sets  $A, A'$ , and  $R$  again, with  $A \subset A'$  as before, but this time  $A' \not\subseteq R$ . Consider the case where  $\mathcal{C}(R, A) = \mathcal{C}(R, A') = 1$ . Now, in this case it is possible that  $\mathcal{C}(A, R) = \mathcal{C}(A', R)$  even though  $A' O_W A$ . But we cannot have that  $\mathcal{C}(A, R) > \mathcal{C}(A', R)$  since  $A \subset A'$  so it is not possible that  $A$  can cover more of  $R$  than  $A'$ . Thus, for a reference set  $R$  and two sets

---

<sup>4</sup>Given an ordered pair of nondominated sets.

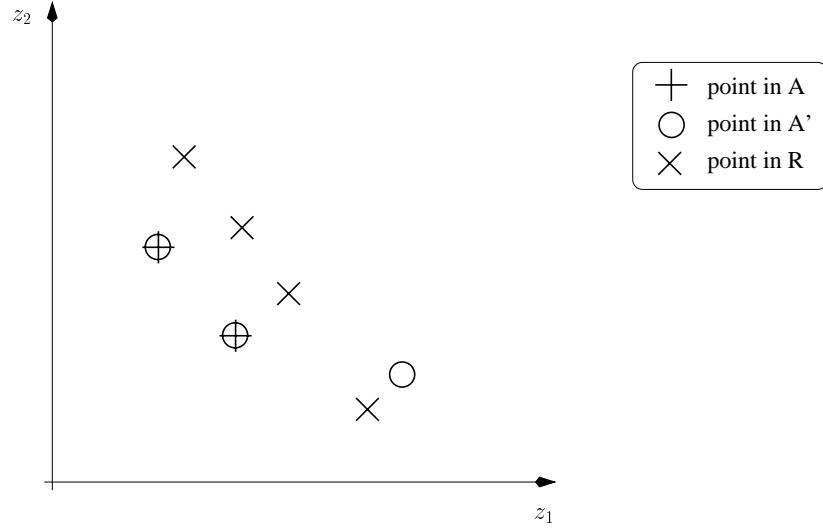


Figure 3.5:  $\mathcal{C}(R, A) = 0$ ,  $\mathcal{C}(R, A') = 1/3$ ,  $\mathcal{C}(A, R) = 3/4$ ,  $\mathcal{C}(A', R) = 3/4$  so against the reference set  $R$ ,  $A'$  seems to evaluate worse than  $A$  according to the  $\mathcal{C}$  metric. But  $A' \text{OW} A$  so we may conclude that the  $\mathcal{C}$  metric is not weakly compatible with the weak outperformance relation when used with a general reference set  $R$ . The alternative view is to say that  $A$  is not evaluated ‘worse’ than  $A'$  by the  $\mathcal{C}$  metric, arguing that we should adopt the convention that unless one of a  $\mathcal{C}$  metric pair gives output 1, the evaluation is meaningless. In this case, however, the  $\mathcal{C}$  metric has very little scope for application, and we would be better off by simply using the outperformance relations directly.

$A, A'$  with  $A' \text{OW} A$  such that  $\mathcal{C}(R, A) = \mathcal{C}(R, A') = 1$  then  $\mathcal{C}(A, R) \leq \mathcal{C}(A', R)$ , and so for this case  $\mathcal{C}$  is weakly compatible with  $\text{OW}$ .

But the  $\mathcal{C}$  metric cannot detect that  $A' \text{OW} A$  using a general reference set  $R$ , that is if it is not the case that  $\mathcal{C}(R, A) = \mathcal{C}(R, A') = 1$ .  $\mathcal{C}$  is not even weakly compatible with  $\text{OW}$  for such general  $R$ . See Figure 3.5 to see this.

Any pair of  $\mathcal{C}$  metric scores for a pair of sets  $A$  and  $B$  in which neither  $\mathcal{C}(A, B) = 1$  nor  $\mathcal{C}(B, A) = 1$ , indicates that the two sets are incomparable according to the weak outperformance relation. Drawing any further conclusions from the output of the  $\mathcal{C}$  metric in this case is inadvisable. For example, Figure 3.6 shows that if three sets are compared using the  $\mathcal{C}$  metric, they may not be ordered. In other words, the  $\mathcal{C}$  metric is cycle-inducing. Furthermore, the  $\mathcal{C}$  metric does not give an output which is even representative of our intuitions about the relative quality of two sets *unless* the two sets contain very evenly distributed points, and are of very similar cardinality. Figure 3.7 illustrates what happens when either of these constraints are contravened.

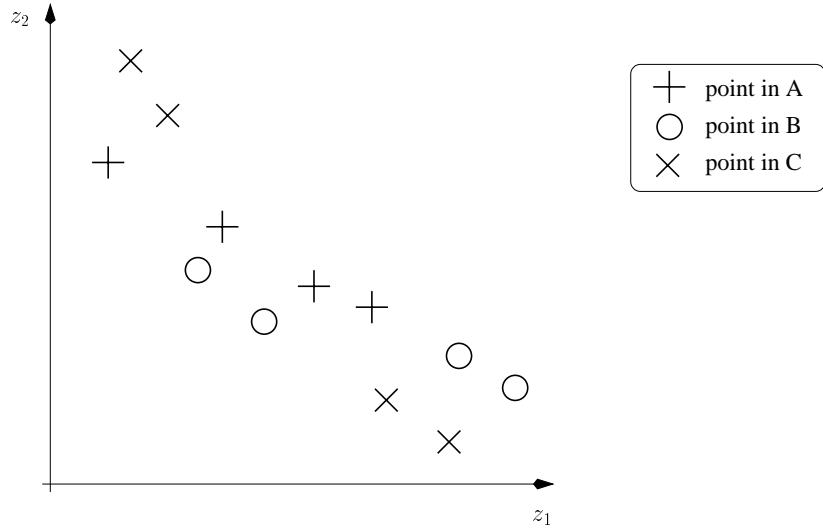


Figure 3.6: Cycling in the  $\mathcal{C}$  metric.  $\mathcal{C}(A, B) = 0$ ,  $\mathcal{C}(B, A) = 3/4$ ,  $\mathcal{C}(B, C) = 0$ ,  $\mathcal{C}(C, B) = 1/2$  and  $\mathcal{C}(A, C) = 1/2$ ,  $\mathcal{C}(C, A) = 0$  so, the  $\mathcal{C}$  metric evaluates B better than A, C better than B, but A better than C.

### Advantages

It has low computational overhead compared to the  $\mathcal{S}$  metric. It is compatible with  $O_S$ . It is scale and reference point independent. It does not require any knowledge of the efficient set or ranges of the feasible set. For two evenly-distributed sets, of the same cardinality, the  $\mathcal{C}$  metric gives results compatible with intuitive notions of quality, to some extent.

### Disadvantages and caveats

Its incompatibility with  $O_W$ . If two sets are of different cardinality and/or the distributions of the sets are non-uniform, then the  $\mathcal{C}$  metric gives unreliable results. It cannot determine the degree of outperformance if one set completely outperforms the other. The purpose of the  $\mathcal{C}$  metric could be better served by simply using the outperformance relations themselves.

#### 3.11.12 $D1_R$ (Czyzak and Jaszkiewicz)

$$D1_R(A, \Lambda) = \frac{1}{|R|} \sum_{r \in R} \min_{\mathbf{z} \in A} \{d(\mathbf{r}, \mathbf{z})\} \quad (3.27)$$

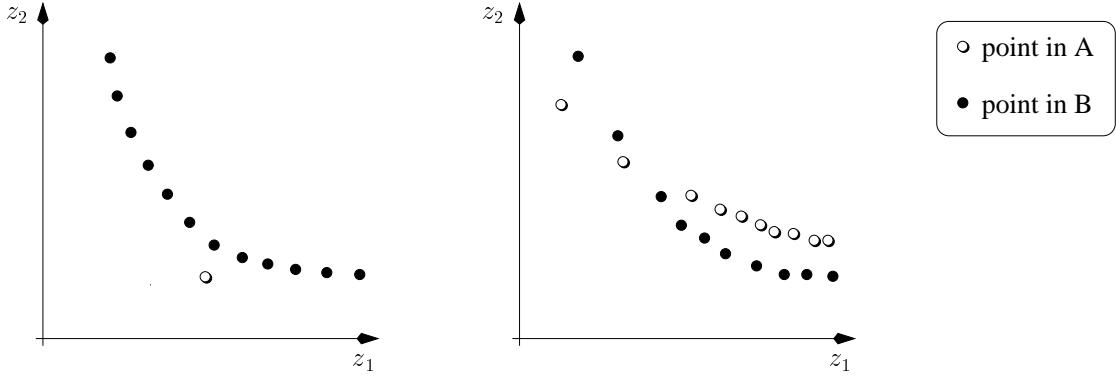


Figure 3.7: Illustrating some potential problems with the  $\mathcal{C}$  metric. On the left, the cardinality of  $A$  and  $B$  differ substantially. As a result  $\mathcal{C}(A, B) = 1/2$  and  $\mathcal{C}(B, A) = 0$ , and yet  $A$  dominates a far larger region of the objective space. On the right, the two sets have the same cardinality and neither seems significantly better than the other. However, because of the denser distribution of points with larger  $z_1$  values, in *both* sets, the  $\mathcal{C}$  metric judges  $B$  better than  $A$ :  $\mathcal{C}(A, B) = 1/10$  and  $\mathcal{C}(B, A) = 8/10$ .

where  $A$  is the approximation set,  $R$  is a reference set,  $d(\mathbf{r}, \mathbf{z}) = \max_k \{\lambda_k(r_k - z_k)\}$  and  $\Lambda = [\lambda_1, \lambda_2, \dots, \lambda_K]$ ,  $\lambda_k = 1/R_k$ ,  $k = 1..K$  with  $R_k$  being the range of objective  $k$  in set  $R$ .

### Purpose and approach

The purpose of  $D1_R$  is to measure the mean distance, over the points in a reference set, of the nearest point in an approximation set. The metric is a reference metric which induces a complete ordering on the set of approximations. It is a non-cardinal measure.

### Pareto compatibility

$D1_R$  is weakly compatible with  $O_W$ . However, it is not compatible even with  $O_C$ .

### Advantages

It is cheap to compute. Its weak compatibility with the outperformance relations. It *can* differentiate between different levels of complete outperformance but this will depend upon an appropriate choice of reference set.

### Disadvantages and caveats

The metric effectively calculates a weighted average where the reference points have equal weight. This means that the score is strongly dependent upon the distribution of points in the reference set.

#### 3.11.13 $R1$ and $R1_R$ (Hansen and Jaszkiewicz)

$$R1(A, B, U, p) = \int_{u \in U} C(A, B, u)p(u)du, \text{ where} \quad (3.28)$$

$$C(A, B, u) = \begin{cases} 1 & \text{if } u^*(A) > u^*(B) \\ 1/2 & \text{if } u^*(A) = u^*(B) \\ 0 & \text{if } u^*(A) < u^*(B) \end{cases} \quad (3.29)$$

where  $A$  and  $B$  are two approximation sets,  $U$  is some set of utility functions,  $u : \Re^K \rightarrow \Re$  which maps each point in the objective space into a measure of utility,  $p(u)$  is an intensity function expressing the probability density of the utility  $u \in U$ , and  $u^*(A) = \max_{\mathbf{z} \in A} \{u(\mathbf{z})\}$  and similarly for  $u^*(B)$ .

### Purpose and approach

$R1$  is based on calculating the probability that approximation  $A$  is better than  $B$  over an entire set of utility functions. It is a direct comparative metric. It does not induce a total ordering on the set of approximations. It is a non-cardinal measure.

$R1_R$  is  $R1$  when it is used with a reference set i.e. as a reference metric. This metric does then induce a total ordering on the set of approximations.

### Pareto compatibility

Making the convention that we are maximizing all objectives, a utility function  $u$  is strictly compatible with the dominance relation iff  $\forall \mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^1 > \mathbf{z}^2 \Rightarrow u(\mathbf{z}^1) > u(\mathbf{z}^2)$ . The set of all utility functions that are strictly compatible with the dominance relation is  $U_{sc}$ .

Let  $U(A > B) = \{u \in U \mid u^*(A) > u^*(B)\}$ . If the probability density function  $p(u)$  is such that the probability of selecting a utility function  $u \in U(A > B)$  is positive whenever  $U(A > B) \neq \emptyset$  and  $U \subseteq U_{sc}$  then  $R1$  is compatible with  $O_W$ .

Under the same conditions,  $R1_R$  is only weakly compatible with  $O_W$  and is not compatible even with  $O_C$ .

### Advantages

The metrics are scaling independent. They have a lower computational overhead than the  $\mathcal{S}$  metric. They are compatible with the outperformance relations. The  $R1_R$  metric can differentiate between different levels of complete outperformance provided that an appropriate reference set is chosen.

### Disadvantages and caveats

The  $R1$  metric cannot differentiate between different levels of complete outperformance. It is cycle-inducing. The metrics depend upon being able to define a set of utility functions. In general, this can be achieved without any knowledge of the Pareto front or the search space, however.

### $R2$ and $R2_R$ (Hansen and Jaszkiewicz)

$$\begin{aligned}
R2(A, B, U, p) &= E(u^*(A)) - E(u^*(B)) \\
&= \int_{u \in U} u^*(A)p(u)du - \int_{u \in U} u^*(B)p(u)du \\
&= \int_{u \in U} (u^*(A) - u^*(B))p(u)du
\end{aligned} \tag{3.30}$$

where  $A$  and  $B$  are two approximation sets,  $U$  is some set of utility functions,  $u : \Re^K \rightarrow \Re$  which maps each point in the objective space into a measure of utility,  $p(u)$  is an intensity function expressing the probability density of the utility  $u \in U$ , and  $u^*(A) = \max_{\mathbf{z} \in A} \{u(\mathbf{z})\}$  and similarly for  $u^*(B)$ .

### Purpose and approach

Where  $R1$  just uses the function  $C(A, B, u)$  to decide which of two approximations is better on utility function  $u$ , without measuring by *how much*,  $R2$  takes into account the expected values of the utility.  $R2$  calculates the expected *difference* in the utility of an approximation  $A$  with another one  $B$ . It is a direct comparative metric. It induces a complete ranking in the

set of all approximations. It is a non-cardinal measure.  $R2_R$  is  $R2$  when used as a reference metric. It also induces a complete ranking in the set of all approximations.

#### Pareto compatibility

$R2$  is compatible with  $O_W$  subject to the same set of conditions on the set of utility functions used as outlined for  $R1$ .  $R2_R$  is also compatible with  $O_W$  given this set of conditions.

#### Advantages

The advantages of  $R2$  arise from its compatibility with all of the outperformance relations and the fact that it can differentiate between different levels of complete outperformance.

#### Disadvantages and caveats

The application of  $R2$  depends upon the assumption that it is meaningful to add the values of different utility functions from the set  $U$ . This simply means that each utility function in  $U$  must be appropriately scaled with respect to the others and its relative importance.

#### 3.11.14 $R3$ and $R3_R$ (Hansen and Jaszkiewicz)

Hansen and Jaszkiewicz also propose a similar metric to  $R2$  whereby the *ratio* of the best utility values is calculated instead of the differences. These metrics are called  $R3$  and  $R3_R$ . The latter is similar to the approach used in single objective optimization, where an approximate solution is evaluated by the ratio of its value to that of a fixed bound, note Hansen and Jaszkiewicz.

#### 3.11.15 Summary of analysis

From the analysis above, we would recommend the use of the  $R1$ ,  $R2$ , and  $R3$  metrics of Hansen and Jaszkiewicz, and the  $\mathcal{S}$  metric of Zitzler. The other metrics may not be as useful because they suffer from poor compatibility with the outperformance relations and cannot differentiate between different levels of complete outperformance.

|          | $O_W$ compatible | $O_S$ compatible | $O_C$ compatible | non-cycle inducing | reference independent | scaling independent | cardinal/non-cardinal | differentiates $O_C$ levels | computational overhead                 |
|----------|------------------|------------------|------------------|--------------------|-----------------------|---------------------|-----------------------|-----------------------------|--|
| $\times$ | ✓✓               | ✓✓               | -                | ✓                  | ✓                     | c                   | ✗                     | low                         | $C$ metric (Zitzler)                   |
| ✓✓       | ✓✓               | ✓✓               | ✓                | ✗                  | ✓                     | n/c                 | ✓                     | high                        | $S$ metric (Zitzler)                   |
| ✗        | ✓✓               | ✓✓               | ✓                | ✗                  | ✗                     | n/c                 | ✗                     | med                         | Generational distance (Veldhuizen)     |
| ✗        | ✗                | ✗                | ✓                | ✗                  | ✓                     | c                   | ✗                     | low                         | Error ratio (Veldhuizen)               |
| ✗        | ✗                | ✗                | ✓                | ✓                  | ✗                     | n/c                 | ✗                     | med                         | Max Pareto front error (Veldhuizen)    |
| ✗        | ✗                | ✗                | ✓                | ✓                  | ✓                     | c                   | ✗                     | low                         | ONVG (Veldhuizen)                      |
| ✗        | ✗                | ✗                | ✓                | ✗                  | ✓                     | c                   | ✗                     | low                         | ONVGR (Veldhuizen)                     |
| ✗        | ✗                | ✗                | ✓                | ✓(✗)               | ✗                     | n/c                 | ✗                     | med                         | Spacing Metric (Deb <i>et al.</i> )    |
| ✗        | ✗                | ✗                | ✓                | ✓                  | ✗                     | n/c                 | ✗                     | med                         | Spacing Metric (Schott <i>et al.</i> ) |
| ✓        | ✓                | ✓                | ✓                | ✗                  | ✗                     | n/c                 | ✓                     | med                         | $D1_R$ (Czyzak and Jaszkiewicz)        |
| ✓✓       | ✓✓               | ✓✓               | ✗                | ✓(✗)               | ✓                     | n/c                 | ✗                     | med                         | $R1$ (Hansen and Jaszkiewicz)          |
| ✓        | ✓                | ✓                | ✓                | ✗                  | ✓                     | n/c                 | ✓                     | med                         | $R1_R$ (Hansen and Jaszkiewicz)        |
| ✓✓       | ✓✓               | ✓✓               | ✓                | ✓(✗)               | ✗                     | n/c                 | ✓                     | med                         | $R2$ (Hansen and Jaszkiewicz)          |
| ✓✓       | ✓✓               | ✓✓               | ✓                | ✗                  | ✗                     | n/c                 | ✓                     | med                         | $R2_R$ (Hansen and Jaszkiewicz)        |

Table 3.2: A summary of the analysis of the 14 different metrics. In the first three columns, a single tick denotes weak compatibility and a double tick denotes compatibility with the outperformance relation. The ‘reference independent’ column indicates if the metric is dependent on a reference set or any reference point(s) in objective space. Deb’s spacing metric is dependent on knowledge of  $Z^*$  iff the extreme points are added to the approximation set. The  $R1$  and  $R2$  measures may depend on knowledge of an ideal point if Tchebycheff utility functions are used. The ‘scaling independent’ column indicates if the ordering of approximations could be affected by scaling of the objectives. The ‘differentiates  $O_C$  levels’ column indicates if, given three approximation sets where  $A \leq_C B \leq_C C$ , the metric gives a larger difference in evaluation when comparing  $A$  and  $C$  than  $A$  and  $B$ . If it always can (given an appropriate reference set, if necessary) then it receives a tick. Note that Generational distance can but does not always differentiate between  $O_C$  levels.

## 3.12 Assessment methods used in this thesis

The previous section analysed some of the current available methods of evaluating or comparing *individual* approximation sets. However, to present information about the performance of a stochastic optimizer one should report results from multiple runs, and preferably estimate the statistical significance of the results, as well. In this thesis, several methods for measuring the performance of a stochastic optimizer are used. These methods are based on Fonseca and Fleming's work on attainment surfaces [FF96] and on the  $\mathcal{S}$  metric of Zitzler. These methods are described next.

### 3.12.1 Methods based on attainment surface sampling

A set of nondominated points in objective space define a region that is dominated by them. The boundary of this region is a 'surface' called the 'attainment surface' [FF96]. By measuring the location and extent of this attainment surface one can judge how good the approximation to the true Pareto front is, in a way that is consistent with our intuitive notions of quality.

Figure 3.8 shows 5 objective vectors in a 2-objective minimization problem. Superimposed on these points is a boundary, which Fonseca and Fleming state is the family of tightest goals that are known to be attainable. This boundary is called the attainment surface and can be used to replace the individual points found, as it is exactly equivalent to them, under the assumption of Pareto optimization. Note that the surface preserves all of the information about both the quality and the distribution of the points found, since it lies in exactly the same place as the points, at the points themselves, but then moves away from them at right angles (and the underlying true Pareto front) in between them. Thus this surface, on its own, gives us a clearer representation of the quality of the approximation found than the points themselves, because it reminds us not to interpolate (by eye) between the points found, but emphasizes where the 'holes' in the approximation set are.

Fonseca and Fleming note that when several runs of an optimizer are performed it is possible to overlay the attainment surfaces from each independent run. This overlaying of surfaces gives a far clearer picture of the different runs than overlaying the points found themselves. Importantly, the combination of the surfaces define a sample worst boundary and a sample best boundary that can easily be identified. In fact, the individual surfaces could be erased, leaving only the upper and lower boundary attainment surfaces. This gives a very clear indication of the range of quality of the approximation of an algorithm. In fact, these surfaces from two or more optimizers could be overlaid on the same graph for comparison. However,

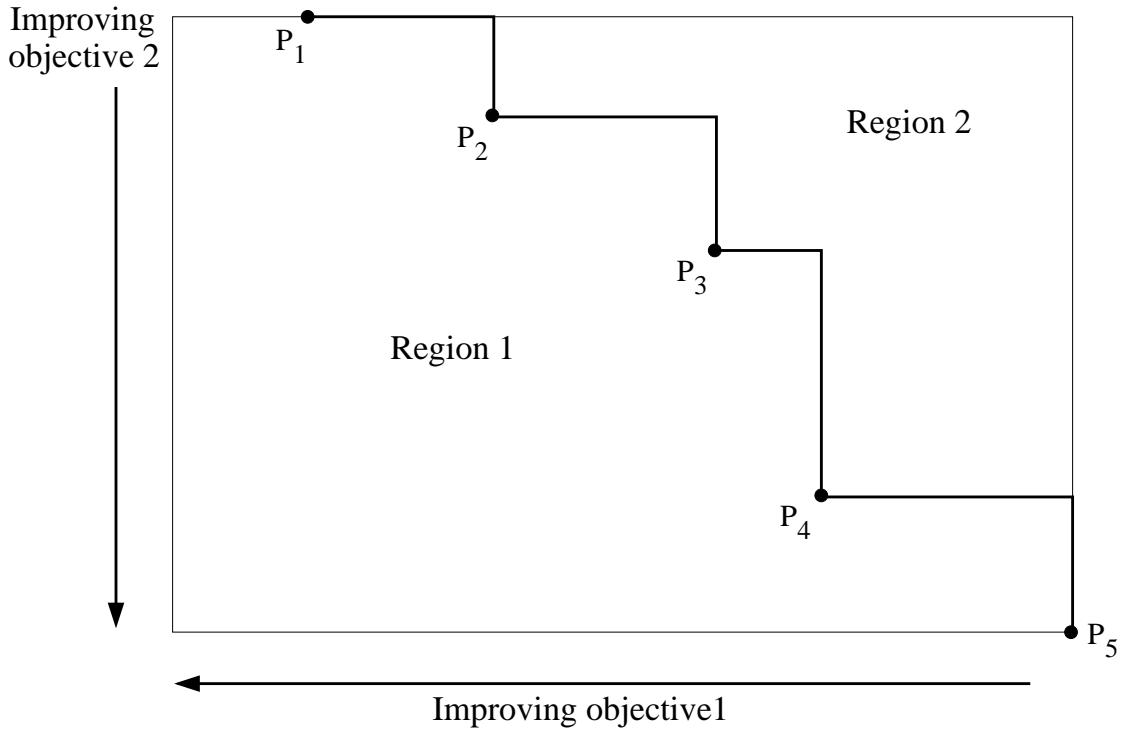


Figure 3.8: How a set of nondominated points divide the objective space to form an attainment surface.

although giving the best and worst that an optimizer attains over some sample number of runs is indeed useful, and is certainly far more information-rich than plotting the vectors from just one run for each optimizer, it would be desirable if one could calculate a ‘typical’ attainment surface, or a representative range in which the attainment surface is expected to lie in some proportion of the runs. Even, better would be the ability to make use of the whole distribution of runs to make some statistical inferences about them. Fortunately, these goals can be achieved, to a certain resolution, by sampling the surfaces using lines angled in the direction of increasing value in all objectives.

Figure 3.9 shows two attainment surfaces and a number of auxiliary angled lines running in the direction of improvement in all (both) objectives. The intersections of the attainment surfaces with an auxiliary line form a univariate distribution from which the median, quartile, and any percentile intersection point could be calculated. With a number of distributed auxiliary lines these can be used to define a median, quartile or any percentile attainment surface up to the resolution defined by the number of auxiliary lines used. Furthermore, if the sets of attainment surfaces come from two different optimizers it is possible to use the univariate distribution of intersection points along the auxiliary lines to perform a statistical test to decide if one distribution is statistically different (to the left of) the other. Fonseca

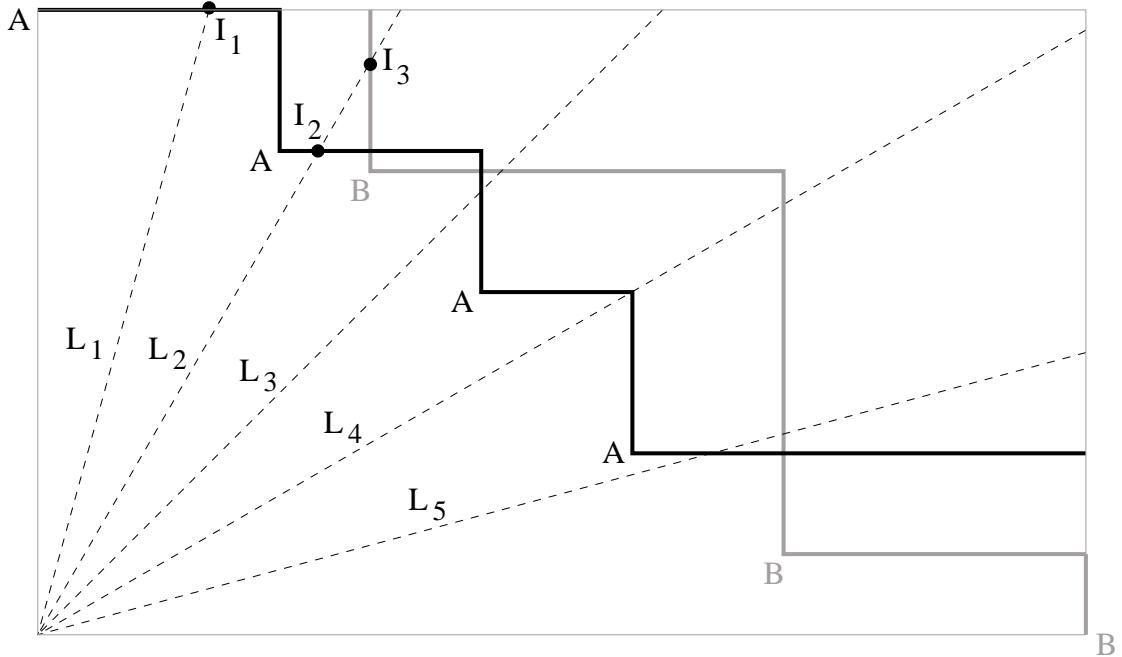


Figure 3.9: Sampling the attainment surface using lines of intersection.

and Fleming recommend using a standard non-parametric test for this.

The article [FF96] also indicates several weaknesses of the approach, and directions for future research. One weakness is that the calculated percentile attainment surfaces do not properly represent the probability of attaining the whole surface in the given percentage of runs; each point on the surface just independently represents the level achieved in the given percentage of runs. This may be a serious weakness if an algorithm were to generate alternately one extreme or other of the Pareto front in different runs, but never all extremes at once. This may not be distinguishable from an algorithm that on some runs does find the **WHOLE** extent of the Pareto front, and on others it does not. Visualization tools and generalizations of the techniques to more than two objectives are also mentioned as areas for future development. Fonseca and Fleming do not consider in the paper that the technique could be used with reference surfaces to give quantitative, statistical measures of performance for an individual algorithm.

The following four methods based on attainment surface sampling are used in this thesis:

**AS1: Plotting percentile or median attainment surfaces** Given a set  $D$  of  $n$  collections of approximation sets from  $n$  different optimizers, normalize the points in  $D$ :  

$$z_k \leftarrow (z_k - (\min_k(\{z_k \mid \mathbf{z} \in D\})) / ((\max_k(\{z_k \mid \mathbf{z} \in D\}) - (\min_k(\{z_k \mid \mathbf{z} \in D\}))) \text{ for all } k \in 1..K.$$

For each collection:

For each of  $s$  evenly distributed sample lines:

Compute the co-ordinates of the intersection of the sample line with attainment surface of each approximation set in the current collection.

From these intersections, compute the co-ordinates of the intersection with the median or  $p$ th percentile attainment surface. Plot this intersection point.

Join together the points plotted to form the percentile or median attainment surface of the current collection.

**AS2: Mann-Whitney two distributions test** Given sets  $C_A$  and  $C_B$  of nondominated sets  $A_j$  and  $B_k$ ,  $j \in 1..n$ ,  $k \in 1..m$  from  $n$  runs of optimizer  $A$  and  $m$  runs of optimizer  $B$ , normalize all points as above, and compute all of the intersections of the attainment surfaces from  $C_A \cup C_B$  with each of  $s$  evenly distributed sample lines in the normalized space, angled in the direction of increasing value in all of the objectives. Taking each sample line in turn perform the Mann-Whitney rank-sum test at confidence level  $\alpha$ , that the intersections from  $C_A$  represent a better distribution than those from  $C_B$ . The outcome of this test can be that  $C_A$  beats  $C_B$ , that  $C_B$  beats  $C_A$ , or that neither is statistically significantly better than the other. Compute the percentage of lines on which  $C_A$  beats  $C_B$ , and the percentage of lines on which  $C_B$  beats  $C_A$ . Return these values.

**AS3: Mann-Whitney multiple distributions test** Normalization is carried out over the union of all the sets. Then each pair of distributions is taken in turn, and for each such pair the Mann-Whitney rank-sum test is applied as in AS2. For each sampling line, the results of all comparisons are stored. From inspection of the results stored at each sampling line, the percentage of sample lines on which a particular set beat *all* other sets is computed, for each set. Similarly, the percentage of the sampling lines on which a particular set was unbeaten by *any* other set is computed, for each set. Return these values, i.e. for 4 sets of nondominated sets the results might be:

|             | A  | B  | C  | D  |
|-------------|----|----|----|----|
| % beats all | 10 | 2  | 6  | 0  |
| % unbeaten  | 23 | 35 | 10 | 15 |

**AS4: Sample probability of achieving an entire reference surface in a single run**

We have a set  $C$  of nondominated sets generated from runs of an optimizer and a reference set  $R$ . First, normalize the space with respect to the  $R$  only. Compute the attainment surfaces of each of the nondominated sets in the set  $C$ . Compute the number of members of  $C$ ,  $n_{achieve}$ , that lie closer to the origin than  $R$  on *every* sample line. The sample probability is then  $n_{achieve}/n$ . The population probability can then be computed using the binomial distribution and a given confidence level.

### 3.12.2 Methods based on the $\mathcal{S}$ Metric

Given a point  $\mathbf{z}^1 = \{z_1^1, z_2^1, \dots, z_K^1\}$  in objective space  $Z$ , and a reference vector  $\mathbf{z}^{ref} = \{z_1^{ref}, z_2^{ref}, \dots, z_K^{ref}\}$  dominated by  $\mathbf{z}^1$ , let the region dominated by  $\mathbf{z}^1$  and bounded by  $\mathbf{z}^{ref}$  be defined as the set:

$$R(\mathbf{z}^1, \mathbf{z}^{ref}) \triangleq \{\mathbf{y} \mid \mathbf{y} < \mathbf{z}^{ref} \text{ and } \mathbf{z}^1 < \mathbf{y}, \mathbf{y} \in \mathbb{R}^K\}. \quad (3.31)$$

For a nondominated set  $A$  of vectors  $\mathbf{z}^i, i = 1..|A|$ , and a reference vector  $\mathbf{z}^{ref}$ , that is dominated by all members of  $A$ , the region dominated by  $A$  and bounded by  $\mathbf{z}^{ref}$  is defined as the set:

$$R(A, \mathbf{z}^{ref}) \triangleq \bigcup_{i \in 1..|A|} R(\mathbf{z}^i, \mathbf{z}^{ref}). \quad (3.32)$$

The  $\mathcal{S}$  metric of  $A$  with respect to the reference vector is the ‘hyperarea’, or Lebesgue integral of the set  $R(A, \mathbf{z}^{ref})$ . In a minimization (maximization) problem, the reference point  $\mathbf{z}^{ref}$  is taken by Zitzler to be the vector whose components are the maximum (minimum) value in each objective. This gives a nonnegative measure for all possible nondominated sets in the feasible objective space. However, note that the choice of this reference vector is fairly arbitrary, and could really be any vector outside the feasible objective space. But different choices will affect the relative  $\mathcal{S}$  value of two different nondominated sets. Although the choice of reference point does affect the ordering of nondominated sets, scaling the objectives does not, i.e., it does not matter if the magnitudes of different objectives are very different.

The  $\mathcal{S}$  metric of a nondominated set  $A$  may be computed by recursively projecting the set of points into fewer dimensions and calculating the Lebesgue integral of these.

To calculate the  $\mathcal{S}$  metric in just two dimensions (objectives) the points are sorted in decreas-

---

```
size_of_set( $A, \mathbf{z}^{ref}, k$ )
```

---

$\mathbf{z}^{high}$  is the vector with the largest value in objective  $k$  from amongst the vectors in the updated set  $A$

$NDk(A, k)$  returns the nondominated vectors from the set  $A$  with respect to the lowest  $k$  objectives only.

---

```

 $\mathcal{S} \leftarrow 0$ 
 $z_k^{prev} \leftarrow z_k^{ref}$ 
while( $A \neq \emptyset$ )
     $A \leftarrow NDk(A, k - 1)$ 
    if( $k < 3$ )
         $\mathcal{S}_{k-1} \leftarrow z_1^{high}$ 
    else
         $\mathcal{S}_{k-1} \leftarrow size\_of\_set(A, \mathbf{z}^{ref}, k - 1)$ 
     $\mathcal{S} \leftarrow \mathcal{S} + \mathcal{S}_{k-1} \cdot |z_1^{high} - z_1^{prev}|$ 
     $z_1^{prev} \leftarrow z_1^{high}$ 
     $A \leftarrow A \setminus \{\mathbf{z}^r \mid z_1^r \geq z_1^{high}, \mathbf{z}^r \in A\}$ 
return  $\mathcal{S}$ 

```

---

Figure 3.10: Recursively computing the  $\mathcal{S}$  metric of a nondominated set  $A$ , in  $k$  objectives. The function has  $O(n^{k+1})$  complexity in general, where  $n$  is the number of points in the set  $A$ .

ing order of objective 1 values, and then the following expression is evaluated:

$$\sum_{i \in 1..|A|} |z_1^i - z_1^{ref}| \cdot |z_2^i - z_2^{i-1}| \quad (3.33)$$

where  $z_2^0$  is initially set to  $z_2^{ref}$ . In higher dimensions this generalizes to the recursive function:  $size\_of\_set(A, \mathbf{z}^{ref}, k)$ , shown in Figure 3.12.2. The function has a time complexity of  $O(n^{k+1})$  for general  $k$ : the  $NDk()$  function is  $O(n^2)$  and this must be performed up to  $n$  times in a call to  $size\_of\_set(A, \mathbf{z}^{ref}, k)$ , not counting further recursive calls; and  $size\_of\_set(A, \mathbf{z}^{ref}, k - 1)$  is called  $n$  times at level  $k$ , for  $k \geq 3$ . An illustration of the calculation of the  $\mathcal{S}$  for three points in three dimensions is given in Figure 3.11.

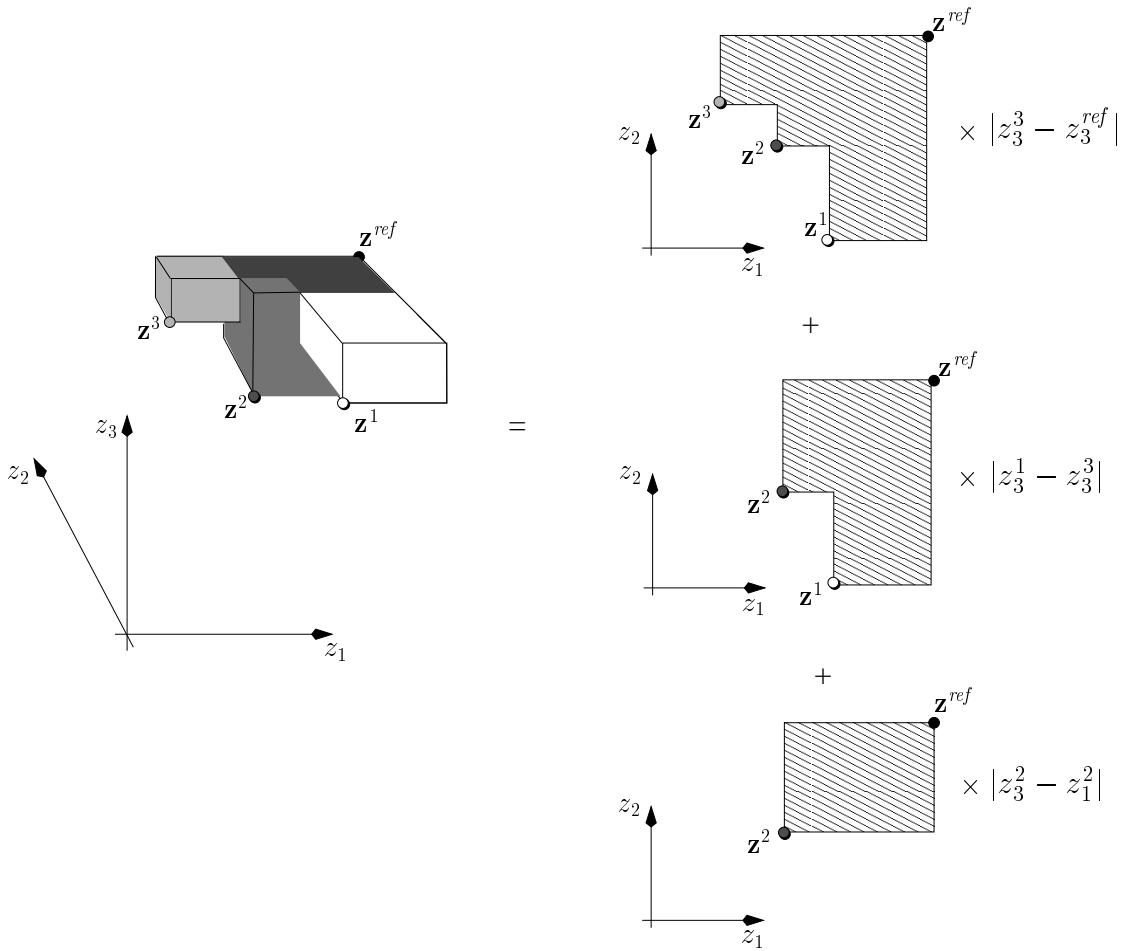


Figure 3.11: How the  $\mathcal{S}$  metric is calculated in multiple dimensions. Three nondominated points in a 3-objective space are shown (left). The regions they weakly dominate, and which weakly dominate the reference point, are shown by the shaded cuboids. The combined size of these regions is shown by the calculation on the right.

For assessment of an algorithm over multiple independent runs, we propose the following metrics based on the  $\mathcal{S}$  metric.

**S1: Total discovered region  $\mathcal{S}_\tau$**  Given a set  $C_A$  of nondominated sets  $A_j, j \in 1..n$  from  $n$  independent runs of an algorithm the size of the total discovered region is:

$$\mathcal{S}_\tau \triangleq \mathcal{S}\left(\bigcup_{j=1}^n A_j\right). \quad (3.34)$$

This gives an idea of the variability between sets found by an optimizer in different independent runs. If the mean value of the  $\mathcal{S}$  metric for the  $n$  runs is much smaller than the total discovered region over the  $n$  runs, then the optimizer discovers different parts of the objective region in different runs. This may or may not be a good thing, depending on the application.

**S2: Percentile (median) attainment region  $\mathcal{S}_p$  ( $\mathcal{S}_\mu$ )** Given a set  $C_A$  of nondominated sets  $A_j, j \in 1..n$  from  $n$  independent runs of an algorithm the size of the  $p$ th percentile attainment region is:

$$\mathcal{S}_p \triangleq \mathcal{S}(AS_p(C_A)) \quad (3.35)$$

where  $AS_p(C_A)$  is the  $p$ th percentile attainment surface of the set  $C_A$ , calculated using **AS1**, defined above. The median attainment region has the same form, but replacing  $AS_p(C_A)$  by  $AS_{median}(C_A)$ . Notice that  $\mathcal{S}_\mu$  is different to the median value of the  $\mathcal{S}$  metric, and similarly  $\mathcal{S}_p$  is different from the  $p$ th percentile value of the  $\mathcal{S}$  metric. The attainment regions give the *total* size of the region attained by some percentage of the runs, whereas the percentile value of the  $\mathcal{S}$  metric gives the size of the region of a *particular* run. The median and interquartile attainment regions give an idea of the centre and spread of the regions attained over a number of runs.

**S3: Median value of the coverage difference** The coverage difference of two nondominated sets  $A$  and  $B$  is defined as:

$$\mathcal{S}_{A \setminus B} \triangleq \mathcal{S}(A \cup B) - \mathcal{S}(B) \quad (3.36)$$

and gives the size of the region weakly dominated by  $A$  but not weakly dominated by  $B$ . The median value of the coverage difference concerns multiple nondominated sets. For sets  $C_A$  and  $C_B$  of nondominated sets  $A_j$  and  $B_j$ ,  $j \in 1..n$ , from  $n$  different runs of two optimizers, the median value of the coverage difference is the median of  $\mathcal{S}_{A_j \setminus B_j}$  for  $j \in 1..n$ . For large  $n$ , this approximates the population median coverage difference, that is the coverage difference achieved on at least 50% of runs. The median value of the

complementary metric  $S_{B \setminus A}$  is also computed. Taking the two median values together allows one to infer whether one algorithm's sets *completely outperform* the other's, on 50% or more of runs. Other statistical tests of the distribution of the coverage difference could also be used.

## Chapter 4

# The Pareto Archived Evolution Strategy (PAES)

### 4.1 Requirements for a Pareto hillclimber

In Section 3.7 of the last chapter, the benefits of developing a ‘Pareto hillclimber’ were discussed. The necessary characteristics of such an algorithm may be summarized using the following specification:

1. Uses single-point local search only;
2. Uses Pareto ranking/dominance selection;
3. Returns a limited set of diverse, mutually nondominated solutions from a single algorithm run;
4. Possesses few parameters;
5. Uses less computational overhead than population-based Pareto EAs.

Unfortunately, in practice, there are several conflicts between these requirements. The third item, in particular, is problematic with respect to some of the others: it conflicts with item 1 because a single-point local searcher cannot usually return multiple solutions. This means that some form of extra store of solutions must be used. However, ensuring this store returns a *diverse* set of solutions is difficult whilst respecting requirements 4 and 5, and the necessity of *nondominance* of these solutions is also difficult to achieve efficiently.

The most troublesome conflict between the identified requirements, however, is that between items 1 and 2. In a single-point local searcher, an essential feature is the rule-set governing the conditions under which the current solution is discarded in favour of accepting the candidate ‘mutant’ solution — the acceptance function. Should the dominance relation be used in the acceptance function, many mutant/current solution pairs would be incomparable, and this difficulty would increase with the number of objectives. It would thus be critical what acceptance rule was employed in the case of incomparable current and mutant solutions, and due to the requirement 2, Pareto dominance should be the primary means of choosing between the two solutions.

Let us examine further the problem of using Pareto dominance in the acceptance function. Consider the acceptance rule:

$$\mathbf{x} \leftarrow \mathbf{x}' \text{ iff } \mathbf{x} \not\prec \mathbf{x}' \quad (4.1)$$

where  $\mathbf{x}$  is the current solution (decision) vector and  $\mathbf{x}'$  is the candidate or mutant solution. This rule, where the mutant is accepted whenever it dominates or is incomparable with the current solution, is not *negative efficiency preserving* [Han99]. This means that the entire region dominated by a current solution is not necessarily preserved by future current solutions. For this reason, over a series of ‘moves’ (applications of the rule), the region dominated can reduce in all objectives. In other words, if  $\mathbf{x}(t_n)$  is the current solution at iteration  $t_n$ , then the following relation  $\mathbf{x}(t_n) \prec \mathbf{x}(t_m)$  can hold for a solution  $\mathbf{x}(t_m)$ ,  $t_m > t_n + 1$ . An example

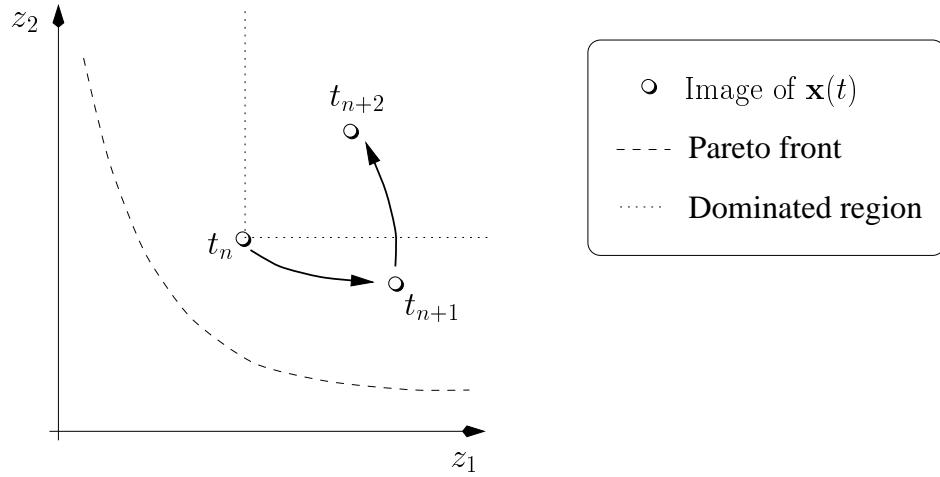


Figure 4.1: An illustration of a not negative efficiency preserving acceptance rule. At iteration  $t_n$ , the image of  $\mathbf{x}(t_n)$  is close to the Pareto front. But after two subsequent iterations the image of  $\mathbf{x}$  has moved into the previous dominated region, further away from the Pareto front, and worse in both objectives.

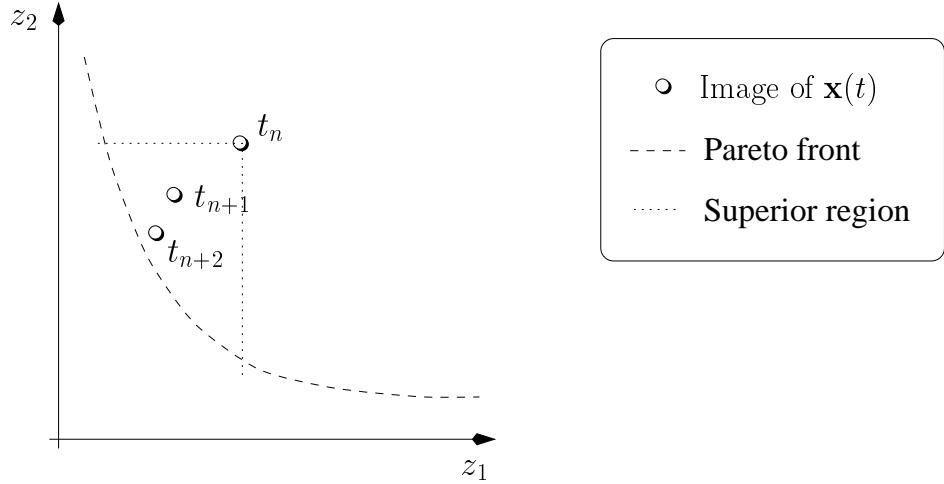


Figure 4.2: An illustration of an efficiency preserving acceptance rule. Only a restricted region of the Pareto front can be reached from a point in the objective space. The image of  $\mathbf{x}(t_n)$  is far away from the Pareto front and progresses towards it in subsequent iterations, but all points outside of the superior region of  $\mathbf{x}(t_n)$  are unreachable. As the image of  $\mathbf{x}$  progresses, less and less of the Pareto front is reachable until ultimately it will become ‘stuck’ at one point on the Pareto front.

of this is given in Figure 4.1. This is obviously a severe problem because there will be little selection pressure towards the Pareto front, and no guarantee that the quality of solutions will not degrade over time.

The complementary acceptance rule:

$$\mathbf{x} \leftarrow \mathbf{x}' \text{ iff } \mathbf{x}' \prec \mathbf{x} \quad (4.2)$$

only accepts the mutant solution if it dominates the current solution. This rule is the one used by Rudolph [Rud98b, RA00] in his convergence proofs for multiobjective evolution strategies, and it is *efficiency preserving* [Han99]. This means that all future current solutions will lie in the feasible region dominating the current solution — the superior region of the current solution. This means that from a given initial point in the feasible region, it may be that only a fraction of the Pareto front is reachable. This is illustrated in Figure 4.2. Furthermore, once a Pareto optimal solution is found, it is impossible to find any further solutions on the Pareto front. Therefore, even if this method were used in conjunction with storing solutions offline, it could only ever find one nondominated solution per algorithm run. This severely limits its viability as a replacement strategy for use in a single point searcher, and obviously conflicts with the third requirement in our list.

---

**Algorithm:** Archiving Hillclimber

---

*Data:*

$M_t$  is the set of best (minimal) evaluated solution vectors discovered

$\mathbf{x}$  is the current solution vector

$\mathbf{x}'$  is the mutant solution vector

*Functions:*

$\text{Init}()$  returns a solution vector  $\mathbf{x} \in X$

$\text{Mutate}(\mathbf{x})$  returns a neighbour of  $\mathbf{x}$

$\text{Inferior}(\mathbf{x}, M_{t-1})$  returns TRUE if  $\mathbf{x}$  is inferior to  $M_{t-1}$

$\text{Reduce}(\{\mathbf{x}'\} \cup M_{t-1})$  reduces the number of elements in the set  $(\{\mathbf{x}'\} \cup M_{t-1})$  e.g. by discarding elements that are no longer minimal

---

```
 $t \leftarrow 0$                                 /* Initialization */
 $M_t \leftarrow \emptyset$ 
 $\mathbf{x} \in X \leftarrow \text{Init}()$ 
 $t \leftarrow t + 1$ 
 $M_t \leftarrow M_{t-1} \cup \{\mathbf{x}\}$ 
while ( $\text{Terminate}(t, M_t) \neq \text{TRUE}$ )           /* Main Loop */
     $t \leftarrow t + 1$ 
     $\mathbf{x}' \in X \leftarrow \text{Mutate}(\mathbf{x})$ 
    if ( $\text{Inferior}(\mathbf{x}', M_{t-1}) \neq \text{TRUE}$ ) {
         $\mathbf{x} \leftarrow \mathbf{x}'$ 
         $M_t \leftarrow \text{Reduce}(\{\mathbf{x}'\} \cup M_{t-1})$ 
    }
return  $M(t_{final})$                                 /* Termination */
```

---

Figure 4.3: Pseudocode for a generic archiving hillclimber, upon which the (1+1)-PAES algorithm is based.

## 4.2 An archiving hillclimber

In the  $(1+1)$ -PAES algorithm developed in this chapter, some of the potential problems with a Pareto hillclimber, particularly how dominance might be used in the acceptance function, are overcome by basing it on an archiving hillclimber. Pseudocode for an archiving hillclimber algorithm is given in Figure 4.3. Notice that the algorithm is similar to the random mutation hillclimber (RMHC), described in [MHF94] and used extensively as a baseline search algorithm in the literature. However, notice that in RMHC, the mutant solution is compared only against the current solution, and accepted if it is at least as good. In contrast, the archiving hillclimber algorithm stores, in a set — or archive — (some of) the solution vectors that have the minimal evaluation on the objective function up to iteration  $t$ . At each iteration, the mutant solution  $\mathbf{x}'$  is compared against the elements in this archive, and accepted only if it is found not to be ‘inferior’ to them. The addition of the archive gives the algorithm the potential to find *all* the optimal solutions to a problem, rather than just one. Importantly, the possession of an archive, also allows this algorithm to be adapted for Pareto multiobjective optimization, where the goal is to find (an approximation to) the set of minimal elements — the Pareto set — in the feasible solution space.

Clearly, the generic archiving hillclimber is not a fully specified search algorithm because it depends upon how its component functions are specified. The functions `Init()` and `Mutate( $\mathbf{x}$ )` have well-understood meanings and do not need further elucidation. In any case, these functions can be tailored to particular problems. For a realizable description of a particular instance of archiving hillclimber, then, only a definition of the two functions `Inferior( $\mathbf{x}'$ ,  $M_{t-1}$ )` and `Reduce( $\{\mathbf{x}'\} \cup M_{t-1}$ )` must be added. A very simple Pareto hillclimber can be specified thus:

```

Inferior( $\mathbf{x}'$ ,  $M_{t-1}$ ):
  if  $\neg(M_{t-1} \prec \mathbf{x}')$ 
    return TRUE
  else
    return FALSE

Reduce( $\{\mathbf{x}'\} \cup M_{t-1}$ ):
  return  $(\{\mathbf{x}'\} \cup M_{t-1}) \setminus \{\mathbf{x} \in M_{t-1} \mid \mathbf{x}' \prec \mathbf{x}\}$ 

```

This Pareto hillclimber addresses two of the issues identified earlier with respect to using a hillclimber for Pareto optimization: It returns a set of mutually nondominated solutions

found during the run; and it overcomes the problems associated with comparing a mutant solution only with the current solution, by comparing the mutant with the whole archive, at each iteration. This algorithm returns the set of *all* nondominated solutions it has found. Notice that, in one sense, the archive is being used like the population in a Pareto EA — a set approximating the current nondominated front that is used to rank or estimate the quality of newly generated solutions. However, the Pareto hillclimber remains a true hillclimber because it only uses a single point local search, with a  $(1 + 1)$  selection strategy. Thus some of the conflicts between requirements 1 and 2 in our list, have been resolved.

Although the Pareto hillclimber does fulfil the role of a Pareto analogue of a standard hillclimber, it is cursed by two potential problems because its archive is unbounded. First, its memory usage is unbounded which is a potential problem for any practical implementation of it. Second, the computational overhead of checking whether the mutant solution is inferior to the current Pareto front, can get progressively larger over time, as the number of non-dominated solutions stored in the archive may increase without bound. To solve both these problems we can change the  $\text{Reduce}(\{\mathbf{x}'\} \cup M_{t-1})$  function so as to limit the capacity of the archive. To do this, extra rules must be added to the  $\text{Reduce}(\{\mathbf{x}'\} \cup M_{t-1})$  function which will ideally ensure that requirements 3, 4, and 5 of our specification are satisfied.

In the next section, we evaluate a number of alternative `Reduce` functions for controlling the contents of the archive. These different archiving strategies progress, in four stages, from the simple `Reduce` function proposed above, through to a strategy of low computational cost which both ensures a bounded archive and promotes solution diversity. The convergence properties of the four strategies are analyzed in detail, and their computational overhead is also considered.

### 4.3 Evaluation of archiving strategies

In the following we analyse the convergence properties of four different archiving strategies. Specifically, we analyse whether the archive converges and whether it converges to the Pareto front, a subset of the Pareto front, or some other set with specific attributes. To prove these convergence properties we employ a model in which at every iteration of the algorithm, a generating process generates a point which must be archived. The proofs rely on the fact that at every time step the generating process draws the point at random from a probability distribution in which each point in the search space has a non-zero probability. In practice, this assumption will be true whenever, for example, a mutation is applied to every bit in a binary string with some small probability; the standard method of generating a new point in

a random mutation hillclimber.

The first archiving strategy we consider is the *Reduce* function already proposed, in which there is no bound on the archive. We prove that, using this strategy, the archive converges to the Pareto front, provided that the process for generating points accords with our assumptions, and the search space is finite.

Using the same model, assumptions and methods of proof, we then consider strategies that update a bounded archive, where the capacity of the archive could be less than the cardinality of the Pareto front. The first of the bounded archiving strategies accepts any nondominated vector until the archive is full, and thereafter accepts only vectors which dominate members of the archive, and removes the dominated vectors. We prove that this simple strategy guarantees that the archive converges to a subset of the Pareto front. However, this archiving strategy is efficiency preserving [Han99] which means that some regions of the Pareto front may not be reachable (see Figure 4.2). In addition, this strategy does not have any mechanism for encouraging the discovery of a diverse and evenly distributed set of nondominated points in the objective space.

To avoid these problems with the efficiency preserving strategy, another, more complicated, archiving strategy is proposed. For full archives, this strategy accepts a vector if it: (a) dominates any member of the archive; or (b) if it is nondominated with respect to (all members of) the archive and its addition would increase the net value of the  $\mathcal{S}$  metric [Zit99] of the archive, when one (selected) member of the archive is removed to allow its entry. A proof that this strategy guarantees convergence to a subset of the Pareto front for bounded archives is given. This result can also be generalized to any quality metric  $\mathcal{Q}$  which has the property that if a set  $Z_1$  weakly outperforms another set  $Z_2$  then  $\mathcal{Q}(Z_1) > \mathcal{Q}(Z_2)$ , preventing cycling. Using the  $\mathcal{S}$  metric also guarantees that the converged set is a local optimum of  $\mathcal{S}$ . This means that there is no point which would result in a net increase in the  $\mathcal{S}$  metric of the archive if it were added and another point were removed. This guarantee is important because any set which is a local optimum of  $\mathcal{S}$  seems to be ‘well-distributed’ (see Figure 4.4) although we offer no formal proof that this is so.

The  $\mathcal{S}$  metric archiving may work well, but its computational overhead would be prohibitively high. The fourth and final archiving strategy, *adaptive grid archiving*, we propose is computationally much less expensive, and also has the further advantage that it does not require the setting of any boundaries in objective space or any other use of reference points. In fact, it only requires the value of one non-critical parameter to be specified. On the other hand, adaptive grid archiving does not have the cycle-preventing property of the  $\mathcal{S}$  metric, so it is not possible to prove that the archive converges, so it is not possible to show that the archive

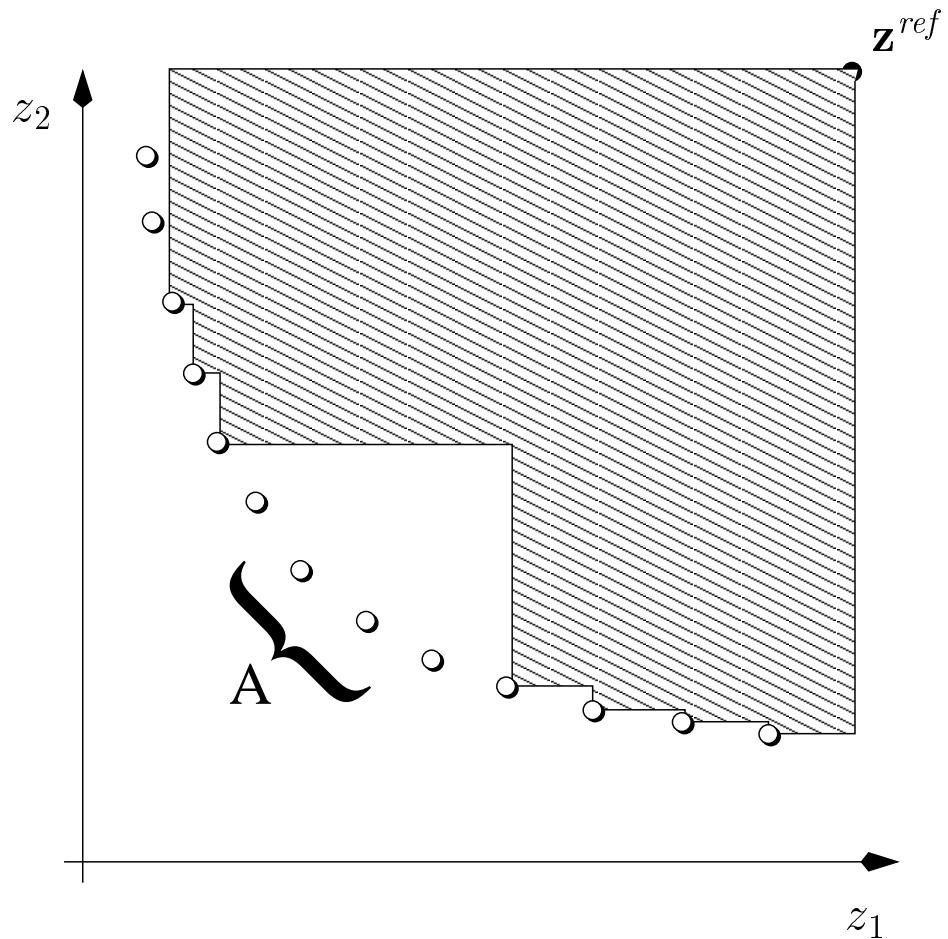


Figure 4.4: The figure shows the true Pareto front (white points) of a minimization problem. The hatched area shows the region dominated by seven of the Pareto optimal points. These points represent the (full) archive of an algorithm using  $\mathcal{S}$  metric archiving. Clearly, these points in the archive are not as well-distributed as they could be. Fortunately, this seems to imply that they are not a local optimum of  $\mathcal{S}$ . A net increase in  $\mathcal{S}$  would result if any vector pointed at by the label A were to replace any other vector in the current archive.

converges to a subset of the Pareto front. However, several other convergence results can be derived, which show that the adaptive grid archiving strategy will maintain a well-distributed and diverse set of vectors.

#### 4.3.1 Definitions of terms used in the analysis

**Definition 4.1** *Let there be a finite set  $Z$ , which is the objective space of all feasible objective vectors  $\mathbf{z}$ .*

**Definition 4.2** *Let there be a set called the Pareto front,  $Z^* = \{\mathbf{z}^* \in Z \mid \forall \mathbf{z} \in Z, \mathbf{z} < \mathbf{z}^*\}$ .*

**Definition 4.3** *Let there be an archive  $M_t \subseteq Z$  of objective vectors.*

**Definition 4.4** *Let there be a generating process  $\text{Gen}(t)$  for generating solutions from  $Z$  to be stored in  $M_t$ . The output of  $\text{Gen}(t)$  at time  $t$  is  $\mathbf{z}_t$ , and associated with  $\text{Gen}(t)$  there is a probability  $pr(t, \mathbf{z})$  of generating solution  $\mathbf{z} \in Z$  at time  $t$ .  $\text{Gen}(t)$  has the property that  $\forall t, \forall \mathbf{z} \in Z, pr(t, \mathbf{z}) > 0$ .*

The convergence proofs for the different archiving strategies that we shall consider, are formulated using the generic archiving algorithm  $AA_{\text{Reduce} \in RED}$ , shown in Figure 4.5. A particular archiving algorithm may be specified from  $AA$  by specifying a particular `Reduce` function from the set of all such functions,  $RED$ .

The generic  $AA_{\text{Reduce} \in RED}$  algorithm, like the archiving hillclimber presented in Figure 4.3, generates one point per iteration and then updates the nondominated solutions archive as necessary. However, the archiving algorithm is far more general, as it does not specify *how* the points are generated — only that there is a generating process that can generate any point in the search space with non-zero probability. The use of the  $AA_{\text{Reduce} \in RED}$  algorithm in the following proofs allows us to make general statements about archiving strategies without concerning ourselves with the details of a particular method of generating points in the search space.

**Definition 4.5** *We say that  $M_t$  is converged under algorithm  $AA_{\text{Reduce} \in RED}$  if it is in a state in which its members will not change for all future iterations, i.e., if  $\forall t > t_n, M_t = M_{t_n}$  then  $M_{t_n}$  is converged. We also use the phrase: “ $AA_{\text{Reduce} \in RED}$  converges” under rule `Reduce`, as shorthand for saying that the archive of  $AA_{\text{Reduce} \in RED}$  converges.*

---

**Algorithm:**  $AA_{\text{Reduce} \in RED}$

---

$M_t$  is the nondominated vectors archive  
 $\text{Gen}(t)$  is a generating function with positive generation probability for all feasible vectors  
 $\mathbf{z}_t$  is the objective vector generated at time  $t$

---

```

 $t \leftarrow 0$ 
 $M_t \leftarrow \emptyset$ 
while(1)
{
     $t \leftarrow t + 1$ 
     $\mathbf{z}_t \leftarrow \text{Gen}(t)$           /* Generate new vector */
     $\text{Reduce}(\{\mathbf{z}_t\} \cup M_{t-1})$   /* Call each archiving rule */
}

```

---

Figure 4.5: Generic Archiving Algorithm  $AA_{\text{Reduce} \in RED}$ , where  $RED$  is the set of all Reduce functions.

We also recall the definitions, 3.12, 3.13, and 3.15, from Chapter 3, which define, respectively, the  $\text{ND}$  function, the meaning of the relations  $<$  and  $\sim$  between an objective vector and a nondominated vector set, and the weak outperformance relation between a pair of nondominated vector sets.

### 4.3.2 Archiving strategy 1: unbounded archive

Using an unbounded archive, the  $\text{Reduce}()$  function defined in Section 4.2 can be rewritten in the form of the objective vector as:

```

Reduce( $\{\mathbf{z}_t\} \cup M_{t-1}$ ):
    return  $\text{ND}(\{\mathbf{z}_t\} \cup M_{t-1})$ 

```

which defines algorithm  $AA_{\text{unbounded}}$ .

By inspection, it is clear that the archive is always a nondominated set. From this fact, several properties of  $AA_{\text{unbounded}}$  can now be proved.

**Lemma 4.1** *If a nondominated set  $Z_a \in Z = \text{ND}(Z_a)$  and  $Z_a \not\subseteq Z^*$  then  $\exists \mathbf{z}^* \in Z^*$  such that  $\mathbf{z}^* < Z_a$ . In other words, a nondominated set that is not a subset of the Pareto optimal set always contains vectors that are dominated by at least one vector in the Pareto optimal set.*

**Proof 4.1**

$$\begin{aligned} & Z_a \not\subseteq Z^* \\ \text{so } & \exists \mathbf{z} \in Z_a, \mathbf{z} \notin Z^* \quad \text{by definition of } \subseteq \\ \text{so } & \exists \mathbf{z}^* \in Z^*, \mathbf{z}^* < \mathbf{z} \quad \text{by definition of } Z^* \end{aligned}$$

**Lemma 4.2**  $M_t$  is converged implies that  $M_t$  is the Pareto front,  $Z^*$ .

**Proof 4.2** Assume at some time  $t = t_i$ ,  $M_{t_i}$  is converged and  $M_{t_i} \neq Z^*$ . From  $M_{t_i} \neq Z^*$ , we have two possibilities:  $M_{t_i} \subset Z^*$  or  $M_{t_i} \not\subseteq Z^*$ . The former implies that there exists an efficient vector which is nondominated with respect to the archive, i.e.  $\exists \mathbf{z}^* \in Z^*$  such that  $\mathbf{z}^* \sim M_{t_i}$ . The latter implies that there exists at least one efficient vector which dominates with respect to the archive, i.e.  $\exists \mathbf{z}^* \in Z^*$  such that  $\mathbf{z}^* < M_{t_i}$  (from Lemma 4.1). So we have,  $\exists \mathbf{z}^* \in Z^*$  such that  $\mathbf{z}^* \sim M_{t_i} \vee \mathbf{z}^* < M_{t_i}$ .

We wish to show that at some future time  $t_j > t_i$ ,  $M_{t_j} \neq M_{t_i}$ . We may choose the time  $t = t_j$  when  $\text{Gen}(t)$  generates the vector  $\mathbf{z}^*$  defined above. This is sure to occur since all points are generated with a positive probability. Now if  $M_{t_j-1} \neq M_{t_i}$  then  $M_t$  was not converged, a contradiction to our assumption. Otherwise, according to the function  $\text{Reduce}(\{\mathbf{z}_t\} \cup M_{t-1})$ ,  $\mathbf{z}^*$  will replace those vectors in  $M_{t_j-1}$  that it dominates. This also contradicts our original assumption that the set  $M_{t_i}$  is converged, and we must conclude that vector sets that are not subsets of  $Z^*$  are not converged, under  $AA_{\text{unbounded}}$ .

**Lemma 4.3**  $\forall i \in \mathbb{N} ((M_{t+i} = M_t) \vee (M_{t+i} \text{ OW } M_t))$ . In other words, the contents of the archive monotonically improve over time: given any initial archive, all future archives are either identical to, or weakly outperform, the initial archive.

**Proof 4.3** By induction:

Let  $i = 1$ .  $M_{t+1}$  arises from  $M_t$  after one application of  $AA_{\text{unbounded}}$ . Either a new vector  $\mathbf{z}$  has been added which is nondominated with respect to  $M_t$ , that is  $M_{t+1} = M_t \cup \{\mathbf{z}\}$ , or if  $\mathbf{z} < M_t$  it has been accepted, throwing others out of the archive, or nothing is added to the archive. So we have that  $(M_{t+1} = M_t) \vee (M_{t+1} \text{ OW } M_t)$ .

Assume true for  $k \in \mathbb{N}$  so  $(M_{t+k} = M_t) \vee (M_{t+k} \text{ OW } M_t)$

Is it true for  $k + 1$ ?

Given the proof for  $i = 1$ , then there are four possibilities:

$(M_{t+k} = M_t) \wedge (M_{t+k+1} = M_{t+k})$  in which case  $M_{t+k+1} = M_t$   
 $(M_{t+k} = M_t) \wedge (M_{t+k+1} \text{ OW } M_{t+k})$  in which case  $M_{t+k+1} \text{ OW } M_t$   
 $(M_{t+k} \text{ OW } M_t) \wedge (M_{t+k+1} = M_{t+k})$  in which case  $M_{t+k+1} \text{ OW } M_t$   
 $(M_{t+k} \text{ OW } M_t) \wedge (M_{t+k+1} \text{ OW } M_{t+k})$  in which case  $M_{t+k+1} \text{ OW } M_t$   
 by transitivity of OW [HJ98]. So we have now shown both  $(M_{t+1} = M_t) \vee (M_{t+1} \text{ OW } M_t)$  and  $(M_{t+k} = M_t) \vee (M_{t+k} \text{ OW } M_t) \Rightarrow (M_{t+k+1} = M_t) \vee (M_{t+k+1} \text{ OW } M_t)$ . The proof of Lemma 4.3 follows by induction.

**Lemma 4.4**  $M_{t_n} \neq M_{t_m}$  for  $t_n > t_m$  implies  $\forall t > t_n, M_t \neq M_{t_m}$ .

**Proof 4.4** Assume  $M_{t_n} \neq M_{t_m}$  for  $t_n > t_m$  and  $\exists t > t_n$  such that  $M_t = M_{t_m}$ . From lemma 4.3 we have  $M_{t_n} \text{ OW } M_{t_m}$  and  $(M_t = M_{t_n}) \vee (M_t \text{ OW } M_{t_n})$ . If the former then  $(M_t \text{ OW } M_{t_m})$  and if the latter  $(M_t \text{ OW } M_{t_m})$ . Therefore, either way we contradict  $M_t = M_{t_m}$ .

**Lemma 4.5**  $M_t$  converges under algorithm  $AA_{\text{unbounded}}$ .

**Proof 4.5** Assume  $M_t$  never converges i.e.  $\forall t_i \exists t_j, t_j > t_i, M_{t_j} \neq M_{t_i}$ . This implies that there are an infinite number of different sets  $M_t$ , since from lemma 4.4 we know that if  $M_{t_j} \neq M_{t_i}$  for  $t_j > t_i$ , then  $\forall k \in \mathbb{N}, M_{t_j+k} \neq M_{t_i}$ . However, since the set  $Z$  is finite, so is  $2^Z$ . But for all  $t$ ,  $M_t \in 2^Z$  so there are finite different  $M_t$ .

**Corollary 4.1** A corollary of lemma 4.5 and lemma 4.2 is that the algorithm  $AA_{\text{unbounded}}$  converges to the Pareto front.

### 4.3.3 Archiving strategy 2: for a bounded archive, accept only dominating vectors when archive is full

Now we consider a simple Reduce function for maintaining a bounded archive, similar to the archiving strategy analysed in [RA00]. Let the set  $M_t$  have a maximum capacity  $arcsize$  such that  $\forall t, |M_t| \leq arcsize$ . The archiving algorithm  $AA_{\text{dom}}$  is defined by the Reduce function:

Reduce( $\{\mathbf{z}_t\} \cup M_{t-1}$ ):

    Fill( $t$ );

    Domination( $t$ );

```

Steady_state( $t$ );
return( $M_t$ )

```

and the rules it calls:

Fill( $t$ ):

```

if (  $|M_{t-1}| < arcsize$  ) then  $M_t \leftarrow \text{ND}(M_{t-1} \cup \{\mathbf{z}_t\})$ 

```

Domination( $t$ ):

```

if (  $|M_{t-1}| = arcsize \wedge \mathbf{z}_t < M_{t-1}$  ) then  $M_t \leftarrow \text{ND}(M_{t-1} \cup \{\mathbf{z}_t\})$ 

```

Steady\_state( $t$ ):

```

if (  $|M_{t-1}| = arcsize \wedge (\mathbf{z}_t \not\leq M_{t-1})$  ) then  $M_t \leftarrow M_{t-1}$ 

```

Define that a rule is executed if and only if its conditional evaluates true. From inspection of the three archiving rules called by Reduce, it is evident that at each time step one and only one conditional must be true. Thus, one and only one rule will be executed at each time step. From this it follows that at each time step:

- Either a new vector is added which dominates some members of the archive, and those dominated members are removed;
- or a new vector which is nondominated with respect to the archive is added to the archive, increasing its cardinality by one;
- or nothing is added to the archive because the new vector is dominated by the archive or the archive has already reached its capacity.

From this it follows that the archive is always a nondominated set. Using this fact, we can prove some convergence properties of  $AA_{dom}$ .

**Lemma 4.6**  $M_t$  is converged under algorithm  $AA_{dom}$  implies  $M_t$  is a subset of  $Z^*$ .

**Proof 4.6** Assume at some time  $t = t_i$ ,  $M_{t_i}$  is converged and  $M_{t_i} \not\subseteq Z^*$ . The latter implies that there exists at least one efficient vector  $\mathbf{z}^* \in Z^*$ ,  $\mathbf{z}^* < M_{t_i}$  (from lemma 4.1) because the archive is always a mutually nondominated set. We wish to show that at some future time  $t_j > t_i$ ,  $M_{t_j} \neq M_{t_i}$ .

We may choose the time  $t = t_j$  when  $\text{Gen}(t)$  generates the vector  $\mathbf{z}^*$  which dominates the archive. Now if  $M_{t_j-1} \neq M_{t_i}$  then  $M_t$  was not converged, a contradiction to our assumption.

Otherwise, by archiving rule Domination( $t$ ),  $\mathbf{z}^*$  will replace those vectors in  $M_{t_j-1}$  that it dominates. This also contradicts our original assumption that the set  $M_{t_i}$  is converged, and we must conclude that vector sets that are not subsets of  $Z^*$  are not converged, under  $AA_{dom}$ .

**Lemma 4.7**  $M_{t_n} \neq M_{t_m}$  for  $t_n > t_m$  implies  $\forall t > t_n (M_t \neq M_{t_m})$ .

**Proof 4.7** Lemma 4.3 still holds when  $AA_{unbounded}$  is replaced by  $AA_{dom}$ . To see this, just consider that whenever the archive is not full,  $AA_{dom}$  reduces to  $AA_{unbounded}$  and whenever it is full it will only accept vectors that dominate the archive. From this it is clear that lemma 4.3 still holds. From this, lemma 4.7 immediately follows in exactly the same way as lemma 4.4.

**Lemma 4.8**  $M_t$  converges under algorithm  $AA_{dom}$

**Proof 4.8** Assume  $M_t$  never converges i.e.  $\forall t_i, \exists t_j, t_j > t_i, M_{t_j} \neq M_{t_i}$ . But this implies that there are an infinite number of different sets  $M_t$ , since from lemma 4.7 we know that if  $M_{t_j} \neq M_{t_i}$  for  $t_j > t_i$ , then  $\forall k \in \mathbb{N}, M_{t_j+k} \neq M_{t_i}$ . However, since the set  $Z$  is finite, so is  $2^Z$ . But for all  $t$ ,  $M_t \in 2^Z$  so there are finite different  $M_t$ .

**Theorem 4.1** The set  $M_t$  converges to a subset of  $Z^*$  under algorithm  $AA_{dom}$ .

**Proof 4.9** From lemma 4.6 we see that all converged sets are Pareto optimal subsets. From lemma 4.8 we see that  $M_t$  always converges.

This completes the proofs relating to algorithm  $AA_{dom}$ . We have shown that it is guaranteed to converge to a subset of the Pareto front. However, because it uses an efficiency preserving strategy certain areas of the Pareto front may not be reachable. In the next section, this algorithm is enhanced to overcome this problem.

#### 4.3.4 Archiving strategy 3: $\mathcal{S}$ metric archiving

Let the  $\mathcal{S}$  metric of the set  $M_t$  be denoted  $\mathcal{S}(M_t)$ . We note that the  $\mathcal{S}$  metric has the following properties necessary for the proofs that follow:

- $\forall M_{t_n} = M_{t_m} (\mathcal{S}(M_{t_n}) = \mathcal{S}(M_{t_m}))$  if the upper bounding values of the dominated region are constant over time.

- The value of the  $\mathcal{S}$  metric is maximal for  $Z^*$  so the value of the  $\mathcal{S}$  metric is bounded.
- The value of the  $\mathcal{S}$  metric is strictly greater for  $M_{t_j}$  than  $M_{t_i}$  if  $M_{t_j} \text{ OW } M_{t_i}$ .

Let the function `Reduce()` for the  $AA_{\mathcal{S}}$  algorithm be defined as:

```
Reduce( $\{\mathbf{z}\} \cup M_{t-1}$ ):
    Fill( $t$ )
    Domination( $t$ )
    Size( $t$ )
    Steady_state( $t$ )
    return( $M_t$ )
```

with archiving rules `Fill( $t$ )` and `Domination( $t$ )` as previously defined for  $AA_{dom}$ , and the following additional rule:

```
Size( $t$ ):
    if (  $|M_{t-1}| = \text{arcsize} \wedge \mathbf{z}_t \not\in M_{t-1} \wedge \max_{\mathbf{z} \in M_{t-1}} \{\mathcal{S}(M_{t-1} \cup \{\mathbf{z}_t\} \setminus \{\mathbf{z}\})\} > \mathcal{S}(M_{t-1})$  )
    then {  $M_t \leftarrow M_{t-1} \cup \{\mathbf{z}_t\} \setminus \{\mathbf{z}^{min}\}$  where  $\mathbf{z}^{min}$  is randomly selected from  $Z_{min} \subseteq M_{t-1}$ 
        where  $Z_{min} = \{\mathbf{z}^i \in Z \mid \forall \mathbf{z} \in M_{t-1}, \mathcal{S}(M_{t-1} \cup \{\mathbf{z}_t\} \setminus \{\mathbf{z}^i\}) \geq \mathcal{S}(M_{t-1} \cup \{\mathbf{z}_t\} \setminus \{\mathbf{z}\})\}$  }
```

In other words, if the archive is full and  $\mathbf{z}_t$  does not dominate the archive then  $\mathbf{z}_t$  may replace  $\mathbf{z}^{min}$  where  $\mathbf{z}^{min}$  is chosen such that the gain in the  $\mathcal{S}$  metric of the archive is maximized, if this gain is positive.

and, a revised version of the rule `Steady_state( $t$ )` to maintain compatibility with the other rules:

```
Steady_state( $t$ ):
    if (  $|M_{t-1}| = \text{arcsize} \wedge \mathbf{z}_t \not\in M_{t-1} \wedge \max_{\mathbf{z} \in M_{t-1}} \{\mathcal{S}(M_{t-1} \cup \{\mathbf{z}_t\} \setminus \{\mathbf{z}\})\} \leq \mathcal{S}(M_{t-1})$  )
    then  $M_t \leftarrow M_{t-1}$ 
```

In other words, the `Steady_state( $t$ )` rule applies if the archive is full and  $\mathbf{z}_t$  does not dominate the archive and it cannot be added under the `Size( $t$ )` rule because there is no vector in the archive which when replaced by  $\mathbf{z}_t$  leads to a positive increase in the  $\mathcal{S}$  metric of the archive.

As in the previous archiving strategy, it is evident that exactly one rule is executed at time step  $t$ , under algorithm  $AA_{\mathcal{S}}$ .

**Lemma 4.9**  $M_{t-1}$  is a nondominated set and the rule  $\text{Size}(t)$  is executed at time  $t$  implies  $M_t$  is a nondominated set. In other words, rule  $\text{Size}(t)$  maintains the archive as a nondominated set.

**Proof 4.10** Assume  $M_{t-1}$  is a nondominated set and the rule  $\text{Size}(t)$  is executed at time  $t$  and  $M_t$  is not a nondominated set. Since rule  $\text{Size}(t)$  is executed,  $\mathbf{z}_t$  cannot dominate  $M_{t-1}$ . Therefore, we have  $\mathbf{z}_t > M_{t-1} \vee \mathbf{z}_t \in M_{t-1} \vee \mathbf{z}_t \in M_{t-1}$ . If  $\mathbf{z}_t > M_{t-1} \vee \mathbf{z}_t \in M_{t-1}$  then  $\mathbf{z}_t$ 's addition would not increase the  $\mathcal{S}$  value of the archive, contradicting the assumption that rule  $\text{Size}(t)$  is executed. So,  $\mathbf{z}_t \in M_{t-1}$ . When rule  $\text{Size}(t)$  executes,  $M_t = M_{t-1} \cup \mathbf{z}_t \setminus \{\mathbf{z}^{\min} \in M_{t-1}\}$ . Since  $\mathbf{z}_t \in M_{t-1}$ ,  $M_{t-1} \cup \mathbf{z}_t$  is a nondominated set. Therefore  $M_t$  is a nondominated set since the removal of any vector from a nondominated set leaves a nondominated set. This contradicts our assumption that  $M_t$  is not a nondominated set.

Lemma 4.9 shows that, as before, the full set of rules in  $AA_S$  ensures that the archive is always a nondominated set. We now prove some convergence properties of  $AA_S$ .

**Lemma 4.10**  $M_t$  is converged under algorithm  $AA_S$  implies  $M_t$  is a subset of  $Z^*$ .

**Proof 4.11** The proof is the same as that for lemma 4.6. No changes are necessary to the proof because the  $\text{Domination}(t)$  rule which the proof depends upon is unchanged and, as before, the archive is always a nondominated set.

**Lemma 4.11**  $M_{t_n} \neq M_{t_m}$  for  $t_n > t_m$  implies  $\forall t > t_n (M_t \neq M_{t_m})$ .

**Proof 4.12** Assume  $M_{t_n} \neq M_{t_m}$  and  $\exists t_p$  such that  $M_{t_p} = M_{t_m}$  with  $t_p > t_n > t_m$ . Since rules  $\text{Size}(t)$ ,  $\text{Fill}(t)$ , and  $\text{Domination}(t)$  all strictly increase  $\mathcal{S}(M)$ , and rule  $\text{Steady\_state}(t)$  leaves  $\mathcal{S}(M)$  unchanged,  $\mathcal{S}(M_{t_n}) > \mathcal{S}(M_{t_m})$ . By the same token,  $\mathcal{S}(M_{t_p}) \geq \mathcal{S}(M_{t_n})$  thus  $\mathcal{S}(M_{t_p}) > \mathcal{S}(M_{t_m})$ , therefore  $M_{t_p} \neq M_{t_m}$ , a contradiction.

**Lemma 4.12**  $M_t$  converges under algorithm  $AA_S$ .

**Proof 4.13** Assume  $M_t$  never converges i.e.  $\forall t_i, \exists t_j, t_j > t_i, M_{t_j} \neq M_{t_i}$ . This implies that there are an infinite number of different sets  $M_t$ , since none can be revisited (lemma 4.11). However, since the set  $Z$  is finite, so is  $2^Z$ . But all  $M \in 2^Z$  so there are finite different  $M$ .

**Theorem 4.2**  $M_t$  converges to a subset of  $Z^*$  under algorithm  $AA_S$ .

**Proof 4.14** From lemma 4.10 we see that all converged sets are Pareto optimal subsets. From lemma 4.12 we see that  $M_t$  converges.

**Note 4.1** Theorem 4.2 also applies with any metric  $\mathcal{Q}$  in place of  $\mathcal{S}$ , so long as  $\mathcal{Q}$  is bounded and  $\forall M_{t_n} = M_{t_m} (\mathcal{Q}(M_{t_n}) = \mathcal{Q}(M_{t_m}))$  and  $\mathcal{Q}(M_{t_j}) > \mathcal{Q}(M_{t_i})$  if  $M_{t_j} \text{ OW } M_{t_i}$  since only these properties of the  $\mathcal{S}$  metric were used to prove Theorem 4.2.

### 4.3.5 Archiving strategy 4: adaptive grid archiving

#### Overview

The archiving strategy considered in this section, adaptive grid archiving (AGA), is more complicated than the other strategies considered above. It uses a kind of ‘crowding’ procedure in objective space which can obtain a well-distributed, bounded archive of points with minimal computational effort. In this section we shall show that, under certain conditions, this strategy is guaranteed to populate certain regions in the objective space, and to maintain the vectors in these regions. The proofs of this are quite involved and long so we first offer an intuitive description of the archiving strategy and an overview of the key concepts involved in the proofs. For this we first need a short description of the archiving strategy.

The crowding procedure used in AGA works by dividing up the  $K$ -dimensional objective space occupied by the vectors in the archive  $M_t$  at iteration  $t$ , into different rectangular polytopes called grid regions. The number of grid regions is set *a priori* by the user, by choosing the value of a parameter,  $div$ , which sets how many divisions of the objective space there are in each objective. The number of grid regions thus remains constant over time, but the space occupied by the vectors changes so that the location and extent of the grid regions in objective space adapts and changes over time as the vectors in  $M_t$  change. This is shown in Figure 4.6. At time  $t$ , each vector in  $M_t$  occupies exactly one grid region, and the number of vectors occupying a particular grid region is called its population. We are able to prove that the boundaries of the adaptive grid, and therefore of all grid regions, under certain conditions, always converge. That is, their location and extent is unchanging after some time  $t$ . We use the base assumption that the grid regions are converged to simplify the proofs relating to the convergence of the archive itself.

The archiving rules in AGA are such that when the archive is at its capacity,  $arcsize$ , new nondominated vectors generated by  $\text{Gen}(t)$  can be still accepted if there is at least one region with a population greater than the current population of the grid region that the new nondominated vector would occupy. In this case, a randomly selected vector from one of the

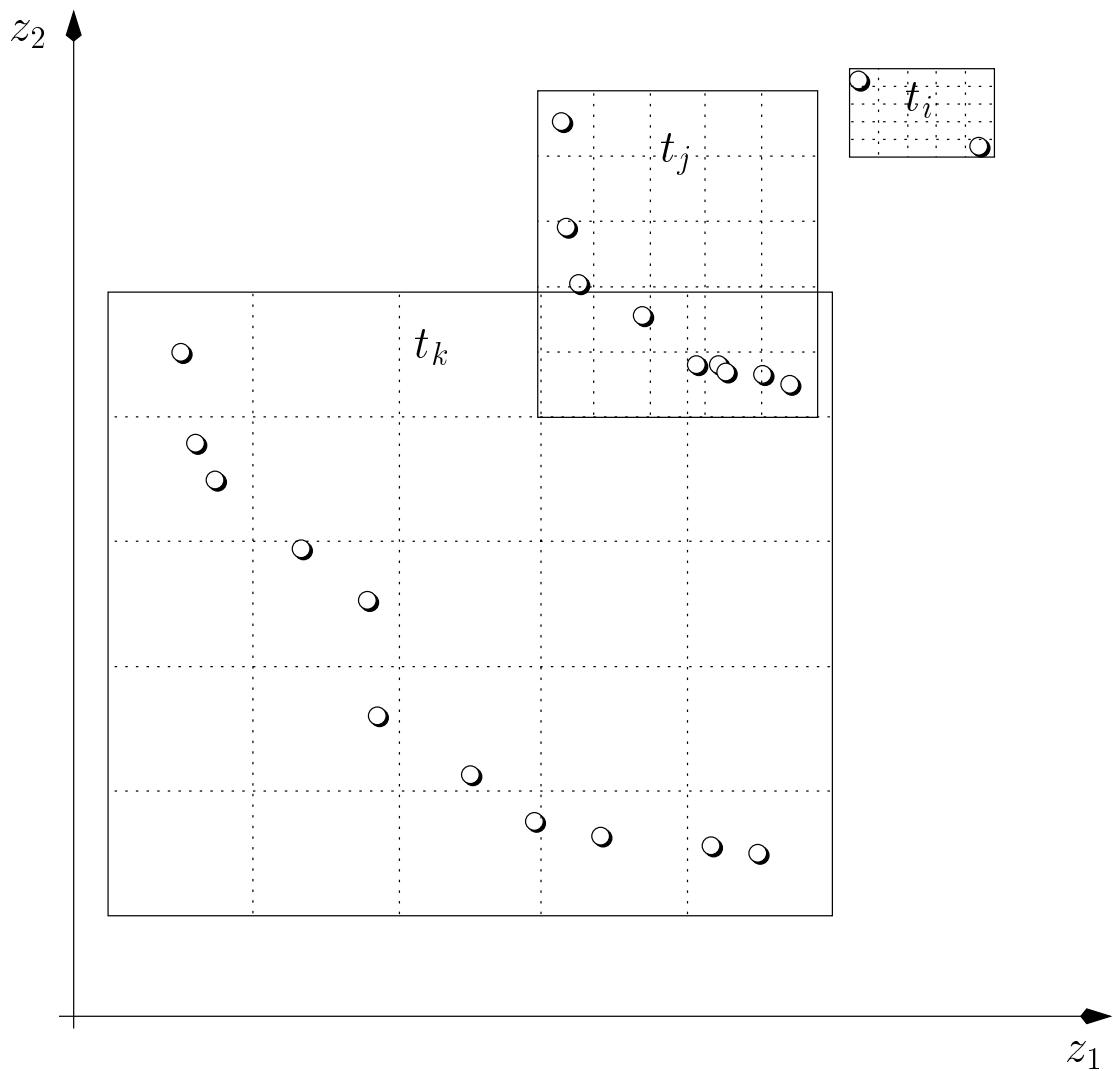


Figure 4.6: How the adaptive grid changes its location and shape in objective space as the vectors in  $M_t$  change over iterations  $t_k > t_j > t_i$ .

‘most crowded’ regions is removed from  $M_t$ . This rule helps to distribute vectors evenly in objective space.

A further pair of rules are designed to ensure that the nondominated vectors stored in the archive cover the largest possible range in each objective. The first rule accepts a nondominated point if it increases the range of the grid in objective space, i.e. the range of objective values represented by the vectors in the archive. The second rule of the pair forbids removal of any nondominated vector which is currently uniquely extremal on any objective.

The distribution of points obtained by the application of the archiving rules in AGA depends upon several factors. Amongst these factors are the number of regions there are, the location and distribution of vectors in the true Pareto front, and the size of the archive. In the following discussion, we try to give a picture of how these factors affect the kind of distribution we should expect to obtain. We assume, in all examples, that the boundaries of the adaptive grid have already converged, that is, that the location and extent of all regions remains constant.

Consider Figures 4.7–4.10. These figures show a finite, discrete Pareto front and some approximations to it. Figure 4.7 shows the true Pareto front and the regions it would occupy if a grid of ten divisions in each objective were used ( $div = 10$ ). The regions occupied by the true Pareto front are called Pareto occupied regions (*PORs*) in our notation. In Figure 4.8 we see what will *ideally* happen if *arcsize* is the same size as the number of *PORs*: the points will all end up in their own *POR*, resulting in an even distribution that is near the entire Pareto front. No region is overly crowded so this distribution should be stable with respect to the crowding procedure<sup>1</sup>. In Figure 4.9(a) we see what happens if the *arcsize* is too small compared to the number of Pareto occupied grid regions. They may end up in regions adjacent to each other (except for the extremal vector on the right), and because there are not enough of them to cover all the regions of interest, the resulting distribution is poor. In Figure 4.9(b) we see what might happen if the *arcsize* is too large for the setting of the number of grid regions: an uneven distribution again. This can occur because although the crowded regions are constantly having points removed from them, they may also have points added back in at a similar rate, simply because points in these regions may be frequently generated by  $Gen(t)$ . Note that this distribution is preferable to having too few points, however, because at least the points can cover all of the Pareto occupied regions, even if they are not distributed evenly between them. Figure 4.10 shows how the larger size of archive considered in Figure 4.9(b) could be better served by using a grid with more divisions.

So, Figures 4.7–4.10 suggest that getting about the right size of archive for the number of *PORs* is important. Of course, one cannot know how many *PORs* there will be in advance.

---

<sup>1</sup>It remains to be shown whether it is stable in terms of all of the archiving rules taken together.

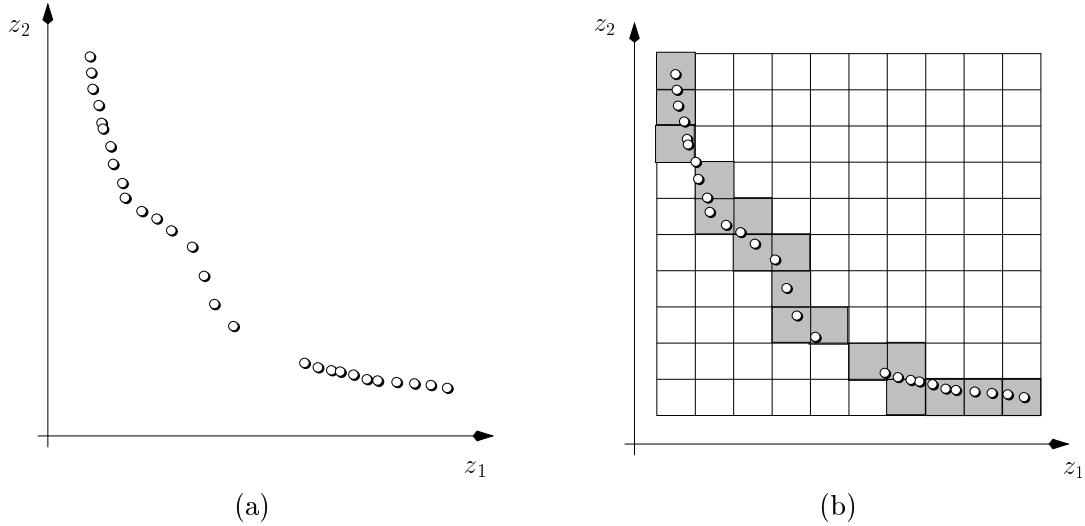


Figure 4.7: In (a) the true Pareto front of a discrete, finite search space is shown with white points representing Pareto optimal points. In (b) the same front is plotted against a grid of regions used for ‘crowding’. The grid shown has 10 divisions in each of the two objectives. The shaded regions are the Pareto occupied regions: those occupied by a Pareto optimal point. If we could guarantee finding points in each shaded region we would have a well distributed approximation to the true Pareto front.

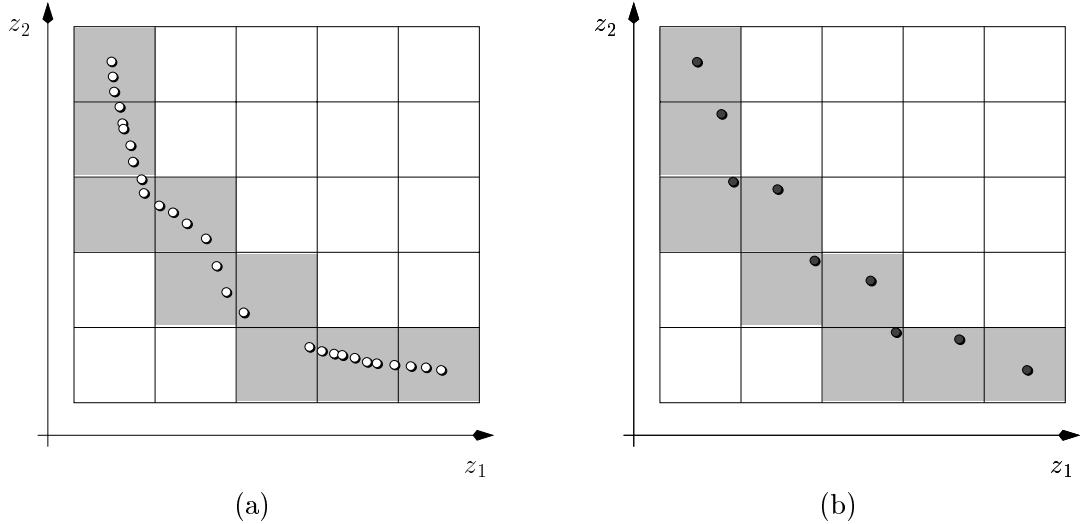


Figure 4.8: In (a) the same front as in Figure 4.7 is shown against a grid with fewer divisions in each objective. In (b) an approximation to this front is shown, where exactly one vector occupies each Pareto occupied region. Clearly these points form a fairly even distribution and approximate the true Pareto front quite well. This is the kind of desirable distribution that might be represented by the archive if *arcsize* equals the number of Pareto occupied regions.

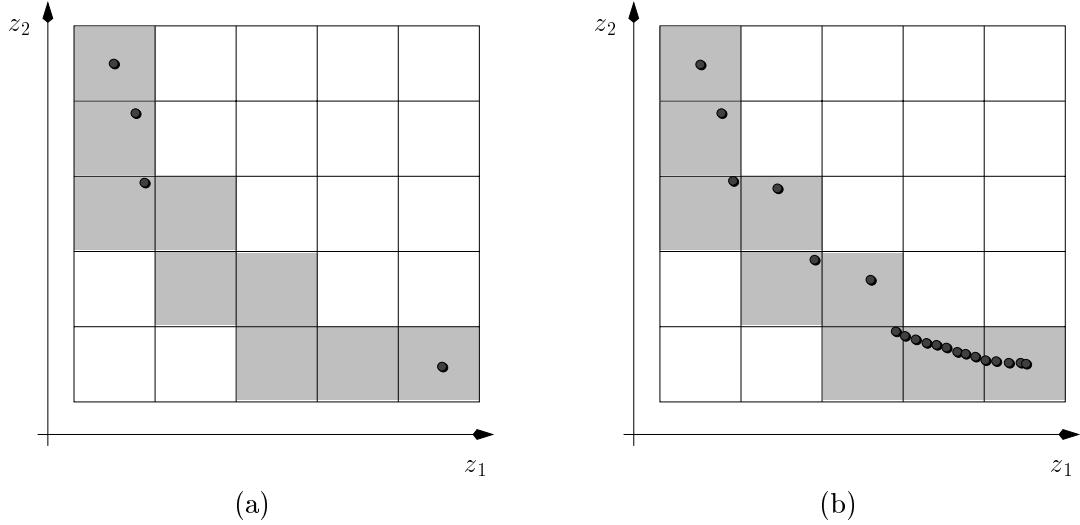


Figure 4.9: In (a) *arcsize* is too small: only about half as big as the number of Pareto occupied regions. This can lead to the situation shown where all the points cluster together in regions that are close to each other. In (b) the opposite effect is shown. Here *arcsize* is rather large. Although there is a point in every Pareto occupied region, some regions are very crowded. This could occur if some regions are much more densely populated in the feasible set than others or if points in them are easier to generate.

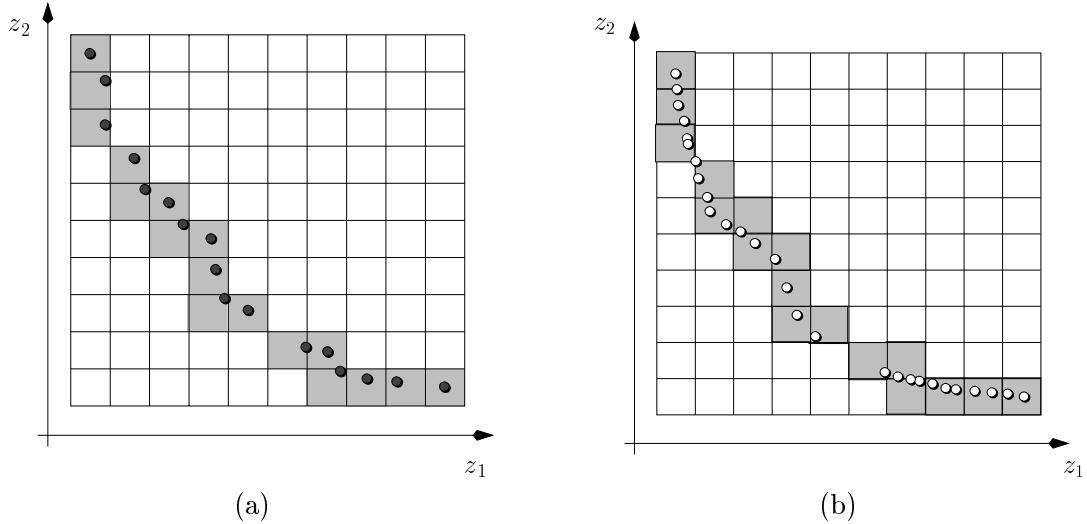


Figure 4.10: With *arcsize* the same as in Figure 4.9(b), it is better to set the grid to have the ‘right’ number of regions as shown in (a). Doing this leads to a better approximation to the true Pareto front shown again in (b), and a more even distribution of points.

However, having enough available points in the archive is also desirable for another important reason: to help the occupied regions to eventually converge to a stable set. Recall that the crowding procedure uses a rule that can accept a nondominated vector into the archive if there is another region that is more crowded than the new vector's region. Now, this rule could cause instability in the occupancy of regions. In Figure 4.11(a) this fact is demonstrated. The plot shows a situation where the archive is full and the vectors have arranged themselves in different regions so that all the occupied regions have a population of only one. When a new vector which would go in an unoccupied region is generated, it will cause one of the occupied regions to be 'lost'. In this situation, the occupied regions might continually alternate, and never converge to a stable set. To avoid this happening, all that is required is that the archive is larger than the number of regions that can be populated by any *nondominated set*. This is demonstrated in Figure 4.11(b). An important part of the proofs shows that the supply of vectors is sufficient to avoid occupied regions alternating if *arcsize* is greater than  $div^K - (div - 1)^K + 2K$ .

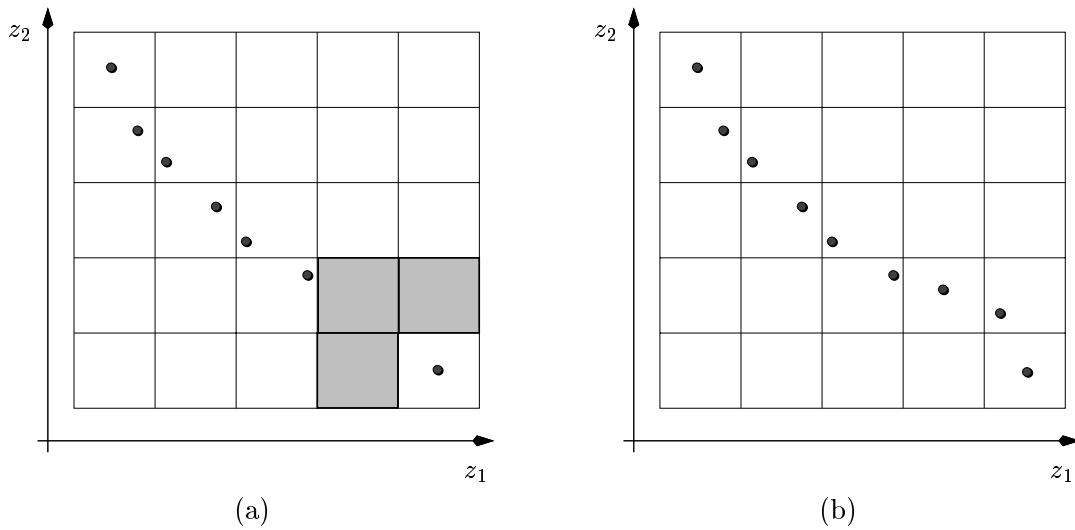


Figure 4.11: Plot (a) shows the points in a full archive of *arcsize* = 7. If a nondominated point is generated which lies in any of the shaded regions then it will be allowed to enter the archive but another point will be removed. This will cause one of the occupied regions to be lost. In plot (b) the archive is larger: *arcsize* = 9. There are enough points so that any occupied region  $A$  can only be lost if a vector in another region dominates all the points in  $A$ .

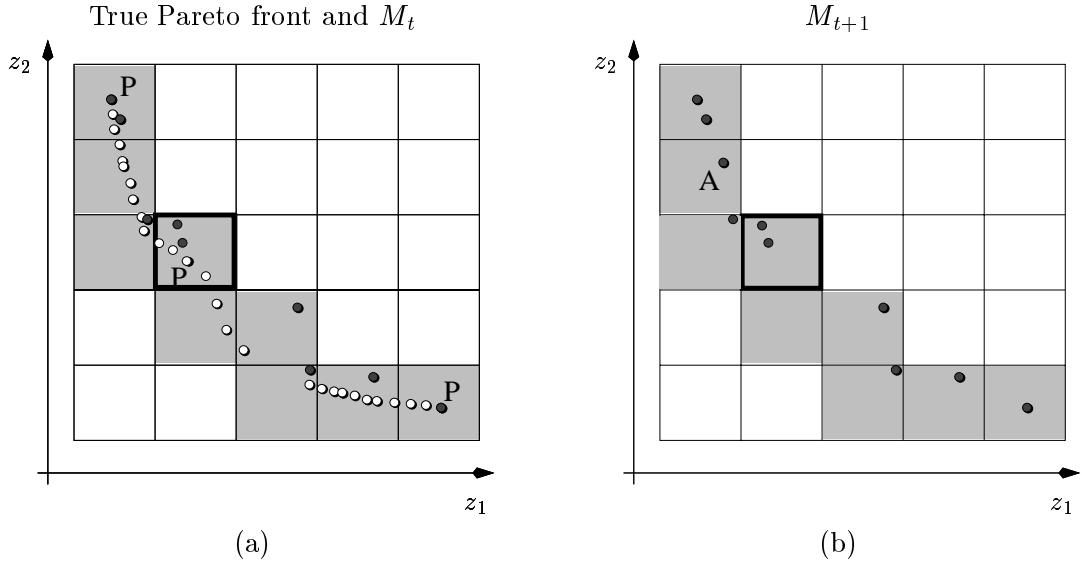


Figure 4.12: In (a), the true Pareto front (white points) and the archive at time  $t$  (dark points) are shown on the same plot, with the *PORs* shaded. The three points labelled  $P$  are Pareto optimal points that are already in  $M_t$ , and  $|M_t| = \text{arcsize} = 10$ , i.e. the archive is full. In (b) the archive at time  $t + 1$  is shown. Point  $A$  was generated by  $\text{Gen}(t)$  and it is allowed to join because it is nondominated and its region is less crowded than the region highlighted with a dark box. The highlighted region has consequently had a solution removed at random, and this happens to be the Pareto optimal point labelled  $P$  in the previous plot.

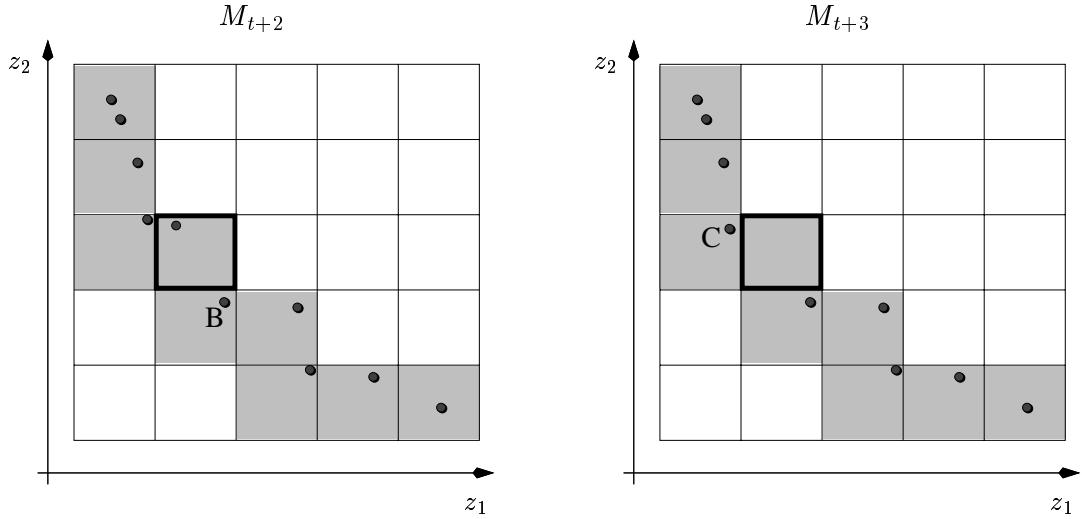


Figure 4.13: These two plots show the next two time steps of the archive. First, point  $B$  joins at iteration  $t + 2$ , and another point is removed from the highlighted region. Finally, point  $C$  is generated at time step  $t + 3$  and this point dominates the remaining point in the highlighted region at the last time step, thus causing it to be removed. Overall, the highlighted region has evolved from having three points in it including a Pareto optimal one, to having no points in it.

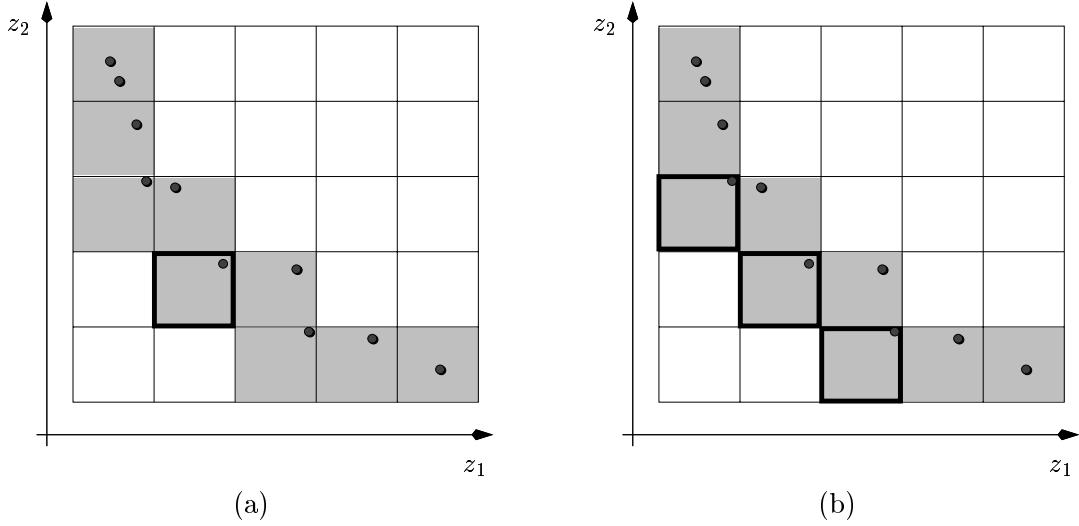


Figure 4.14: In plot (a) the highlighted region can never be lost provided there is a sufficiently large archive. This is because there is no region which contains feasible points and which has coordinates that are lower than the coordinates of this region in any objective dimension. In other words there is no region to the left or below it which contains feasible points. Thus points in other regions cannot ever dominate any points in the highlighted region. This is sufficient to guarantee that this region will always remain occupied. We call a region such as this, a critical Pareto occupied region (*CPOR*). Plot (b) shows the other *CPORs* in this objective space. Notice that because the *CPORs* are populated, no region with coordinates higher in both objective dimensions can ever become populated. This means that all future points will be restricted to the shaded regions. In this case these correspond to the Pareto occupied regions (see Figure 4.8(a)).

Unfortunately, sizing the archive correctly is not sufficient to ensure that all the regions occupied by the Pareto front will become and remain populated. In Figures 4.12 and 4.13, the evolution of an archive over four time steps is shown. The plots indicate that even if a Pareto optimal point is in the archive at time  $t$ , it will not necessarily stay there. Furthermore, the region it occupies need not even remain occupied because all of the vectors in it can at a later time become dominated. This simple example demonstrates that we will not be able to guarantee that all the Pareto occupied regions will remain stably populated by the archive, even if it is larger than  $\text{div}^K - (\text{div} - 1)^K + 2K$ .

Fortunately, we can prove a slightly weaker property. Consider Figure 4.14(a), which shows another approximation to the Pareto front considered previously in Figure 4.7. Provided the size of the archive is large enough, the highlighted region will *always* remain occupied. This is clear because no feasible vector in any other region dominates *any* point in the highlighted region. Therefore, any single point occupying the highlighted region can only be ‘lost’ if it is

replaced by another vector in the *same* region which dominates it. Thus the region can never be lost. We call this kind of region a critical Pareto occupied region (*CPOR*). Figure 4.14(b) shows all of the *CPORs* in the objective space. Our proofs show that all of the *CPORs* will become occupied, and once occupied they will forever remain so. We are also able to prove that all of the vectors in the archive will eventually occupy a (fluctuating) set of regions that are non-inferior to the *CPORs* (see Figure 4.14 to get some feeling for this.)

### Formal definitions and proofs

The above descriptions should give some motivation as to the meaning and relevance of the proofs which follow. However, the proofs formalize these notions and allow us to make more general statements about the convergence properties of the adaptive grid archiving strategy. For convenience, we include Table 4.3.5 which tabulates the key terms introduced and used in the following proofs.

**Definition 4.6** *Let the nondominated set of vectors from amongst the archive at time step  $t - 1$  together with the new vector  $\mathbf{z}_t$  generated at time step  $t$  be denoted by  $N_t$ . That is,  $N_t = \text{ND}(M_{t-1} \cup \{\mathbf{z}_t\})$ . Note that  $|N_t|$  may be larger than *arcsize*.*

**Definition 4.7** *Let the minimum and maximum scalar values of an objective  $k$  amongst the vectors in a vector set  $Z$  be denoted  $\text{min}_{\mathbf{z} \in Z}(z_k)$  and  $\text{max}_{\mathbf{z} \in Z}(z_k)$ , respectively. That is,  $\text{min}_{\mathbf{z} \in Z}(z_k) = \min_{\mathbf{z} \in Z}(z_k)$  and  $\text{max}_{\mathbf{z} \in Z}(z_k) = \max_{\mathbf{z} \in Z}(z_k)$ .*

**Definition 4.8** *There are  $2K$  boundaries of the adaptive grid:  $ub_{k,t}$  and  $lb_{k,t}$  for all  $k \in 1..K$ . The boundaries are set so that  $\forall t, \forall k, (ub_{k,t} > \text{max}_{\mathbf{z} \in N_t}(z_k)) \wedge (lb_{k,t} < \text{min}_{\mathbf{z} \in N_t}(z_k))$ . (See rule *Update\_boundaries*( $t$ ) below.)*

**Definition 4.9** *The rectangular polytope defined by the ‘corners’  $(ub_{1,t}, ub_{2,t}, \dots, ub_{K,t})$  and  $(lb_{1,t}, lb_{2,t}, \dots, lb_{K,t})$  is divided into a set  $R_t$  of similar rectangular polytope regions  $r_{\mathbf{i},t} \in R_t$ , where  $\mathbf{i} = (i_1, i_2, \dots, i_K)$  is the co-ordinate vector of the region, and  $\forall k \in 1..K, i_k \in 1..div$  where  $div \in j \mid j \geq 2, j \in \mathbb{Z}$  is a constant parameter, the number of divisions of the objective space in each dimension, set by the user. The set of all region co-ordinates is  $I$  and  $|I| = div^K$ .*

**Definition 4.10** *The boundaries of the regions  $r_{\mathbf{i},t}$  are given by:*

$\forall k \in 1..K, rub_{k,\mathbf{i},t} = lb_{k,t} + i_k/div.(ub_{k,t} - lb_{k,t})$  and  
 $rlb_{k,\mathbf{i},t} = lb_{k,t} + (i_k - 1)/div.(ub_{k,t} - lb_{k,t})$ . See Figure 4.16.

| Symbol                       | Meaning   | References                  |
|------------------------------|---|-----------------------------|
| $AA_{Reduce \in RED}$        | A generic archiving algorithm   | Figure 4.5                  |
| $CPOR$                       | A critical Pareto occupied region: a converged region which contains no vectors that can be dominated by feasible vectors in a different region | Definition 4.25, Lemma 4.15 |
| $CR_t$                       | The set of crowded regions  | Definition 4.18             |
| $div$                        | The number of divisions in each objective dividing up the adaptive grid into separate regions   | Definition 4.9              |
| $Gen(t)$                     | A process that generates every vector in $Z$ with positive probability  | Definition 4.4              |
| $K$                          | The number of objectives in the vector objective space  | Definition 4.7              |
| $lb_{k,t}$                   | The lower boundary of the adaptive grid in objective $k$ at time $t$  | Definition 4.7              |
| $M_t$                        | The archive at time step $t$  | Definition 4.3              |
| $minz_{k,N_t}$               | The minimum value of the $k$ th component of any vector in $N_t$  | Definition 4.7              |
| $maxz_{k,N_t}$               | The maximum value of the $k$ th component of any vector in $N_t$  | Definition 4.7              |
| $N_t$                        | The nondominated vectors from the union of $M_{t-1}$ and $\mathbf{z}_t$   | Definition 4.6              |
| $ND(Z_1)$                    | The set of nondominated vectors from $Z_1$  |                             |
| $Occ(r_{\mathbf{i},t}, Z_1)$ | The set of vectors in $Z_1$ occupying a region $r_{\mathbf{i},t}$   | Definition 4.14             |
| $O_W$                        | The weak outperformance relation  |                             |
| $PNIR$                       | Pareto non-inferior region: a region that is weakly inferior to a $CPOR$ or is a $CPOR$   | Definition 4.26, Lemma 4.6  |
| $POR$                        | Pareto occupied region: a region that is occupied by a vector in $Z^*$  | Definition 4.12             |
| $p(r_{\mathbf{i},t})$        | The number of vectors in $ M_{t-1} $ occupying region $r_{\mathbf{i},t}$  | Definition 4.13             |
| $r_{\mathbf{i},t}$           | A region with coordinates $\mathbf{i}$ at time $t$  | Definition 4.9              |
| $rt_{\mathbf{c}}$            | A root region of a region with co-ordinate vector $\mathbf{c}$  | Definition B.1              |
| $ub_{k,t}$                   | The upper boundary of the adaptive grid in objective $k$ at time $t$  | Definition 4.7              |
| $Z$                          | The finite objective space  | Definition 4.1              |
| $Z^*$                        | The Pareto optimal front  | Definition 4.2              |
| $Z_{c,t}$                    | The set of vectors available for removal from the crowded regions, $CR_t$   | Definition 4.19             |
| $\mathbf{z}_t$               | A vector in $Z$ generated at time $t$ by $Gen(t)$   | Definitions 4.1 and 4.4     |

**Definition 4.11** We say that a vector  $\mathbf{z}$  in  $M_{t-1}$  occupies a region  $r_{\mathbf{i},t}$  if  $\forall k$ ,  $z_k \geq rlb_{k,\mathbf{i},t} \wedge z_k < rub_{k,\mathbf{i},t}$ . We call a region  $r_{\mathbf{i},t}$  that has a vector  $\mathbf{z}$  in  $M_{t-1}$  occupying it, an occupied region.

**Definition 4.12** We say that the vector  $\mathbf{z} \in Z^*$  Pareto occupies a region  $r_{\mathbf{i},t}$  if for all  $k$ ,  $z_k \geq rlb_{k,\mathbf{i},t} \wedge z_k < rub_{k,\mathbf{i},t}$ , even if the vector is not in  $M_{t-1}$ . We call a region that has a  $\mathbf{z} \in Z^*$  (and not necessarily in  $M_{t-1}$ ) occupying it, a Pareto occupied region (POR).

**Definition 4.13** The population  $p(r_{\mathbf{i},t})$  of a region is the number of vectors in  $M_{t-1}$  occupying it.

**Definition 4.14** The set of vectors from a vector set  $Z$  occupying a region is  $\text{Occ}(r_{\mathbf{i},t}, Z) \subseteq Z$ .

**Definition 4.15** The region occupied by a vector  $\mathbf{z}$  is denoted  $r_{\mathbf{z},t}$ .

If there is no difference between  $\min z_{k,N_t}$  and  $\max z_{k,N_t}$  for some objective  $k$  then the adaptive grid boundaries are undefined, and the boundaries of all regions are undefined. In this case, the rules called by `Reduce()` that use the grid populations to perform crowding are also undefined. In the following, we make the assumption that the grid boundaries are always well-defined. This is a reasonable assumption since the archive is full whenever any of the rules that use the adaptive grid are executed, so there will normally be some difference between the vectors in  $M_t$  in each of the objectives.

We also make the constraint that  $\text{arcsize} > 2K$ . This ensures that the archive is large enough to accommodate all nondominated extremal vectors.

In the adaptive grid archiving strategy we want to protect uniquely extremal vectors from being removed from the archive once they have entered it (except by domination), so that the vectors in the archive will converge to a set which covers the largest possible range in objective space, in each objective. However, the archiving strategy will be removing vectors from crowded regions. To avoid removing extremal vectors from these regions we will need a way of counting the number of vectors that occupy a region, (i.e. how crowded it is) *excluding* the number of uniquely extremal vectors. In the following, we define some terms needed to do this.

**Definition 4.16** Let the set of uniquely extremal vectors in  $N_t$  be defined:

$$Z_t^{ext} = \{\mathbf{z}^{ext} \in N_t \mid (\exists k \in 1..K, \nexists \mathbf{z} \in N_t, \mathbf{z} \neq \mathbf{z}^{ext}, z_k \leq z_k^{ext}) \vee (\exists k \in 1..K, \nexists \mathbf{z} \in N_t, \mathbf{z} \neq \mathbf{z}^{ext}, z_k \geq z_k^{ext})\}$$

**Definition 4.17** Let the population of non-uniquely extremal (nue) vectors of a region at time  $t$  be defined:

$$p_{nue}(r_{\mathbf{i}}, t) = p(r_{\mathbf{i},t}) - |\text{Occ}(r_{\mathbf{i},t}, Z_{t-1}^{ext})|$$

**Definition 4.18** Let the set of crowded regions be defined:

$$CR_t = \{r_{\mathbf{i},t} \in R \mid p_{nue}(r_{\mathbf{i},t}) = \max_{\mathbf{i} \in I} (p_{nue}(r_{\mathbf{i},t}))\}$$

**Definition 4.19** Let the set of vectors in the set of crowded regions that are available for removal from the archive at time  $t$  be defined:

$$Z_{c,t} = \bigcup_{r_{\mathbf{i},t} \in CR_t} \text{Occ}(r_{\mathbf{i},t}, M_{t-1})$$

In other words, this is the set of vectors that are in the set of most crowded regions, where the most crowded regions are defined as those with the largest population, not counting uniquely extremal vectors. Let the vector  $\mathbf{z}^{c,t}$  be a vector selected uniformly at random from within  $Z_{c,t}$ .

Let the function `Reduce()` for the  $AA_{grid}$  algorithm be defined as:

```
Reduce({z} ∪ M_{t-1}):
    Update_boundaries(t)
    Fill(t)
    Domination(t)
    Diverge(t)
    Low_pop_region(t)
    Steady_state(t)
    return(M_t)
```

with archiving rules  $\text{Fill}(t)$  and  $\text{Domination}(t)$  as previously defined for  $AA_{dom}$ , and the following additional rules:

```

Update_boundaries( $t$ ):
    foreach ( $k \in 1..K$ )
    {
         $range_{k,t} \leftarrow \max_{z \in N_t}(z_k) - \min_{z \in N_t}(z_k)$ 
         $ub_{k,t} \leftarrow \max_{z \in N_t}(z_k) + (1/(2.div))(range_{k,t})$ 
         $lb_{k,t} \leftarrow \min_{z \in N_t}(z_k) - (1/(2.div))(range_{k,t})$ 
        Re-calculate grid region boundaries for  $k$ 
    }

```

In other words, `Update_boundaries( $t$ )` updates the boundaries based on the nondominated set  $N_t$  defined above. Note that this rule is not conditional: it is executed at every time step. The effect of the rule is to set the regions at time  $t$  *before* the archive  $M_t$  is updated to allow the archive to be updated based on the current region boundaries. The effect of the `Update_boundaries( $t$ )` rule is illustrated in Figure 4.15

`Diverge( $t$ )`:

```

if ( $|M_{t-1}| = arcsize \wedge \mathbf{z}_t \sim M_{t-1} \wedge (\exists k, (maxz_{k,N_t} \neq maxz_{k,M_{t-1}}) \vee (minz_{k,N_t} \neq minz_{k,M_{t-1}})) \wedge Z_{c,t} \neq \emptyset$ ) then {  $M_t \leftarrow (M_{t-1} \cup \{\mathbf{z}_t\}) \setminus \{\mathbf{z}^{c,t}\}$  }

```

In other words, if the archive is full and the new vector has a component beyond the extremes of the archive at the last time step. then accept the new vector and remove a vector from one of the set of most crowded regions, ensuring it is not extremal on any objective. Notice that this rule (and the two that follow below) also depend on the condition that the set of vectors available for removal from the set of crowded regions is not empty. This is always true provided that  $arcsize > 2K$ , as we have already assumed. Note also that the fact that the boundaries have changed implies that  $\mathbf{z}_t < M_{t-1}$  or  $\mathbf{z}_t \sim M_{t-1}$ ; the latter case will have been dealt with by the `Domination( $t$ )` rule.

`Low_pop_region( $t$ )`:

```

if ( $|M_{t-1}| = arcsize \wedge \mathbf{z}_t \sim M_{t-1} \wedge (\forall k, (maxz_{k,N_t} = maxz_{k,M_{t-1}}) \wedge (minz_{k,N_t} = minz_{k,M_{t-1}})) \wedge \exists \mathbf{i} \in I, p_{nue}(r_{\mathbf{i},t}) > 1 \wedge \exists \mathbf{i} \in I, p_{nue}(r_{\mathbf{i},t}) > p_{nue}(r_{\mathbf{i}_{\mathbf{z}_t},t}) \wedge Z_{c,t} \neq \emptyset$ ) then  $M_t \leftarrow (M_{t-1} \cup \{\mathbf{z}_t\}) \setminus \{\mathbf{z}^{c,t}\}$ 

```

In other words, if the archive is full and the new vector is nondominated with respect to the archive and the new vector lies within the extremes of the archive and the region (at the last time step) that the new vector would go in is less crowded than some other region(s), and there is a region with a population of non-uniquely extremal vectors of greater than 1, then accept the new vector and remove a vector from one of the set of most crowded regions.

The rule `Steady_state(t)` is also updated to remain compatible with the other rules, as follows:

`Steady_state(t):`

```

if (  $|M_{t-1}| = \text{arcsize} \wedge (\forall k, (maxz_{k,N_t} = maxz_{k,M_{t-1}}) \wedge (minz_{k,N_t} = minz_{k,M_{t-1}})) \wedge$ 
 $(Z_{c,t} = \emptyset \vee \exists \mathbf{i} \in I, p_{nue}(r_{i,t-1}) > p_{nue}(r_{i,z_t,t-1}) \vee \exists \mathbf{i} \text{ such that } p_{nue}(r_{i,t}) \geq 1) \wedge \mathbf{z}_t \not\in M_{t-1}$ 
) then  $M_t \leftarrow M_{t-1}$ 

```

By inspection, it can be seen that in function `Reduce` for algorithm  $AA_{grid}$  exactly two rules will execute at each time step,  $t$ : rule `Update_boundaries(t)` always executes, and then exactly one of `Fill(t)`, `Domination(t)`, `Diverge(t)`, `Low_pop_region(t)`, `Steady_state(t)` executes. The boundaries of the grid at time  $t$  are set so that the nondominated vectors from the archive at the *previous time step* and the new vector  $\mathbf{z}_t$  all lie within the boundaries. Figure 4.15 shows how the boundaries are set.

Also, by inspection, it is clear that the archiving rules ensure that the archive is always a nondominated set.

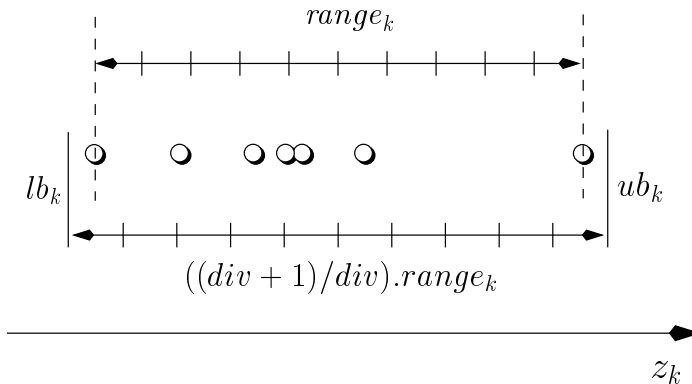


Figure 4.15: When a new vector  $\mathbf{z}_t$  increases the range of the grid in some objective  $k$ , a new  $range_{k,t}$  is calculated, and the grid boundaries are set so that  $range_{k,t} \leftarrow \max_{z \in N_t}(z_k) - \min_{z \in N_t}(z_k)$  and  $ub_{k,t} \leftarrow \max_{z \in N_t}(z_k) + (1/(2.div))(range_{k,t})$  and  $lb_{k,t} \leftarrow \min_{z \in N_t}(z_k) - (1/(2.div))(range_{k,t})$ , which means that the two extremal vectors in objective  $k$  will be located at the center of the outer grid regions.

We begin the proofs relating to the adaptive grid archiving strategy by showing that the lower boundaries of the adaptive grid always converge.

**Lemma 4.13** *If a vector  $\mathbf{z} \in Z_f$  with component  $z_k = \min_{z \in Z_f}(z_k)$  for some  $k$  is generated at time  $t$ , then  $\min_{z \in N_t}(z_k) = \min_{z \in Z_f}(z_k)$ .*

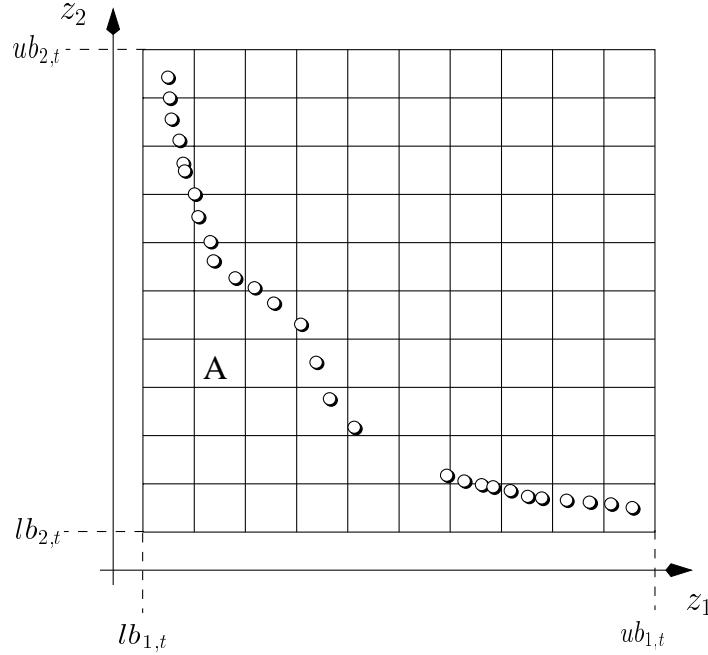


Figure 4.16: The figure shows the meaning of  $ub_{k,t}$  and  $lb_{k,t}$ , for a two objective plot. The region labelled A has co-ordinates  $\mathbf{i} = (2, 4)$  and boundaries,  $rub_{1,\mathbf{i},t} = 2/10.(ub_{1,t} - lb_{1,t})$ ,  $rlb_{1,\mathbf{i},t} = 1/10.(ub_{1,t} - lb_{1,t})$ ,  $rub_{2,\mathbf{i},t} = 4/10.(ub_{2,t} - lb_{2,t})$  and  $rlb_{2,\mathbf{i},t} = 3/10.(ub_{2,t} - lb_{2,t})$ .

**Proof 4.15** Assume that at time  $t$ , a vector  $\mathbf{z} \in Z_f$  with component  $z_k = \min_{Z_f} z_k$  for some  $k$  is generated. It will enter the archive if  $\text{Domination}(t)$  or  $\text{Diverge}(t)$  or  $\text{Low\_pop\_region}(t)$  or  $\text{Fill}(t)$  executes, and hence  $\min_{M_t} z_k = \min_{Z_f} z_k$ . If  $\mathbf{z}^*$  does not enter the archive, it is because  $\text{Steady\_state}(t)$  executes. However, one of the conditions for  $\text{Steady\_state}(t)$  to execute is that  $\exists k$  such that  $\min_{Z_f} z_k \neq \min_{M_{t-1}} z_k$ . So we have  $z_k = \min_{Z_f} z_k$  and  $\forall k, \min_{Z_f} z_k = \min_{M_{t-1}} z_k$ , and  $\text{Steady\_state}(t)$  executes. Therefore, since  $\mathbf{z} \in N_t$ , then  $\min_{M_{t-1}} z_k = \min_{Z_f} z_k = \min_{M_t} z_k$ , and because  $\text{Steady\_state}(t)$  executes  $\min_{M_t} z_k = \min_{M_{t-1}} z_k$ , so  $\min_{M_t} z_k = \min_{Z_f} z_k$ , as required.

**Lemma 4.14** If, for some  $k \in 1..K$ ,  $\exists \mathbf{z} \in M_{t_m}$  with  $z_k = \min_{Z_f} z_k$  then  $\forall t > t_m, \exists \mathbf{z} \in M_t$  with  $z_k = \min_{Z_f} z_k$ .

**Proof 4.16** Assume that for some  $k \in 1..K$ ,  $\exists \mathbf{z} \in M_{t_m}$  with  $z_k = \min_{Z_f} z_k$ . We show that no archiving rule is capable of removing all the vectors with component  $z_k = \min_{Z_f} z_k$ .

Let us consider each of the rules that can remove vectors. These are the rules  $\text{Diverge}(t)$ ,  $\text{Low\_pop\_region}(t)$ , and  $\text{Domination}(t)$ .

$\text{Domination}(t)$  can remove multiple vectors at once. However, it cannot remove any vector with component  $z_k = \min z_{k,Z}$  except by replacing it with another vector also with a component  $z_k = \min z_{k,Z}$  since the new vector must dominate the one(s) replaced.

$\text{Diverge}(t)$  can remove only one vector  $\mathbf{z}^{c,t}$  from the archive. If there is only one vector  $\mathbf{z} \in M_t$  with  $z_k = \min z_{k,Z}$  then it is a unique extremum in the archive and  $\text{Diverge}(t)$  cannot remove it by the definition of  $\text{Diverge}(t)$ . If there are  $n > 1$  vectors in the archive with  $z_k = \min z_{k,Z}$  then  $\text{Diverge}(t)$  may remove one but one or more will remain.

$\text{Low\_pop\_region}(t)$  can also remove only one vector from the archive and only if it is not a unique extremum. The same argument as for  $\text{Diverge}(t)$  applies.

**Theorem 4.3** *The lower boundaries of the grid  $lb_{k,t}$  converge for all  $k$ .*

**Proof 4.17** *To show that the lower boundaries converge it is sufficient to show that there is a time  $t_m$  such that  $\forall t \geq t_m, \forall k, \min z_{k,M_t} = \min z_{k,Z_f}$ . This is proved by Lemma 4.13 and Lemma 4.14.*

Although it is possible to prove that the lower boundaries converge in all cases, unfortunately this is not true for the upper boundaries. However, it is possible to specify an additional condition under which the upper boundaries can be proved to converge. Appendix A contains proofs showing that the upper boundaries converge under this condition.

**Assumption 4.1** *The upper boundaries  $ub_{k,t}$  converge for all  $k$ .*

**Corollary 4.2** *Since the lower boundaries of the grid converge (Theorem 4.3) and by Assumption 4.1, the upper boundaries of the grid also converge, then so do all the boundaries of all of the regions in the grid.*

In the following, our proofs rely on Corollary 4.2. This allows us to consider all of the regions as being static, enabling us to prove that some of the regions containing Pareto optimal points will become constantly occupied by points in the archive. We now introduce some additional terminology relating to the grid regions, needed for these proofs.

**Definition 4.20** *Let a set of regions whose boundaries are converged be called a converged set of regions, and each region in the set be called a converged region.*

**Definition 4.21** If a converged set of regions also has a subset of regions which remain occupied over time then we say that there is a converged set of occupied regions,  $R_{COR}$ . The regions that comprise this set are called constantly occupied regions (CORs).

The following three definitions introduce dominance relationships between regions, and are illustrated by Figure 4.17.

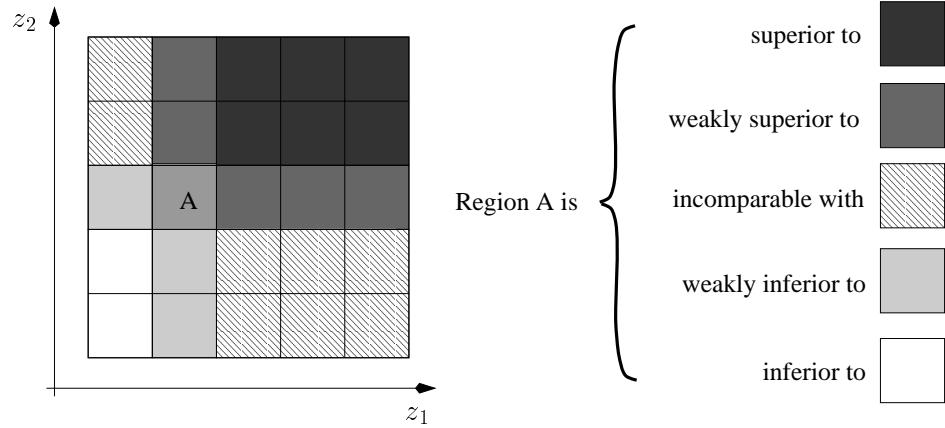


Figure 4.17: The dominance relationships between a region  $A$  and the other regions in a grid.

**Definition 4.22** We say that a region  $r_1$  is superior to another region  $r_2$  iff  $r_1$ 's coordinates are all strictly less than  $r_2$ 's. In addition,  $r_2$  is said to be inferior to  $r_1$ . Notice that all vectors in  $r_2$  are dominated by vectors in  $r_1$ . Thus, any region that is inferior to another Pareto occupied region cannot itself be a Pareto occupied region.

**Definition 4.23** We say that a region  $r_1$  is weakly superior to a region  $r_2$  if  $r_1$ 's coordinates dominate  $r_2$ 's and  $r_1$  is not superior to  $r_2$ . In addition,  $r_2$  is said to be weakly inferior to  $r_1$ . Notice that it is possible for a region that is weakly inferior to another region, to contain efficient vectors, and hence to be a Pareto occupied region.

**Definition 4.24** Two regions are incomparable with respect to each other if neither is superior nor weakly superior to the other.

**Definition 4.25** In a set of converged regions, Pareto occupied regions that are not weakly inferior to any other Pareto occupied region are called critical Pareto occupied regions (CPORs). An important property of a CPOR is the following: no vector that occupies a critical Pareto

occupied region can be dominated by any vector  $z \in Z$  that occupies any other region. The set of CPORs is denoted  $R_{CPOR}$ .

**Definition 4.26** A Pareto non-inferior region PNIR is any (converged) region that is weakly inferior to a Pareto occupied region, or is itself a Pareto occupied region. Figure 4.19 illustrates the concept of a PNIR.

Note that the set of Pareto occupied regions (PORs) contains the Pareto front. The set of PNIRs includes these, together with any additional regions (such as  $r_{25}$  in Figure 4.19) which, informally, fill in the gaps. That is, given any path  $(r_1, r_2, r_3, r_4, \dots, r_n)$  where  $r_{j+1}$  shares at least one corner with  $r_j$ , and in which  $r_1$  is inferior to a POR and  $r_n$  is superior to a POR, the path must include at least one PNIR.

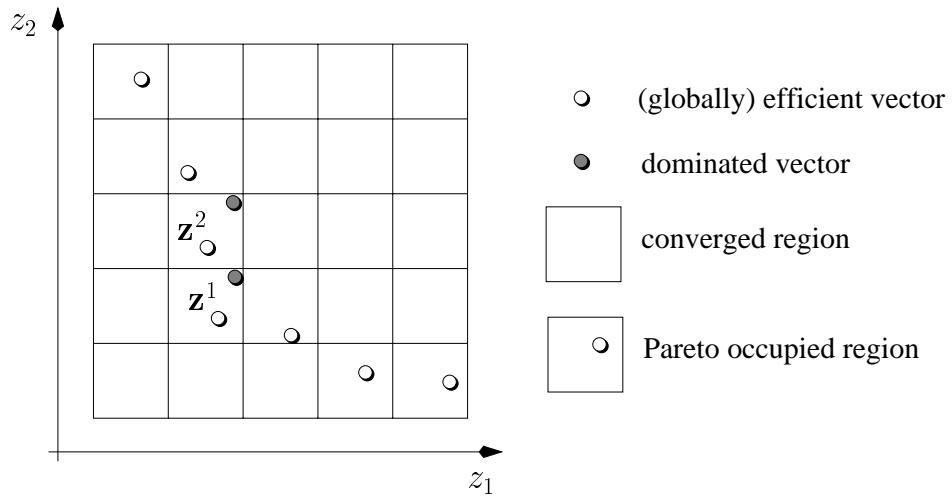


Figure 4.18: A set of (globally) efficient vectors (white points) within a space of converged regions is shown. Although both vectors  $\mathbf{z}^1$  and  $\mathbf{z}^2$  are nondominated, the region that  $\mathbf{z}^1$  occupies,  $r_{22}$ , is weakly superior to the region  $\mathbf{z}^2$  occupies,  $r_{23}$ . Because  $r_{23}$  is weakly inferior to  $r_{22}$ , there are points in  $r_{23}$  that are dominated by points in  $r_{22}$ , e.g., the grey point in  $r_{22}$  is dominated by  $\mathbf{z}_1$ . Region  $r_{22}$ , in contrast, is a critical Pareto occupied region because it is not weakly inferior to any other Pareto occupied region. This means that any vector in it cannot be dominated by any feasible vector in any other region. Consider the grey vector in  $r_{22}$ . Although it is dominated (by  $\mathbf{z}^1$ ), clearly it cannot be dominated by any vector not in  $r_{22}$ .

**Lemma 4.15** If a critical Pareto occupied region is occupied at time  $t_i$ , then it is occupied for all  $t > t_i$ .

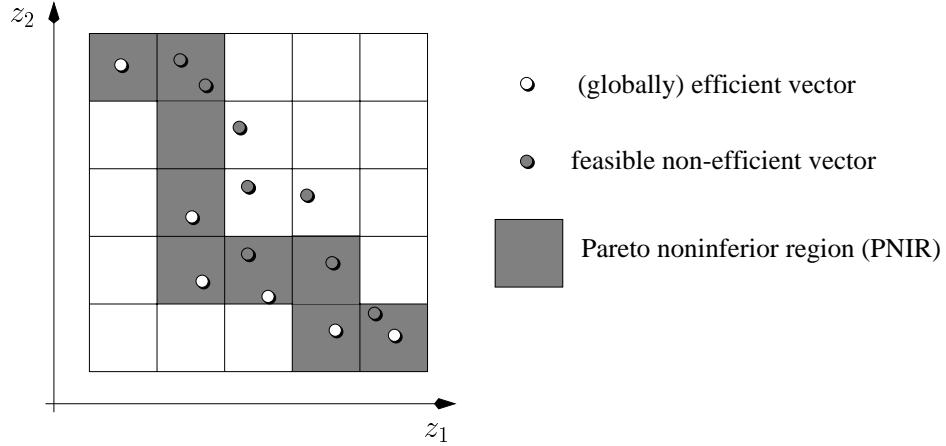


Figure 4.19: The feasible objective vectors within a converged set of regions is shown. The Pareto non-inferior regions (*PNIRs*) are shaded.

**Proof 4.18** Assume at time  $t_i$  a CPOR has a population of  $n \geq 1$  vectors  $\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3, \dots, \mathbf{z}^n$ , and at some time  $t_j > t_i$  all of them are removed from the archive.

Only Rules `Low_pop_region(t)` and `Domination(t)` and can remove vectors from the archive<sup>2</sup> However, `Low_pop_region(t)` can only remove one vector from a region. Therefore, if  $n > 1$  it cannot remove all the vectors.

On the other hand, if  $n = 1$ , then `Low_pop_region(t)` cannot remove the single vector in the CPOR because  $p_{nue}(r_{i_{CPOR}, t}) \not> 1$ . That is, the critical Pareto occupied region we are considering does not have a population (of non-uniquely extremal vectors) greater than 1. Therefore, either it is not one of the most crowded regions i.e. it is not in  $CR_t$  or there does not exist any region with a population greater than 1. In either case, `Low_pop_region(t)` cannot remove a vector from this critical Pareto occupied region.

Thus, Rule `Domination(t)` must remove the  $n$  vectors. But, by the definition of a critical Pareto occupied region, no vectors exist that can dominate any vector in a CPOR, except another vector in the same CPOR. Therefore `Domination(t)` can remove the  $n$  vectors only by replacing them with another vector in the same CPOR.

Therefore all cases contradict our original assumption that there is a time  $t_j > t_i$  such that no vectors in the archive occupy the CPOR.

**Theorem 4.4** The maximum number of mutually non-inferior regions in a  $K$  dimensional

<sup>2</sup>`Diverge(t)` cannot execute because we have already assumed that the region boundaries have converged.

vector space divided up into  $div$  equal divisions in each dimension, is  $div^K - (div - 1)^K$ .

**Proof 4.19** The proof of this theorem is given in Appendix B.

**Lemma 4.16** For all  $t$ , if  $arcsize > div^K - (div - 1)^K + 2K$  and an efficient point  $\mathbf{z}_t \in Z^*$  is generated by  $\text{Gen}(t)$ , then  $p(r_{\mathbf{i}_{\mathbf{z}_t}, t}) \geq 1$ . In other words, if a Pareto optimal point is generated at time  $t$ , its region will be occupied in the same time step, provided the archive capacity is greater than  $div^K - (div - 1)^K + 2K$ .

**Proof 4.20** Consider the time  $t$  when  $\mathbf{z}_t \in Z^*$  is generated by  $\text{Gen}(t)$ . Then following the execution of  $\text{Update\_boundaries}(t)$  (which may or may not alter the grid regions), exactly one of  $\text{Fill}(t)$ ,  $\text{Domination}(t)$ ,  $\text{Low\_pop\_region}(t)$  or  $\text{Steady\_state}(t)$  executes. If either  $\text{Fill}(t)$ ,  $\text{Domination}(t)$  or  $\text{Low\_pop\_region}(t)$  executes then  $\mathbf{z}_t$  is accepted and clearly  $p(r_{\mathbf{i}_{\mathbf{z}_t}, t}) \geq 1$ .

In order to prove that one of  $\text{Fill}(t)$ ,  $\text{Domination}(t)$  or  $\text{Low\_pop\_region}(t)$  always executes it is sufficient to prove that  $\text{Steady\_state}(t)$  does not execute. We show that  $\text{Steady\_state}(t)$  does not execute when  $p(r_{\mathbf{i}_{\mathbf{z}_t}, t-1}) = 0$  i.e. it can only execute when the  $\mathbf{z}_t$ 's region is already occupied, leaving it occupied.

Assume  $M_{t-1} = arcsize$ . Then we have  $M_{t-1} > div^K - (div - 1)^K + 2K$ . From Theorem 4.4, it follows that  $M_t$  occupies at most  $div^K - (div - 1)^K$  regions. Now, at most  $2K$  vectors are uniquely extremal. Therefore, it follows that  $\exists \mathbf{i}, p_{\text{nue}}(r_{\mathbf{i}, t}) > 1$ . Since  $M_{t-1} > 2K$ ,  $Z_{c,t} \neq \emptyset$ . Now, if  $p(r_{\mathbf{i}_{\mathbf{z}_t}, t-1}) = 0$  then  $\exists \mathbf{i}, p(r_{\mathbf{i}, t-1}) > p(r_{\mathbf{i}_{\mathbf{z}_t}, t-1})$ . So, we now have  $\exists \mathbf{i}, p_{\text{nue}}(r_{\mathbf{i}, t}) > 1 \wedge Z_{c,t} \neq \emptyset \wedge \exists \mathbf{i}, p(r_{\mathbf{i}, t-1}) > p(r_{\mathbf{i}_{\mathbf{z}_t}, t-1})$ , therefore  $\text{Steady\_state}(t)$  does not execute.

On the other hand, if  $M_{t-1} \neq arcsize$ ,  $\text{Steady\_state}(t)$  cannot execute either.

**Theorem 4.5**  $\exists t_m$  such that  $\forall t > t_m, \forall \mathbf{i}, t \in R_{CPOR}, p(r_{\mathbf{i}, t}) > 0$ , provided  $arcsize > div^K - (div - 1)^K + 2K$ . In other words, there is a time after which all of the critical Pareto occupied regions become constantly occupied, provided the archive is sized appropriately.

**Proof 4.21** From lemmata 4.15 and 4.16 we see that the number of critical Pareto occupied regions that are occupied by vector in the archive monotonically increases over time, provided  $arcsize > div^K - (div - 1)^K + 2K$ . Since the number of critical Pareto occupied regions is finite this implies that the set of occupied CPORs converges to  $R_{CPOR}$ .

**Theorem 4.6** If at time  $t_m$  the set of occupied CPORs has converged to  $R_{CPOR}$  then for all  $t > t_m$ , all vectors in  $M_t$  reside in a PNIR.

**Proof 4.22** *Regions that are not PNIRs are either weakly superior to, superior to, or inferior to, a critical Pareto occupied region. But for all  $t$ , no vector generated by  $\text{Gen}(t)$  can lie in a region that is superior or weakly superior to any critical Pareto occupied region, since they are all occupied for all future  $t$ . And, any vector generated by  $\text{Gen}(t)$  that is inferior to an occupied region will not be accepted (from the definition of domination and the rule  $\text{Steady\_state}(t)$ ).*

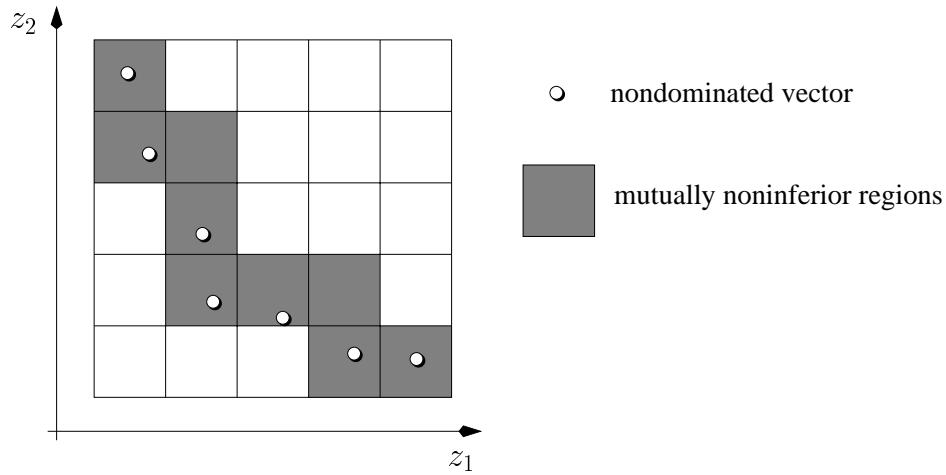


Figure 4.20: A set of mutually nondominated vectors is shown. All of them must lie within a set of regions that is mutually non-inferior.

The implications of Theorems 4.5 and 4.6 are that the vectors in the archive will become well distributed and close to the Pareto front in a well-defined way: after a certain number of iterations, a subset of them will always occupy all of the critical Pareto occupied regions, and the remainder will be in a *PNIR*. In order to guarantee achieving this, the archive should be given a capacity of  $1 + \text{div}^K - (\text{div} - 1)^K + 2K$ , at least.

In practice, however, the archive can probably be given a capacity smaller than this, provided it is larger than the number of critical Pareto occupied regions. We made the constraint that  $\text{arcsize} > \text{div}^K - (\text{div} - 1)^K + 2K$  in order to prove Lemma 4.16, which ensures that if an efficient point is generated, its region becomes or remains occupied. This was needed to prove that all *CPORs* will become constantly occupied. But we would generally expect all of the *CPORs* would become (constantly) occupied even without Lemma 4.16. This is because, as a result of rule  $\text{Domination}(t)$ , non-efficient vectors will generally be removed in favour of efficient ones that dominate them, over time. Thus, any set of mutually nondominated vectors that do not occupy all of the critical Pareto occupied regions, will *usually* give way to an archive in which one more critical Pareto occupied region is occupied. The reason this

is not *guaranteed* is because it may be the case that all of the unfound vectors in the critical Pareto occupied regions do not dominate any vector in the archive, for all future iterations. Assuming this unlikely event does not occur, there will still be a monotonic increase in the occupation of critical Pareto occupied regions. Thus, if the archive is greater than the number of critical Pareto occupied regions then eventually all of the critical Pareto occupied regions will become constantly occupied.

So, if a small chance of “sub-optimal” convergence can be tolerated then the requirement for *arcsize* goes down to the maximum number of critical Pareto occupied regions (plus the number of extremal Pareto occupied regions) that there are in a grid space.

**Conjecture 4.1** *The maximum number of critical Pareto occupied regions in an objective space of  $K$  dimensions and divided into  $div$  divisions in each dimension is given by:*

$$v_s = \sum_{j=0}^{(s-K)/div} \left[ (-1)^j \cdot \frac{K!}{(K-j)!K!} \cdot \frac{(K/2.(div+1) - div.j - 1)!}{(K-1)!(K/2.(div+1) - div.j - K)!} \right] \quad (4.3)$$

when  $s$  is set to  $\lfloor K/2(div+1) \rfloor$ .

A proof of this conjecture is not provided but some supporting argument is given. A critical Pareto occupied region (*CPOR*) cannot have coordinates that put it on the same one dimensional ‘row’ as any other *CPOR*, otherwise one of them would be weakly superior to the other. From this we can see that in any list of critical Pareto optimal regions, no pair of regions can have an identical set of any  $K - 1$  coordinates. In addition, all the regions must be non-inferior so no two vectors should share the same root region. Such an arrangement of regions can be achieved if the sum of all the coordinates of each region is equal to a constant value. This ensures that the above constraints are satisfied because any two regions that are different will be different in  $K - 1$  coordinates if their sum is equal, and any two regions with the same root region would surely have a different coordinate sum. The number of regions that have the same coordinate sum is given by equation 4.3. This equation actually gives the number of ways there are of achieving a total of  $s$  from  $K$  dice, each with  $div$  faces. The equation was obtained from [Rob01], and a derivation is given in Appendix C. When the sum is set to  $\lfloor K/2(div+1) \rfloor$  this number appears to be maximized, although no proof of this has yet been found.

In summary, the adaptive grid archiving algorithm is guaranteed to:

- store and maintain points at the extremes of all objectives (Theorems 4.3 and A.1);
- store and maintain points in all of the critical Pareto occupied regions (Theorem 4.5);

- and distribute its remaining points amongst the Pareto non-inferior regions (Theorem 4.6),

provided the upper boundaries of the grid converge,  $arcsize > div^K - (div - 1)^K + 2K$ , the generating function generates all points with positive probability, and the search space is finite.

Table 4.1 gives the required size of the archive for different values of  $div$  and  $K$ , for the case where convergence is *guaranteed* (3rd column), and the case where it is only probable (4th column). Clearly, in the two-dimensional case, the required archive size to guarantee convergence is small, even with 64 divisions per objective. As the number of objectives increases, the required archive sizes for guaranteed convergence rapidly increase to numbers that are larger than usually used as the population in an EA. However, with 8 objectives and 4 divisions per objective,  $arcsize > 8092$ , a manageable number, is large enough to cover the *CPORs*, if our conjecture 4.1 is true.

The formal description of the AGA strategy given in this section was required to evaluate its convergence properties. In the next section, we consider how the archiving rules may be actually implemented to give an archiving algorithm that is computationally efficient.

## 4.4 Efficient implementation of an adaptive grid archiving algorithm

In the last section, several alternative archiving strategies for use in the archiving hillclimber were proposed and analyzed. The adaptive grid archiving (AGA) strategy was shown to have several advantages over the other proposed strategies: it preserves a bounded archive, it is not efficiency preserving so that no region of the Pareto front is ever unobtainable, and it works adaptively, without the need for setting any critical parameters; the number of divisions of the objective space in each objective must be chosen but this is an independent choice, not affected by any properties of the objective space. Furthermore, we showed that under certain conditions (see Appendix A), this strategy also converges to a set spanning the extremes of the Pareto front, and provided the archive can be given a large enough capacity, it distributes the other vectors so that a certain set of superior regions — the critical Pareto occupied regions — are always occupied, and the remaining vectors distribute themselves between grid regions that are weakly inferior to these.

In this section, we propose an efficient implementation of an adaptive grid archiving algorithm

| $K$      | $div$     | $div^K - (div - 1)^K + 2K$ | $v_{\lfloor K/2(div+1) \rfloor}$ |
|----------|-----------|----------------------------|----------------------------------|
| 2        | 2         | 7                          | 2                                |
| 2        | 4         | 11                         | 4                                |
| 2        | 8         | 29                         | 8                                |
| 2        | 16        | 35                         | 16                               |
| 2        | 32        | 67                         | 32                               |
| <b>2</b> | <b>64</b> | <b>131</b>                 | <b>64</b>                        |
| 3        | 2         | 13                         | 3                                |
| 3        | 4         | 43                         | 12                               |
| <b>3</b> | <b>8</b>  | <b>175</b>                 | <b>48</b>                        |
| <b>3</b> | <b>16</b> | <b>727</b>                 | <b>192</b>                       |
| 4        | 2         | 23                         | 6                                |
| <b>4</b> | <b>4</b>  | <b>183</b>                 | <b>44</b>                        |
| <b>4</b> | <b>8</b>  | <b>1703</b>                | <b>344</b>                       |
| 4        | 16        | 14919                      | 2736                             |
| 8        | 2         | 272                        | 70                               |
| 8        | 4         | 58991                      | 8092                             |
| 8        | 8         | 11012431                   | 1012664                          |

Table 4.1: The required size of the archive for various values of  $div$  and  $K$ . The third column is the required size if convergence to all the critical Pareto occupied regions must be guaranteed. The fourth column is the required size to allow storage of vectors occupying the maximum number of critical Pareto occupied regions, but without guaranteeing that convergence to this set of regions will occur. The bold lines are the numbers of objectives and divisions chosen for problems tackled in this thesis.

based on the AGA strategy. The archiving algorithm, as specified here, drops the requirement, seen in the AGA, that ‘crowded regions’ (from which solutions may be removed) must have a population greater than one. This could easily be included if guaranteed convergence is required but we drop this requirement to make the algorithm slightly simpler. In other respects the archiving algorithm is the same as the AGA strategy, except for one more restriction.

Efficient implementation of the adaptive grid archiving algorithm is achieved by using a quadtree encoding of the grid regions (Figure 4.22). In order to use this approach, the number of divisions of each objective range,  $div$ , is constrained to be a value of  $2^l$ , where  $l \in \mathbb{Z}^+$  is the user-selected subdivision parameter, controlling  $div$ . Now, when each new vector is generated, its grid region is calculated using iterated subdivision of the objective space, and recorded using a quadtree encoding. At each step, the relevant bit in the encoding is set, if the vector lies in the larger half of the division currently being checked, in the

---

**Algorithm:** *Quadtree\_encode*


---

**z** is the vector whose region is to be located  
 The two-dimensional matrix  $Q$  is the required quadtree coding of **z**'s region, and every element of it,  $q_{k,j}$  is initially set to zero  
 $ub_k$  and  $lb_k$  are the values of the upper and lower boundary of the grid in objective  $k$ , respectively

---

```

foreach ( $k \in 1..K$ )
     $subrange_k \leftarrow ub_k - lb_k$ 
     $sublb_k \leftarrow lb_k$ 
    foreach ( $j \in 1..l$ )
         $subhalf_k \leftarrow sublb_k + subrange_k/2$ 
        if ( $z_k > subhalf_k$ )
             $q_{k,j} \leftarrow 1$ 
             $sublb_k \leftarrow subhalf_k$ 
         $subrange_k \leftarrow subrange_k/2$ 
    
```

---

Figure 4.21: Algorithm for calculating the region occupied by an objective vector **z**, within a grid with known upper and lower boundaries.

objective currently being checked. The number of steps required to encode a vector's region is thus  $l.K$ . An algorithm for calculating the region occupied by an objective vector is given in Figure 4.21.

The overall time complexity of the adaptive grid archiving algorithm, in terms of the number of comparisons that must be made, may be calculated in terms of the number of solutions in the archive,  $|M|$ , the number of subdivision levels being used,  $l$ , and the number of objectives of the problem,  $K$ . We have seen that finding the grid region of a single objective vector requires  $l.K$  comparisons. Once this is found for a vector in the archive, it is stored with the vector for future lookup. Only if the grid boundaries are changed, must it be recalculated. The population of each region is also stored in an array so that this can be retrieved in  $O(1)$  time.

If the functions that must be performed are ordered in an efficient way then very few operations must be performed to keep the adaptive grid updated and carry out the crowding procedure. An efficient implementation of the archiving algorithm is shown in Figure 4.23, giving the number of comparisons and significant operations for each function of the algo-

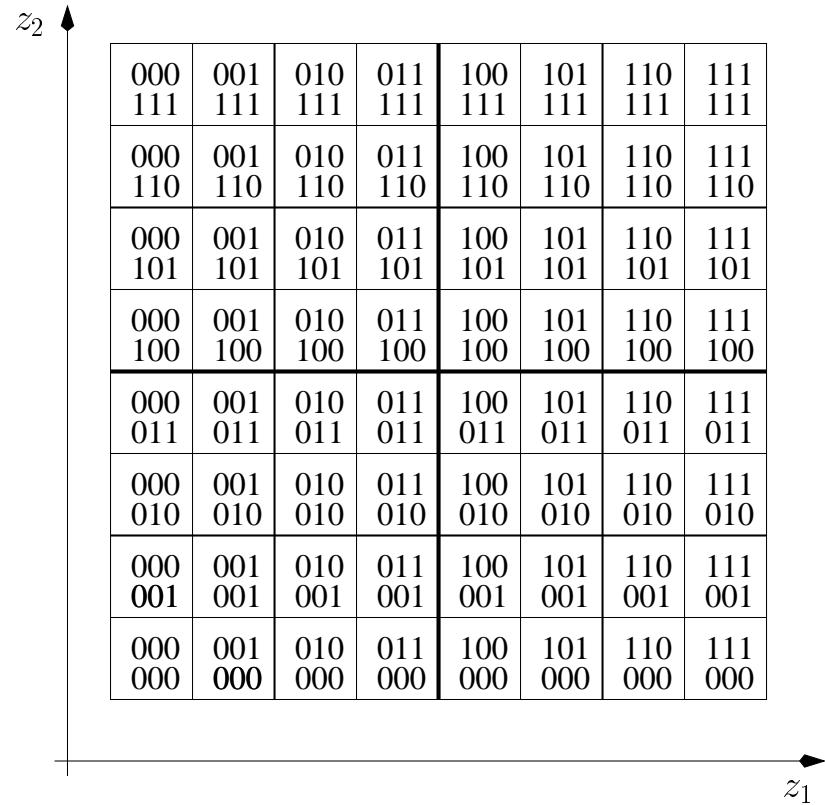


Figure 4.22: A grid for a 2-objective problem with the subdivision parameter  $l = 3$ , giving  $div = 2^l = 8$  grid divisions in each objective. Each region is coded using a quadtree encoding, in which the  $k, j$ th bit is set in a  $K \times l$  binary matrix if the region is in the larger half of the  $j$ th bisection of the grid in the  $k$ th objective. The matrices for each region are shown.

rithm. Using this implementation, if  $\mathbf{z}_t$  is not dominated by any members of the archive and is not outside the current grid boundary then only  $O(|M_{t-1}| + l.K)$  operations are needed altogether, worst case. If it is outside the boundary then updating the boundaries and grid populations takes an additional  $O(|M_{t-1}|.l.K)$  significant operations. Since  $l$  and  $K$  will typically be very small compared to the size of the archive, the overall time complexity can be written as  $O(|M_t|)$  per iteration. Any time  $\mathbf{z}_t > M_{t-1}$ , or the archive is not full, or  $\mathbf{z}_t$  dominates just one or two vectors in  $M_{t-1}$  then the actual number of operations is significantly less than this, however. In contrast, niching methods require  $O(N^2)$  comparisons, for a population of size  $N$ , to calculate niche counts.

```

if  $M_{t-1} \not\prec \mathbf{z}_t$ 
    if  $\mathbf{z}_t$  is outside current boundaries of the grid
        -update boundaries      (= 2.K comparisons)
        -recalculate grid populations      (= |M_{t-1}|.l.K comparisons)
        -calculate grid region of  $\mathbf{z}_t$       (= l.K comparisons)
    if  $\mathbf{z}_t < M_{t-1}$ 
        -increment population of  $\mathbf{z}_t$ 's region and decrement population of all
            dominated vectors' regions      (= O(|M_{t-1}|) operations (worst case))
    else if  $\mathbf{z}_t \sim M_{t-1}$  and archive is full and  $\mathbf{z}_t$  in less crowded region than some vector in  $M$ 
        -find all vectors in more crowded regions      (= O(|M_{t-1}|) operations)
        -increment population of  $\mathbf{z}_t$ 's region and decrement
            population of discarded vector's population      (= O(1) operations)
    else if  $\mathbf{z}_t \sim M_{t-1}$  and archive is not full
        -add  $\mathbf{z}_t$  to the archive and increment population of  $\mathbf{z}_t$ 's region      (= O(1) operations)

```

Figure 4.23: Time complexity of operations in the adaptive grid archiving algorithm.

## 4.5 Enhancing the acceptance function

In the last sections, an efficient and effective means of maintaining a bounded archive of the best solutions found has been proposed and analyzed. By incorporating the adaptive grid archiving strategy in the Reduce function of the archiving hillclimber, an algorithm that satisfies the requirements put forward earlier is obtained. However, in order to accelerate the discovery of diverse and high quality solutions it may also be beneficial to bias the acceptance function to favour (a) solutions in less crowded regions, and (b) solutions that dominate one or more members of the current archive.

In general, some sort of neighbourhood function will be used to alter the current solution to generate the mutant. In other words, a mutant will usually be ‘close’ in decision space to its

parent solution. Assuming that close in decision space implies close in objective space (which we do, implicitly, when we use any sort of selection process and neighbourhood function), then a mutant of a solution in an uncrowded region will also be more likely to be in an uncrowded region of objective space. Thus, it may be beneficial to accept these more isolated solutions over the current solution, because they will encourage the further exploration of under-populated regions of the objective space.

Similarly, a mutant that dominates one or more members of the archive is, in a sense, stronger than a solution that is just nondominated with respect to the archive (as is the current solution). Therefore, such a mutant might be considered preferable to the current solution as a place from which to explore further, and should thus be accepted.

In order to bias the acceptance function towards solutions in uncrowded regions, the adaptive grid crowding algorithm, used as the archiving strategy, can be re-utilized. This could be achieved by accepting the mutant  $\mathbf{x}'$  only if its image in objective space is in a less crowded region than the image of  $\mathbf{x}$ .

In order to bias the acceptance function towards solutions that dominate one or more members of the archive, mutants that dominate the archive can be accepted independently of whether or not they occupy a less crowded region than the current solution.

In order to incorporate this biasing of the acceptance function into the archiving hillclimber algorithm given in Figure 4.3, the functions `Reduce` and `Inferior` must be combined, because the replacement of  $\mathbf{x}$  by  $\mathbf{x}'$  depends upon knowing the region that  $\mathbf{z}'$  occupies, and its population, and whether  $\mathbf{z}'$  actually dominates any member of the archive. The (1+1)-PAES algorithm described in the next section orders these functions so that biasing of the acceptance function and the adaptive grid archiving strategy are achieved with minimal computational overhead.

## 4.6 (1+1)-PAES

The complete (1+1)-PAES algorithm is shown in Figure 4.24. Like a hillclimber, it has a population of 1 (the current solution), it generates a mutant of this, then selects the “best” from the pair to become the current solution of the next iteration. This is known as a (1+1) selection strategy in ESSs [Rec65], and gives rise to the name (1+1)-PAES.

In (1+1)-PAES, the acceptance function is biased towards accepting mutants which dominate the archive, and mutants in less crowded regions of the adaptive grid. It also incorporates an efficient implementation of the adaptive grid archiving algorithm described earlier, although

the bookkeeping operations to maintain the grid are not shown. The algorithm may be made still more efficient if a domination-free tree [SS96a, SS96b] were used and maintained, to store the vectors in the archive<sup>3</sup>. However, to date, this has not been implemented.

## 4.7 The time complexity of (1+1)-PAES

The time complexity of (1+1)-PAES can be reduced to the complexity of the acceptance/rejection of the mutant solution, and of the updating of the archive. In these processes, PAES requires  $K$  comparisons to compare the candidate solution with the current solution and a further  $|M_{t-1}|.K$  comparisons (in the worst case) to compare the current solution with the archive, where  $|M_{t-1}|$  is the size of the archive of the previous iteration and  $K$  is the number of objectives in the problem. It requires  $l.K$  to find the candidate's grid location. A further  $2.K$  comparisons are required to update the grid ranges and another  $|M_t|.l.K$  comparisons if the grid ranges have changed and the archive grid region populations require updating. The complexity of finding a solution in a crowded region to be discarded from the archive is also  $O(|M_{t-1}|)$ , as is discarding all dominated members of the archive. Overall, this gives (1+1)-PAES a time complexity of  $O(\text{arcsize})$ , per iteration, where *arcsize* is the capacity of the archive. This worst-case time complexity is equivalent to the  $O(n^2)$  time complexity of many generational MOEAs.

However, the best and average case number of operations used by PAES is significantly different from the worst case outlined above. It requires only  $K$  comparisons to ascertain that the candidate solution is dominated by the current solution and in this case no further comparisons are required in that generation. Similarly, if the candidate is dominated by anything in the archive, no updating of the grid ranges or region populations is necessary. In PAES, the latter case occurs frequently since the archive represents a diverse sample of the best solutions ever found. Also, in many cases, the archive is not full, i.e.  $|M_t| < \text{arcsize}$ , so that the number of comparisons to accept or discard a solution is relatively small. If large archive sizes were needed then it would be worth implementing the archive as a domination-free tree so that the number of comparisons to determine the candidate solution's dominance relationship to the archive would be reduced further.

---

<sup>3</sup>In a domination-free tree, nondominated vectors are stored in a quadtree data-structure, based on relations between the components of all the vectors, that can reduce the number of comparisons required to identify if a new vector is nondominated. The method is particularly effective when the number of objectives is large.

---

**Algorithm:** (1+1)-PAES

---

*Data:*

$M_t$  is the archive of solution vectors discovered

$\mathbf{x}$  is the current solution vector

$\mathbf{x}'$  is the mutant solution vector

*Functions:*

$\text{Terminate}(t, M_t)$  returns TRUE if a termination condition based on the archive or number of iterations is satisfied

$\text{Init}()$  returns a solution vector in  $\mathbf{x} \in X$

$\text{Mutate}(\mathbf{x})$  returns a neighbour of  $\mathbf{x}$

$\text{Grpop}(\mathbf{x})$  returns the population of  $\mathbf{x}$ 's grid region

$\text{ND}(X)$  returns the nondominated vectors from a set  $X$

---

```
t  $\leftarrow$  0                                     /* Initialization */
 $M_t \leftarrow \emptyset$ 
 $\mathbf{x} \in X \leftarrow \text{Init}()$ 
t  $\leftarrow$  t + 1
 $M_t \leftarrow M_{t-1} \cup \{\mathbf{x}\}$ 
while ( $\text{Terminate}(t, M_t) \neq \text{TRUE}$ )           /* Main Loop */
  t  $\leftarrow$  t + 1
   $\mathbf{x}' \in X \leftarrow \text{Mutate}(\mathbf{x})$ 
  if ( $\mathbf{x}' \prec \mathbf{x}$ )
     $\mathbf{x} \leftarrow \mathbf{x}'$ 
     $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1})$ 
  else if ( $\mathbf{x}' \not\prec M_{t-1}$ )
    if ( $\mathbf{x}' \prec M_{t-1}$ )
       $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1})$ 
       $\mathbf{x} \leftarrow \mathbf{x}'$ 
    else if ( $\mathbf{x}' \sim M_{t-1}$ )
      if ( $|M_{t-1}| < \text{arcsize}$ )
         $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1}$ 
      else if ( $\mathbf{x}'$  increases the extent of the grid)
         $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$  where  $\mathbf{x}^{cr}$  is one
        randomly selected solution from the
        set  $\{\mathbf{x}^i \in M_{t-1} \mid \forall \mathbf{x}^j \in M_{t-1}, \text{Grpop}(\mathbf{x}^i) \geq \text{Grpop}(\mathbf{x}^j)\}$ 
    else if ( $\exists \mathbf{x}^i \in M_{t-1}$  such that  $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x}^i)$ )
       $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$ 
      if ( $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x})$ )
         $\mathbf{x} \leftarrow \mathbf{x}'$ 
return  $M_{t_{final}}$                                      /* Termination */
```

---

Figure 4.24: Pseudocode for (1+1)-PAES.

## 4.8 Other variants of PAES

The (1+1)-PAES algorithm can be extended in a number of different ways, bridging the gap between it and the more sophisticated, and computationally costly, MOEAs seen in the literature. In Section 5.2.3 of the next chapter, we propose two population-based variants of PAES. These, it was hoped, might be able to reduce the likelihood of PAES getting stuck in local optima. However, as the experimental results in Section 5.2.5 show, these variants did not generally perform as well as (1+1)-PAES, on a number of test functions we used (although they did in some isolated cases). Here we present a number of other alternative variants of the basic (1+1)-PAES algorithm.

### 4.8.1 Multi-start PAES

In a multi-start hillclimber, the overall best solution from each mini-run (restart) is stored and updated so that at the end of the run, it can be returned as the best solution found. The equivalent of this in PAES would be to store and update the best combined nondominated front found from all the mini-runs. Ideally, this set should be bounded and should be as diverse a set as possible, as in the normal PAES archive.

Thus, the simplest way to store solutions effectively in a multi-start version of PAES, is to have two archives: a global archive that maintains a diverse sample of the nondominated solutions discovered, independently of which of the mini-runs the solutions were generated in. A second, local archive is needed because a mini-run of PAES will not be independent if it interacts with the global archive, which may have solutions in it from previous mini-runs. The local archive is voided for each mini-run, so that PAES starts from ‘zero’ each time. The global archive is never voided. At the end of the run the global archive is returned. Both the local and global archive are updated at each iteration, according to a similar archiving strategy.

To control when restarts occur, a multi-start hillclimber usually estimates the rate of change of the evaluation of the current solution, restarting when this rate falls below a threshold. The estimate may be achieved by simply using the number of consecutive iterations that have the same evaluation, or the number of iterations between ‘moves’, as a threshold to initiate a restart. This simple method can be used in a multi-start version of PAES, as shown in the pseudocode given in Figure 4.25. Some investigation of the multi-start PAES algorithm has given promising results (not reported in this thesis) and it appears that it can be an even more challenging algorithm than (1+1)-PAES for baselining MOEAs against, on some

---

**Algorithm:** multi-start PAES

---

*Data:*

$M_t$  is the (local) archive of solution vectors discovered  
 $G_t$  is the global archive of solution vectors discovered  
 $\mathbf{x}$  is the current solution vector;  $\mathbf{x}'$  is the mutant solution vector  
 $it$  records the number of iterations between accepting a candidate  
 $max\_it$  is a threshold for  $it$  that causes a restart

*Functions:*

$\text{Terminate}(t, G_t)$  returns TRUE if a termination condition based on the global archive or number of iterations is satisfied  
 $\text{Init}()$ ,  $\text{Mutate}(\mathbf{x})$ ,  $\text{Grpop}(\mathbf{x})$ ,  $\text{ND}(X)$  all as previously defined for the (1+1)-PAES algorithm  
 $\text{Update}(G_t, \mathbf{x})$  updates  $G_t$  with  $\mathbf{x}$  according to the adaptive grid archiving strategy

---

```

 $t \leftarrow 0$                                      /* Initialization */
 $G_t \leftarrow \emptyset$ 
while ( $\text{Terminate}(t, M_t) \neq \text{TRUE}$ )           /* Main Loop */
   $M_t \leftarrow \emptyset$ 
   $\mathbf{x} \in X \leftarrow \text{Init}()$ 
   $t \leftarrow t + 1$ 
   $M_t \leftarrow M_{t-1} \cup \{\mathbf{x}\}; \text{Update}(G_t, \mathbf{x})$ 
   $it \leftarrow 0$ 
  while ( $(it < max\_it) \wedge (\text{Terminate}(t, M_t) \neq \text{TRUE})$ ) /* Mini loop */
     $t \leftarrow t + 1$ 
     $\mathbf{x}' \in X \leftarrow \text{Mutate}(\mathbf{x})$ 
    if ( $\mathbf{x}' < \mathbf{x}$ )
       $\mathbf{x} \leftarrow \mathbf{x}'; it \leftarrow 0$ 
       $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1}); \text{Update}(G_t, \mathbf{x}')$ 
    else if ( $\mathbf{x}' \not> M_{t-1}$ )
      if ( $\mathbf{x}' < M_{t-1}$ )
         $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1}); \text{Update}(G_t, \mathbf{x}')$ 
         $\mathbf{x} \leftarrow \mathbf{x}'; it \leftarrow 0$ 
      else if ( $\mathbf{x}' \sim M_{t-1}$ )
        if ( $|M_{t-1}| < arcsize$ )
           $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1}$ 
        else if ( $\mathbf{x}'$  increases the extent of the local archive's grid)
           $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$  where  $\mathbf{x}^{cr}$  is one
          randomly selected solution from the
          set  $\{\mathbf{x}^i \in M_{t-1} \mid \forall \mathbf{x}^j \in M_{t-1}, \text{Grpop}(\mathbf{x}^i) \geq \text{Grpop}(\mathbf{x}^j)\}$ 
        else if ( $\exists \mathbf{x}^i \in M_{t-1}$  such that  $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x}^i)$ )
           $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$ 
        if ( $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x})$ )
           $\mathbf{x} \leftarrow \mathbf{x}'; it \leftarrow 0$ 
         $\text{Update}(G_t, \mathbf{x}')$ 
    return  $G_{t_{final}}$                                      /* Termination */

```

---

Figure 4.25: Pseudocode for multi-start PAES.

problems.

### 4.8.2 Thresholding and annealing PAES

In thresholding algorithms, candidate solutions are accepted if they are not worse than the current solution by a certain threshold; the rule is deterministic, and the threshold may be constant or decreasing according to a schedule. Simulated annealing algorithms accept candidate solutions probabilistically, based on a distribution (usually the Boltzmann distribution), a ‘temperature’ parameter, and the difference between the current and candidate solutions’ evaluations. A cooling schedule reduces the ‘temperature’ and hence the probability of accepting degrading moves. In general, simulated annealing and other probabilistic strategies have better convergence properties, and work better in practice, than deterministic thresholding strategies, but both allow escape from local optima.

Unfortunately, neither thresholding nor annealing appears to be easily adaptable to a variant of PAES. There are two reasons for this: first, measurable differences between solutions are never utilized in PAES — only the dominance relation is used; and second, if the algorithm were to sometimes accept dominated solutions, and these were simultaneously accepted into the archive, then the archiving strategy would be grossly affected, because it is based on keeping an entirely nondominated set of solutions at every stage, and much of its efficiency comes from this simplification. It is possible to overcome these problems, to some extent, however. Let us first consider the acceptance function only, and then we will consider the archiving strategy.

In order to preserve the use of the dominance relation from PAES within a thresholding algorithm, a threshold could be set in terms of the ‘number’ of archive members that dominate the candidate solution, or in terms of the number of different grid regions that are occupied by solutions that dominate the current solution. These calculations do increase the mean number of comparisons that are necessary between the candidate solution and the archive, however. Another alternative, that is non-deterministic, is to use a random *sample* of the archive for comparison with the candidate. This rule would first check if the candidate dominates any member of the entire archive. If it does, it is accepted in the normal way. Otherwise it is checked against the smaller sample of the archive. If the candidate is nondominated with respect to this *sample* then it can be accepted, other things allowing it. Obviously, the size of the sample can be adjusted easily, allowing for a cooling schedule to be used. The sampling can be made more or less evenly distributed in objective space if occupied regions are selected with uniform probability, and then solutions are selected from within the selected region (without replacement), as is done in PESA-II [CJKO01]. This approach

does not increase the number of comparisons as much as counting the number of solutions or regions dominating the candidate, and is in keeping with the main principles of PAES.

The archiving strategy in (1+1)-PAES always accepts a candidate solution into the archive if it is accepted as the new current solution. However, this is not necessary. If candidate solutions that are dominated by the archive are accepted as the new current solution then they need not be accepted into the archive. This makes the archiving strategy simpler, and also ensures that the convergence properties of the archiving strategy still apply. Obviously, once a dominated solution is accepted as the current solution, a new candidate solution may be accepted based solely on its dominating the current, even if it is dominated by members of the archive. However, this is fine, and in keeping with how simulated annealing algorithms work. After a ‘bad’ solution is accepted, it becomes easier to accept other ‘bad’ solutions. Eventually, though, a good solution is found, and stronger selection pressure returns.

So, if the normal archiving strategy is maintained, and the acceptance function is based on sampling the archive, as outlined above, then one obtains the annealing PAES algorithm shown in Figure 4.26. Some preliminary experiments with this algorithm have been carried out but many more are needed to establish how the cooling schedule should be determined, and whether the algorithm outperforms (1+1)-PAES significantly on any problems.

### 4.8.3 Tabu search PAES

Tabu search algorithms generally use a strategy whereby the best move from a set of potential moves is chosen, irrespective of whether this represents a degradation from the current solution’s evaluation. This can be seen as  $(1, \lambda)$  selection, in the terminology of ESs. Using this kind of selection allows local optima to be traversed, which would be advantageous in PAES. Tabu search algorithms also incorporate the use of memory to avoid returning to recently or frequently visited solutions. (1+1)-PAES does not have this facility, at present, and in many search landscapes this may be beneficial.

In general, tabu search algorithms incorporate the following features:

1.  $(1, \lambda)$  selection, allowing traversal of local optima;
2. A tabu list that restricts the set of possible moves to those that have not been recently or frequently performed;
3. Additional mechanisms for guiding (intensifying or diversifying) the search.

The first two of these features would be beneficial in PAES, as it suffers from the potential

---

**Algorithm:** Annealing-PAES

---

*Data:*

$M_t$  is the archive of solution vectors discovered  
 $\mathbf{x}$  is the current solution vector  
 $\mathbf{x}'$  is the mutant solution vector  
 $T$  is the temperature parameter and  $T_{start}$  is its initial value

*Functions:*

$\text{Terminate}(t, M_t)$  returns TRUE if a termination condition based on the archive or number of iterations is satisfied  
 $\text{Init}()$  returns a solution vector in  $\mathbf{x} \in X$   
 $\text{Mutate}(\mathbf{x})$  returns a neighbour of  $\mathbf{x}$   
 $\text{Grpop}(\mathbf{x})$  returns the population of  $\mathbf{x}$ 's grid region  
 $\text{ND}(X)$  returns the nondominated vectors from a set  $X$   
 $\text{Sm}(M_t, T)$  takes a sample of the solutions in  $M_t$ . The size of the sample depends on the temperature,  $T$   
 $\text{Cool}(T, M_t)$  returns a new value of  $T$  depending on the current value of  $T$ , and possibly the state of  $M_t$

---

```

 $t \leftarrow 0$                                      /* Initialization */
 $M_t \leftarrow \emptyset$ 
 $\mathbf{x} \in X \leftarrow \text{Init}()$ 
 $t \leftarrow t + 1$ 
 $T \leftarrow T_{start}$ 
 $M_t \leftarrow M_{t-1} \cup \{\mathbf{x}\}$ 
while ( $\text{Terminate}(t, M_t) \neq \text{TRUE}$ )           /* Main Loop */
   $t \leftarrow t + 1$ 
   $\mathbf{x}' \in X \leftarrow \text{Mutate}(\mathbf{x})$ 
  if ( $\mathbf{x}' < \mathbf{x}$ )
     $\mathbf{x} \leftarrow \mathbf{x}'$ 
     $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1})$ 
  else
    if ( $\mathbf{x}' < M_{t-1}$ )
       $M_t \leftarrow \text{ND}(\{\mathbf{x}'\} \cup M_{t-1})$ 
       $\mathbf{x} \leftarrow \mathbf{x}'$ 
    else if ( $\mathbf{x}' \sim \text{Sm}(M_{t-1}), T$ )
      if ( $|M_{t-1}| < \text{arcsize}$ )
         $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1}$ 
      else if ( $\mathbf{x}'$  increases the extent of the grid)
         $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$  where  $\mathbf{x}^{cr}$  is one
        randomly selected solution from the
        set  $\{\mathbf{x}^i \in M_{t-1} \mid \forall \mathbf{x}^j \in M_{t-1}, \text{Grpop}(\mathbf{x}^i) \geq \text{Grpop}(\mathbf{x}^j)\}$ 
    else if ( $\exists \mathbf{x}^i \in M_{t-1}$  such that  $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x}^i)$ )
       $M_t \leftarrow \{\mathbf{x}'\} \cup M_{t-1} \setminus \{\mathbf{x}^{cr}\}$ 
    if ( $\text{Grpop}(\mathbf{x}') < \text{Grpop}(\mathbf{x})$ )
       $\mathbf{x} \leftarrow \mathbf{x}'$ 
   $T \leftarrow \text{Cool}(T, M_t)$ 
return  $M(t_{final})$                                      /* Termination */

```

---

Figure 4.26: Pseudocode for Annealing-PAES.

to get stuck in local optima, and also to cycle (e.g. on plateaus). The third item, which is really optional in tabu search, may be beneficial for use in PAES also, but its expediency depends upon finding good rules for the problem at hand.

Incorporating these features in PAES is not straightforward, however. This is because PAES was designed for a (1+1) selection strategy. In Section 5.2.5, some of the drawbacks of the population-based variants of PAES are explained. Any tabu search-based variant of PAES would have to avoid the deficiencies of the proposed selection and archiving strategies put forward for the population-based variants of PAES. Along these lines, we sketch the elements needed for selection and archiving to be effectively performed.

In order for the set of neighbours of the current solution,  $N$ , to be evaluated with respect to the current archive,  $M_{t-1}$ , the union of the two sets can be taken and nondominated sorting [SD93] can be applied. The best ‘neighbours’ are then those from the set  $N$  for which there does not exist any other element of  $N$  in a higher equivalence class in  $M_{t-1} \cup N$ . From this set of best neighbours, one of them from amongst the least crowded regions is selected as the overall best, to replace the current solution.

Archiving should allow that any of the set  $N$  in the first equivalence class of  $M_{t-1} \cup N$  are given the opportunity to enter  $M_{t-1}$ . To achieve this, a temporary store  $Temp$  of solutions can be used and the following order of operations can be carried out. 1.  $Temp \leftarrow ND(M_{t-1} \cup N)$ . 2. If  $|Temp| > arcsize$  then repeatedly select a solution from the most crowded region(s) to remove, excluding solutions that are uniquely extremal on one or more objectives, until  $|Temp| = arcsize$ . 3.  $M_t \leftarrow Temp$ .

These selection and archiving strategies allow for an implementation of a tabu search variant of PAES. Its main relation to (1+1)-PAES is its use of the adaptive grid for performing crowding, which ensures that the same convergence guarantees exist as for (1+1)-PAES. However, it is also more complicated than PAES because of the need to perform nondominated sorting on the set of solutions in order to rank them. It is unclear whether this added complication in selection would be balanced by improved ability to escape local optima, and to avoid cycling. No implementation of Tabu-PAES has yet been implemented, but pseudocode for it is given in Figure 4.27.

## 4.9 Summary

Hillclimbers are good, general-purpose search methods for optimization. They are simple to use, with few controlling parameters, they use little computation beyond the evaluation

---

**Algorithm:** Tabu-PAES

---

*Data:*

$M_t$  is the archive of solution vectors discovered  
 $\mathbf{x}$  is the current solution vector  
 $\lambda$  is the number of neighbour solutions generated at each iteration  
 $N$  is the set of neighbour solutions of  $\mathbf{x}$   
 $B$  is the set of best solutions from among  $N$   
 $Temp$  is a temporary set used for updating the archive with the new solutions  
 $tabu\_l$  is the set of tabu lists  
 $attr\_l$  is the set of attribute lists

*Functions:*

$\text{Terminate}(t, M_t)$  returns TRUE if a termination condition based on the archive or number of iterations is satisfied  
 $\text{Init}()$  returns a solution vector in  $\mathbf{x} \in X$   
 $\text{Mutate}(\mathbf{x})$  returns a neighbour of  $\mathbf{x}$   
 $\text{ND}(X)$  returns the nondominated vectors from a set  $X$   
 $\text{Tabu}(N, tabu\_l, attr\_l)$  returns the subset of  $N$  allowed by the current tabu lists and attribute lists  
 $\text{Eq\_best}(X, Y)$  returns the set of solutions in  $Y$  which are in the best nondominated sorting equivalence class of  $X$   
 $\text{Low\_pop}(X, Y)$  returns the solutions in  $Y$  the from the least crowded regions in  $X$   
 $\text{Rand}(X)$  returns a uniformly randomly selected solution from  $X$   
 $\text{Most\_crowded}(X)$  returns the solutions in  $X$  from amongst the most crowded regions

---

```

 $t \leftarrow 0$                                      /* Initialization */
 $M_t \leftarrow \emptyset$ 
 $\mathbf{x} \in X \leftarrow \text{Init}()$ 
 $t \leftarrow t + 1$ 
 $T \leftarrow T_{start}$ 
 $M_t \leftarrow M_{t-1} \cup \{\mathbf{x}\}$ 
while ( $\text{Terminate}(t, M_t) \neq \text{TRUE}$ )           /* Main Loop */
   $t \leftarrow t + 1$ 
   $N \leftarrow \emptyset$ 
  foreach ( $j \in 1.. \lambda$ )
     $\mathbf{x}^j \in X \leftarrow \text{Mutate}(\mathbf{x})$ 
     $N \leftarrow N \cup \{\mathbf{x}^j\}$ 
   $B \leftarrow \text{Tabu}(N, tabu\_l, attr\_l)$ 
   $B \leftarrow \text{Eq\_best}(B \cup M_{t-1}, B)$ 
   $B \leftarrow \text{Low\_pop}(B \cup M_{t-1}, B)$ 
   $\mathbf{x} \leftarrow \text{Rand}(B)$ 
   $Temp \leftarrow \text{ND}(B \cup M_{t-1})$ 
  while ( $|Temp| > arcsize$ )
     $Temp \leftarrow Temp \setminus \text{Most\_crowded}(Temp)$ 
  Update  $tabu\_l$ 
  Update  $attr\_l$ 
return  $M(t_{final})$                                      /* Termination */
  
```

---

Figure 4.27: Pseudocode for Tabu-PAES.

of solutions, and they provide adequate performance on many problems. They are also useful for ‘baselining’ the performance of more sophisticated approaches, such as evolutionary algorithms.

In this chapter, we developed a hillclimber for Pareto optimization, called (1+1)-PAES. This algorithm is unique in combining the use of a (1+1) selection strategy based on Pareto dominance, and an archive for storing nondominated vectors. Its acceptance function uses the archive as a comparison set to help estimate the relative quality of current and mutant solutions so that ‘incomparable’ solutions can be effectively evaluated.

PAES relies on a strategy for updating and maintaining a bounded archive of nondominated solutions. We proposed and evaluated several alternative archiving strategies and considered their convergence properties. In particular, a new crowding strategy, based on an adaptive grid in the objective space was shown to have several desirable attributes. The computational cost of an algorithm based on this adaptive grid archiving strategy was shown to be low, in terms of the number of solutions stored in the archive. The algorithm also requires no knowledge of the objective space for parameters to be selected (in contrast to phenotypic niching), and maintains those solutions that lead to a diverse and well-distributed set of points.

The generality of the methods used in the (1+1)-PAES algorithm facilitates the specification of many different variants of it. In order to demonstrate this we outlined the design of multi-start, simulated annealing, and tabu search variants of PAES.

In the next chapter, the efficacy and efficiency of the (1+1)-PAES algorithm is tested empirically.

# Chapter 5

## Empirical Testing of PAES

### 5.1 Overview

In this chapter we present two separate empirical studies comparing the performance of various MOEAs and our PAES algorithm. The first study, described in the next section, has been previously published in very similar form in [KC00b]. It is a comparison of the well known MOEAs, NPGA [HNG94] and NSGA [SD94], on a suite of six test functions. The second study, described in Section 5.3, has also been previously published in similar form, in [KC99a]. This study compares the performance of SPEA [ZT98a] with (1+1)-PAES on a different suite of test functions from [ZDT00].

### 5.2 Initial comparative assessment of PAES

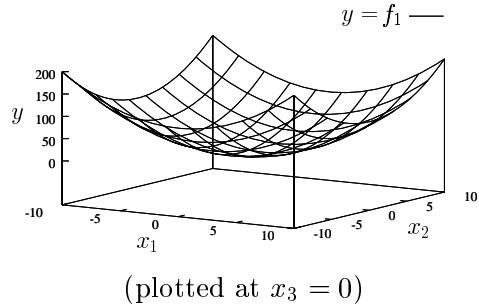
#### 5.2.1 Test functions $F_1-F_5$

In the suite of six test functions we use in this study, the first four are the same as used by Bentley and Wakefield [BW97]; i.e. Schaffer's functions  $F_1$ ,  $F_2$ , and  $F_3$  [Sch84], and Fonseca's  $f_1$  [FF95], the latter renamed here as  $F_4$ . These functions are now<sup>1</sup> commonly used by researchers to test multiobjective optimization algorithms. For reasons noted next we also designed a further test function which we call here  $F_5$ .

- $F_1$  is a single-objective minimization problem with one optimum:

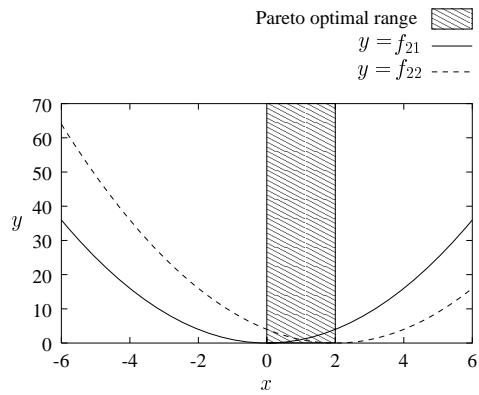
---

<sup>1</sup>At the time of performing these experiments (1998).



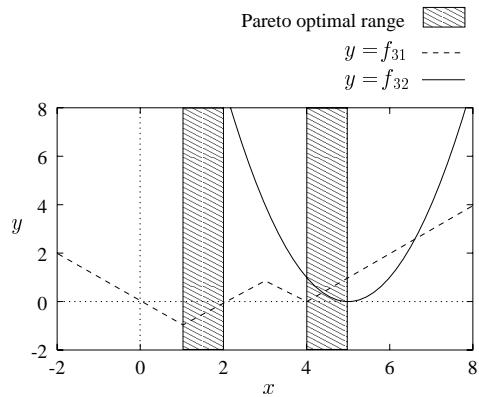
$$f_1 = x_1^2 + x_2^2 + x_3^2 \quad (5.1)$$

- $F_2$  is a two-objective minimization problem with a single range of Pareto optima which lie in  $0 \leq x \leq 2$ :



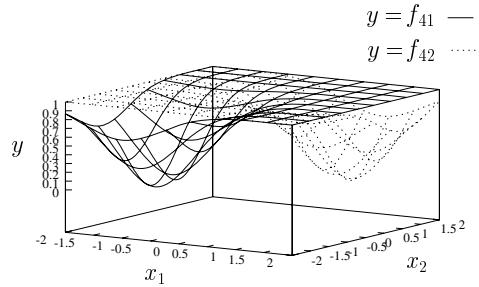
$$\begin{aligned} f_{21} &= x^2 \\ f_{22} &= (x - 2)^2 \end{aligned} \quad (5.2)$$

- $F_3$  is two-objective minimization problem with two separate ranges of Pareto optima which lie in  $1 \leq x \leq 2$  and  $4 \leq x \leq 5$ :



$$\begin{aligned}
 f_{31} &= -x & \text{where } x \leq 1 \\
 &= -2 + x & \text{where } 1 < x \leq 3 \\
 &= 4 - x & \text{where } 3 < x \leq 4 \\
 &= -4 + x & \text{where } 4 < x \\
 f_{32} &= (x - 5)^2
 \end{aligned} \tag{5.3}$$

- $F_4$  is a two-objective minimization problem on two variables with a single range of Pareto optima running diagonally from  $(-1, 1)$  to  $(1, -1)$ :



$$\begin{aligned}
 f_{41} &= 1 - e^{(-(x_1-1)^2 - (x_2+1)^2)} \\
 f_{42} &= 1 - e^{(-(x_1+1)^2 - (x_2-1)^2)}
 \end{aligned} \tag{5.4}$$

The above test functions are useful in testing multiobjective optimizers because they implicitly set two challenges: First, the set of nondominated solutions delivered by the optimizer should contain all of the function's Pareto optima; second, it is generally felt best if there is no strong bias favouring one Pareto optimum over others. In other words, in a MOGA, for example, the number of copies of each Pareto optimum in the final population should be similar. If not, this would seem to reveal a bias which may be undesirable in practical applications.

We designed  $F_5$  (described below) to provide stronger challenges in these respects; it is easily defined, but is a non-trivial problem. Each Pareto optimum is intrinsically difficult to find, and there are  $k$  distinct Pareto optima for chromosomes of length  $k$ , each of which has a different frequency i.e. some are far easier to find than others. This makes both challenges (as described above) stringent tests for any multiobjective optimizer.

The function  $F_5$  uses a  $k$ -ary chromosome of  $k$  genes. There are two objectives to be minimized, defined by the following two functions:

$$f_{51} = k - 1 - \sum_{i=0}^{k-2} \begin{cases} 1 & \text{if } G_{i+1} - G_i = 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

$$f_{52} = k - 1 - \sum_{i=0}^{k-2} \begin{cases} 1 & \text{if } G_i - G_{i+1} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.6)$$

where  $G_i$  is the allele value of the  $i$ th gene. For example a chromosome of length  $k = 5$  with allele values ‘1 2 3 2 2’ scores  $(5 - 1) - 2 = 2$  for the first objective (because there are two sites where, reading the chromosome from left to right, the allele value increases by exactly 1), and  $(5 - 1) - 1 = 3$  for the second objective, using similar reasoning. From this, we can see that the best score possible for either objective is 0 and the worst is  $k - 1$ , and the Pareto front is formed by solutions for which  $f_{51} + f_{52} = k - 1$ .

### 5.2.2 Test problem $F_6$

The Adaptive Distributed Database Management problem (ADDMP) is a problem faced by the providers of distributed database services, such as video-on-demand, scientific databases (such as genome databanks), and so forth. Oates and Corne (1998) gives a detailed description of the ADDMP, and C source code for the evaluation function can be found via the author’s website<sup>2</sup>. In this paper, we provide basic details of the ADDMP, aimed at conveying an understanding of its multiobjective nature, and an appreciation of the various forms which instances of the ADDMP can take.

The provider of a distributed database service needs to regularly ensure that database users (clients) are receiving an adequate quality of service (QoS). Indeed, clients’ subscription to the database may have come with guarantees from the service provider of distinct levels of QoS (perhaps different for different clients, according to subscription cost). The key factor (sometimes the only factor) in QoS is the response time experienced by a client for a typical database query.

If client  $c_i$  issues a request (database query)  $Q$  at time  $t_1$ , and the response arrives at time  $t_2$ , then the client has experienced a delay  $d_i = t_1 - t_2$ . In maximizing QoS, a service provider aims to minimize  $d_i$  for each client  $c_i$ . However, in a distributed database provision scenario, in which copies of all or part of the database are on several servers in places distributed throughout the globe, this minimization must occur in the context of load balancing. That is, we may be able to minimize the delays experienced by certain clients by routing their queries to the fastest server which contains the required data; however, the extra load on

<sup>2</sup><http://www.reading.ac.uk/~ssr97jdk>

this server will degrade the response times. So, the optimal solution will involve a careful balancing of clients across servers.

The ADDMP is hence the problem of finding the best client/server connection configuration, based on a static environment which details the underlying communications network, server speeds, and current access rates for each client. What counts as best depends on many things, but a single-objective QoS measure will typically involve a measure of the *worst* client delay (i.e.:  $\max_i d_i$ ) combined with a measure of the mean or median client delay. However, the single objective view is inadequate for many current versions of the ADDMP, and growing more so as distributed database service provision becomes more widespread and complex as regards the range of service guarantees on offer. For example, consider two potential solutions to a 5-client ADDMP in which the client delays (in milliseconds) are as in Table 5.1.

| Clients    | 1   | 2   | 3   | 4   | 5   |
|------------|-----|-----|-----|-----|-----|
| Solution 1 | 155 | 130 | 140 | 140 | 140 |
| Solution 2 | 350 | 80  | 90  | 90  | 90  |

Table 5.1: Two solution configurations for a simple 5-client ADDMP.

In solution 1, the worst delay is 155 ms, and the mean delay is 141 ms. In solution 2, the worst delay is far worse than in solution 1 at 350 ms, but the the mean delay is slightly better than solution 1 at 140 ms. Also, the median delay is considerably better in solution 2 than in solution 1. In a single-objective approach, which of solution 1 or 2 is preferred depends very much on the relative weightings given to the worst and mean (or median) components. It is therefore complex, and perhaps impossible, to arrive at a ‘correct’ relative weighting for these components, especially considering the widely different kinds of ADDMP scenarios (in terms of numbers of clients and patterns of database usage) which exist.

A multiobjective approach therefore seems more sensible and flexible. The growing complexity of QoS accounting and guarantees makes this even more necessary. Client 1, for example, may have paid for QoS guarantee which indicates that their delay will always be below 200 ms. Client 2, on the other hand, may have been given a guarantee that their delay would be always within 20median delay level at any snapshot in time. meanwhile, clients 3–5 may have taken a standard QoS guarantee specifying that their mean delays for a session would never exceed 150 ms. With a complex set of factors like this, the task of an optimizer addressing an ADDMP would be to quickly produce a good and diverse spread of solution configurations, leaving it to a later decision process to then choose from these on the basis of the various QoS guarantees in operation for the clients currently using the service. This makes the ADDMP a problem well-suited to Pareto optimization.

Test function  $F_6$  is an example instance of the ADDMP problem, involving ten nodes (each is both a client and a server), and in which quality of service is measured by two objectives, both of which must be minimized. The first objective is the worst response time (measured in milliseconds) seen by any client. This is clearly something which a database provider needs to minimize by reconfiguration. However, it is insufficient as a quality of service measure by itself. For example, if we have just three clients, then a situation in which the response times are respectively 750ms, 680ms, and 640ms will appear better, with this quality of service measure, than if the response times were 760ms, 410ms, 300ms. To achieve a more rounded consideration of quality of service, we therefore look at the tradeoff between this objective and another: the mean response time of the remaining (non-worst) clients. Hence, the two scenarios in this example would yield the following nondominated points: (750, 660), (760, 355).

### 5.2.3 Population-based variants of PAES

In this study we also investigate the performance of  $(1 + \lambda)$  and  $(\mu + \lambda)$  variants of the PAES algorithm.

The generation of  $\lambda$  mutants of the current solution increases the problem of deciding which one to accept as the next current solution(s). This is, in fact, carried out by assigning a fitness to each mutant based upon both a comparison with the archive and its grid location population.

Each of the  $\mu + \lambda$  population members is compared to the archive as it appeared after the last iteration and is assigned a *dominance score* as follows: Its score is initially zero and is set to 1 if it finds an archive member which it dominates. A score of 0 thus indicates it is nondominated by the archive. If it is dominated by any member of the archive its score is set to -1 and no more comparisons are necessary. All those mutants which could potentially join the archive are used to recalculate the ranges of the phenotype space. If this has changed by more than a threshold amount then the grid locations of the archive and potential archive members is recalculated. The archive is then updated. Finally, a fitness is assigned to each population member such that members with a higher dominance score are always given a higher fitness regardless of their grid location population. Points of the same dominance score have higher fitness the lower the population of their grid location.

Updating of the archive occurs as in  $(1 + 1)$ -PAES, ensuring that it contains only nondominated solutions and no copies. If it becomes full then solutions in sparse regions of the space will be favoured. This ensures that the comparison set covers a diverse range of individuals

so that the dominance score assigned to population members reflects their true quality.

In  $(\mu + \lambda)$ -PAES the  $\lambda$  mutants are generated by mutating one of the  $\mu$  current solutions, which is selected via binary tournament selection using the fitness values assigned in the previous iteration.

#### 5.2.4 Rival MOGAs tested

We compare the performance of PAES with that of two of the most well-known and respected MOEAs, the niched Pareto GA of Horn and Nafpliotis [HNG94] and a GA employing non-dominated sorting [SD94]. In order that we give each algorithm an equal opportunity of generating a good set of diverse solutions we add two extensions to the two genetic algorithms:

1. An archive of the nondominated solutions found is maintained (as in PAES) for presentation at the end of a run.
2. An elitist strategy.

The archive is not used to aid in selection, acceptance or any other part of the GA; it is merely there to give the GA the same opportunity as PAES has to present the best solutions it has found. Elitism is implemented as follows: In the case of the NSGA this is straightforward as fitness values are assigned and we can merely place into the new population the  $g$  fittest solutions, where  $g$  is the generation gap parameter. Thus, the NSGA has four different variants: the standard NSGA without elitism or archiving (NSGA), the NSGA with archiving (NSGA+ARC), the NSGA with elitism (NSGA+ELITE) and the NSGA with both elitism and archiving (NSGA+A+E). Elitism cannot be carried out easily in the niched Pareto GA, however, because explicit fitness values are never assigned. Thus, we have only three variants of the niched Pareto GA. These are the standard niched Pareto GA (NPGA), the NPGA with archiving (NPGA+ARC) and the NPGA with archiving and elitism (NPGA+A+E). The latter works by placing all individuals which were archived in the previous generation into the next generation.

Each of the algorithms require two or more parameter values to be set. A complete discussion regarding these choices is not included here. Instead, they are summarized in Table 5.2.

The NPGA uses the simple triangular sharing function  $Sh[d] = 1 - d/\sigma_{share}$  for  $d \leq \sigma_{share}$  and  $Sh[d] = 0$  for  $d > \sigma_{share}$ , where  $d$  is the phenotypic Euclidean distance between two solutions. We find that the NPGA requires a fairly large comparison set size in order for its

|                           | NPGA variants            | NSGA variants | PAES variants  |
|---------------------------|--------------------------|---------------|----------------|
| Population size $n$       | 100                      | 100           | 1 or $\lambda$ |
| Archive size $a$          | 100                      | 100           | 100            |
| Tournament size $t_{dom}$ | $4 \leq t_{dom} \leq 10$ | —             | 2              |
| Crossover $p_{cross}$     | 0.9                      | 0.9           | —              |
| Mutation $p_m$            | $1/k$                    | $1/k$         | $1/k$          |

Table 5.2: Summary of algorithm parameters.

estimate of the dominance ranking of individuals to remain fairly accurate. Similarly, the tournament size, cannot be set too low if accurate selection is to occur. Values of  $cs_{size} = 80$  and tournament size,  $10 \geq t_{dom} \geq 4$  are usually acceptable. The niche size parameter,  $\sigma_{share}$ , must also be set. Here, some experimentation is required as the niched Pareto GA can be quite sensitive to this parameter. So, for each of the problems attempted several test runs were undertaken to find reasonable values for the niche count parameter and the tournament size.

The nondominated sorting GA requires fewer parameters to be set. To set the niche size parameter several test runs were carried out to obtain reasonable performance. In our elitist variants of the NSGA we must also set the number of solutions,  $g$ , which are to be carried through to the next generation. In all experiments  $g = 5$  was used.

With the exception of test problem,  $F_5$ , uniform crossover was used in both of the above MOGAs. Single point crossover is more suited to finding solutions in  $F_5$  and this was used, again in both MOGAs. Values of crossover probability,  $p_{cross} = 0.9$  were used in both MOGAs and a bit mutation rate,  $p_m = 1/k$  for a chromosome of  $k$  genes, was used in all of the algorithms including PAES. In addition,  $(\mu + \lambda)$ -PAES requires a tournament size for selection. For this, a value of  $t_{dom} = 2$  was found to be acceptable on all problems.

### 5.2.5 Results and discussion

#### Experiments conducted

Each algorithm tested was allowed the same number of function evaluations,  $max\_evals$ , on each of the test problems. Following a number of trial runs to obtain good parameter settings, twenty uniquely seeded runs were carried out, and the resulting solution sets recorded. For each test function, the number of function evaluations allowed and the length of the chromosome,  $k$ , are given in Table 5.3.

|                  | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |
|------------------|-------|-------|-------|-------|-------|-------|
| <i>max_evals</i> | 1000  | 5000  | 5000  | 20000 | 20000 | 5000  |
| $k$              | 16    | 14    | 14    | 16    | 16    | 10    |

Table 5.3: The allowed number of function evaluations and chromosome length for each of the six test functions.

The single objective test problem,  $F_1$ , presents no difficulty to any of the optimizers considered in this section. The PAES algorithms all converge to the optimal solution and return, in their archive, the single nondominated solution only. The three GA versions which employ archiving exhibit the same behaviour, as expected. When no archive is used, the population of both the NPGA and the NSGA converge to this solution, subject to random mutations in the last generation. Because  $F_1$  presents no difficulty to any of the optimizers here, and is not itself a multiobjective problem, no further discussion or results relating to this problem are presented.

For each of the remaining five problems, tests were carried out in the following way: First, all of the NPGAs were compared, in pairs, one against the other (and also against (1 + 1)-PAES as a baseline), each time taking the combined space of the pair as the range over which to test, and using the statistical techniques outlined in Section 3.12. Next, in the same way, the NSGAs and the PAES algorithms were internally compared. From these three sets of internal tests, we chose the best NPGA, NSGA and PAES algorithm and compared these in the same fashion. Sometimes it was not clear from the original tests which algorithm in the initial groups should be carried forward to the ‘final’. Where this happens, further internal tests were performed and/or two inseparable algorithms were both carried forward for inclusion in the final. The set of best algorithms were also compared on their total combined space in terms of the percentages on which they were unbeaten, and beat all of the others. Finally, the combined space of *all* the algorithms was used and  $n(n - 1)$  comparisons were performed on the  $n = 13$  algorithms. Again, results were collected in terms of the percentages of the space on which each algorithm was unbeaten and beat all. All comparisons use a Mann-Whitney rank-sum test at the 95% confidence level.

For reasons of clarity we do not present the complete set of results described above. Nonetheless, only the tests carried out to decide on the best algorithm to carry forward to the ‘finals’ and the finals themselves are absent. All of the results for the internal tests for the niched Pareto GA are presented in Table 5.4. Similar sets of results for the NSGA and the PAES algorithms can be found in Tables 5.5 and 5.6, respectively. The results of testing all algorithms against each other on their combined phenotype space are given in Table 5.7.

## 5.2 Initial comparative assessment of PAES

|                                   | NPGA+arc  | NPGA+a+e    | (1 + 1)-PAES |
|-----------------------------------|-----------|-------------|--------------|
| $F_2$ - Schaffer's function $F_2$ |           |             |              |
| NPGA                              | [0, 97.0] | [0, 98.4]   | [0, 96.2]    |
| NPGA+arc                          | —         | [7.1, 9.6]  | [11.0, 7.6]  |
| NPGA+a+e                          | —         | —           | [10.9, 4.1]  |
| $F_3$ - Schaffer's function $F_2$ |           |             |              |
| NPGA                              | [0, 99.5] | [0, 99.4]   | [0, 99.0]    |
| NPGA+arc                          | —         | [4.9, 1.0]  | [0.9, 13.0]  |
| NPGA+a+e                          | —         | —           | [0.2, 19.3]  |
| $F_4$ - Fonseca's function $f_1$  |           |             |              |
| NPGA                              | [0, 100]  | [0, 100]    | [0, 100]     |
| NPGA+arc                          | —         | [12.8, 1.3] | [12.8, 1.6]  |
| NPGA+a+e                          | —         | —           | [2.9, 8.6]   |
| $F_5$ - $k$ -optima problem       |           |             |              |
| NPGA                              | [0, 100]  | [0, 100]    | [0, 100]     |
| NPGA+arc                          | —         | [93.6, 0]   | [34.7, 48.2] |
| NPGA+a+e                          | —         | —           | [0, 100]     |
| $F_6$ - the ADDMP                 |           |             |              |
| NPGA                              | [0, 99.8] | [0, 98.0]   | [0, 95.7]    |
| NPGA+arc                          | —         | [0.4, 0]    | [6.6, 90.0]  |
| NPGA+a+e                          | —         | —           | [2.2, 89.5]  |

Table 5.4: Comparison of three variants of the niched Pareto GA.

On  $F_5$ , the  $k$ -optima problem, the results presented warrant further analysis and discussion. To this end, plots of the best, worst and median distributions over the phenotype range are included. These plots help to clarify the statistical data and also illustrate different methods of visualizing the performance of multiobjective optimizers.

We find that the test, described above, in which all algorithms are tested against all others, in general, accurately reflects the results from the comparisons on pairs of algorithms on their own combined space. The percentage of the space on which an algorithm is unbeaten seems particularly reliable. For this reason, most of the following discussion is limited to the results presented Table 5.7 only. In addition, a summary of these results is provided in Table 5.8, at the end.

|                                   | NSGA+arc  | NSGA+elite   | NSGA+a+e   | (1 + 1)-PAES |
|-----------------------------------|-----------|--------------|------------|--------------|
| $F_2$ - Schaffer's function $F_2$ |           |              |            |              |
| NSGA                              | [0, 97.8] | [9.4, 6.0]   | [0, 99.9]  | [0, 97.6]    |
| NSGA+arc                          | —         | [100, 0]     | [7.6, 5.4] | [10.4, 2.7]  |
| NSGA+elite                        | —         | —            | [0, 98.2]  | [0, 98.2]    |
| NSGA+a+e                          | —         | —            | —          | [11.3, 2.3]  |
| $F_3$ - Schaffer's function $F_3$ |           |              |            |              |
| NSGA                              | [0, 99.7] | [41.6, 43.6] | [0, 99.0]  | [0, 98.7]    |
| NSGA+arc                          | —         | [100, 0]     | [3.8, 1.7] | [3.8, 2.8]   |
| NSGA+elite                        | —         | —            | [0, 100]   | [0, 100]     |
| NSGA+a+e                          | —         | —            | —          | [2.6, 3.3]   |
| $F_4$ - Fonseca's function $f_1$  |           |              |            |              |
| NSGA                              | [0, 97.7] | [1.8, 3.7]   | [0, 98.8]  | [0, 99.0]    |
| NSGA+arc                          | —         | [98.8, 0]    | [4.0, 3.2] | [9.7, .32]   |
| NSGA+elite                        | —         | —            | [0, 99.1]  | [0, 99.3]    |
| NSGA+a+e                          | —         | —            | —          | [8.5, 5.1]   |
| $F_5$ - $k$ -optima problem       |           |              |            |              |
| NSGA                              | [0, 38.1] | [6.1, 0]     | [0, 28.9]  | [0, 100]     |
| NSGA+arc                          | —         | [70.0, 0]    | [1.4, 0]   | [0, 77.4]    |
| NSGA+elite                        | —         | —            | [0, 70.7]  | [0, 100]     |
| NSGA+a+e                          | —         | —            | —          | [0, 79.0]    |
| $F_6$ - the ADDMP                 |           |              |            |              |
| NSGA                              | [0, 84.4] | [1.0, 19.3]  | [0, 92.9]  | [0, 84.8]    |
| NSGA+arc                          | —         | [53.9, 0]    | [0, 69.4]  | [1.7, 16.6]  |
| NSGA+elite                        | —         | —            | [0, 76.2]  | [0, 38.6]    |
| NSGA+a+e                          | —         | —            | —          | [4.0, 0]     |

Table 5.5: Comparison of four variants of the nondominated sorting GA.

|                                   | (1+10)-PAES | 1+50        | 10+1        | 10+10       | 10+50       |
|-----------------------------------|-------------|-------------|-------------|-------------|-------------|
| $F_2$ - Schaffer's function $F_2$ |             |             |             |             |             |
| (1 + 1)-PAES                      | [5.8, 2.1]  | [17.1, 1.4] | [12.7, 1.4] | [3.9, 5.4]  | [5.3, 3.7]  |
| 1+10                              | —           | [16.8, 1.8] | [9.3, 2.8]  | [1.1, 6.8]  | [4.2, 5.7]  |
| 1+50                              | —           | —           | [4.3, 10.4] | [0.8, 24.5] | [1.6, 19.9] |
| 10+1                              | —           | —           | —           | [1.1, 16.5] | [1.7, 10.9] |
| 10+10                             | —           | —           | —           | —           | [6.1, 4.7]  |
| $F_3$ - Schaffer's function $F_3$ |             |             |             |             |             |
| (1 + 1)-PAES                      | [9.5, 0.9]  | [10.0, 0]   | [11.6, 0.7] | [3.3, 1.4]  | [5.2, 0.8]  |
| 1+10                              | —           | [3.1, 1.8]  | [2.5, 1.2]  | [0.6, 6.5]  | [1.3, 55.3] |
| 1+50                              | —           | —           | [2.0, 3.3]  | [0.1, 8.2]  | [0.2, 45.3] |
| 10+1                              | —           | —           | —           | [0.7, 6.9]  | [0.9, 55.5] |
| 10+10                             | —           | —           | —           | —           | [2.5, 2.1]  |
| $F_4$ - Fonseca's function $f_1$  |             |             |             |             |             |
| (1 + 1)-PAES                      | [6.5, 5.2]  | [4.4, 3.9]  | [19.0, 1.6] | [8.1, 2.9]  | [12.4, 1.8] |
| 1+10                              | —           | [3.1, 7.6]  | [18.0, 1.5] | [6.1, 2.0]  | [9.5, 1.3]  |
| 1+50                              | —           | —           | [19.6, 0.9] | [6.8, 2.2]  | [7.1, 0.7]  |
| 10+1                              | —           | —           | —           | [2.4, 15.3] | [4.3, 8.4]  |
| 10+10                             | —           | —           | —           | —           | [6.9, 1.8]  |
| $F_5$ - $k$ -optima problem       |             |             |             |             |             |
| (1 + 1)-PAES                      | [74.7, 0]   | [100, 0]    | [100, 0]    | [89.3, 0]   | [92.3, 0]   |
| 1+10                              | —           | [38.6, 0]   | [100, 0]    | [53.5, 0]   | [70.0, 0]   |
| 1+50                              | —           | —           | [100, 0]    | [19.6, 0]   | [2.2, 0]    |
| 10+1                              | —           | —           | —           | [0, 82.1]   | [0, 100]    |
| 10+10                             | —           | —           | —           | —           | [0, 1.9]    |
| $F_6$ - the ADDMP                 |             |             |             |             |             |
| (1 + 1)-PAES                      | [79.0, 0]   | [22.0, 0]   | [48.7, 0]   | [15.4, 0]   | [15.4, 0.5] |
| 1+10                              | —           | [0, 0.2]    | [0, 0]      | [0, 0]      | [8.0, 75.3] |
| 1+50                              | —           | —           | [4.3, 0]    | [0, 0]      | [0, 37.6]   |
| 10+1                              | —           | —           | —           | [0, 0]      | [0, 12.3]   |
| 10+10                             | —           | —           | —           | —           | [0, 3.0]    |

Table 5.6: Comparison of six variants of the Pareto archived evolution strategy.

### (1 + 1)-PAES results

The performance of (1 + 1)-PAES on the test functions used here provides considerable evidence that it is a capable multiobjective optimizer on a range of problem types. In fact, amongst the thirteen algorithms tested here, it is the most consistent performer. Table 5.7 shows that, where all algorithms are pair-wise compared against the combined nondominated front, (1 + 1)-PAES is unbeaten on, in the worst case, 68% of the front (problem  $F_2$ ). On problem  $F_5$ , (1 + 1)-PAES covers the largest part of the Pareto front and manages to find the most demanding solutions, not generated by any of the other algorithms tested. (Problem  $F_5$  is discussed further towards the end of the results section.) It seems that (1 + 1)-PAES works well for the same reasons that it is computationally simple: it is an aggressive algorithm, testing each solution generated in a stringent manner, and investing few resources in solutions which do not pass the test. In this sense, it is the analogue of a single-objective hillclimber. This has drawbacks too. (1 + 1)-PAES (or  $1+\lambda$ ) would be stumped by any search space which contained local optima which could not be traversed by its small change (mutation) operator, as it has no facility for moving from the current solution to an inferior one (in the Pareto sense). This is possibly less of a flaw in multiobjective spaces than in single objective ones because with more objectives the occurrence of functions with true local optima may be reduced. However, test function  $F_3$  is an example of a function where a hillclimbing approach could get stuck. If an optimizer were to start in the right hand range of optima i.e. with  $5 \geq x > 4$  it would not be able to move to the left optima by small changes to the variable  $x$ . PAES does not suffer from this problem because  $x$  is encoded as an  $n$ -bit binary string and PAES is allowed to move by changing one or more of the  $n$  bits. Therefore, it is able to jump across the divide.

Timings for six of the algorithms presented in this section are also included in Table 5.9. In this case, the test function ( $F_5$ ) takes only a small proportion of the total computation time, so the differing computation times of each algorithm are clear. (1 + 1)-PAES is 37% faster than its nearest rival, the NPGA, on this test problem.

### (1 + $\lambda$ )-PAES results

The (1+10) and (1+50) variants of PAES do not do nearly as well as the baseline (1 + 1) approach. Only on one problem,  $F_4$ , does (1+50)-PAES generate better distributions over the 20 runs than (1 + 1)-PAES, and (1+10)-PAES never does. The lack of competitiveness of (1+ $\lambda$ )-PAES might be explained with relation to its strategy for replacing the current solution. As in (1 + 1)-PAES, the current solution is first compared with each mutant. In the

| Algorithm      | Distribution | Test Problem |       |       |       |       |
|----------------|--------------|--------------|-------|-------|-------|-------|
|                |              | $F_2$        | $F_3$ | $F_4$ | $F_5$ | $F_6$ |
| NPGA           | unbeaten     | 0.2          | 0     | 0     | 0     | 31.5  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |
| NPGA+arc       | unbeaten     | 75.1         | 66.6  | 88.4  | 51.9  | 37.7  |
|                | beats all    | 1.1          | 0.1   | 0.2   | 31.1  | 0     |
| NPGA+a+e       | unbeaten     | 77.8         | 17.7  | 67.3  | 0     | 37.4  |
|                | beats all    | 0.1          | 0     | 0     | 0     | 0     |
| NSGA           | unbeaten     | 0            | 0     | 0     | 0     | 32.6  |
|                | beats all    | 0            | 0     | 0.1   | 0     | 0     |
| NSGA+arc       | unbeaten     | 80.9         | 51.9  | 87.0  | 27.9  | 42.7  |
|                | beats all    | 0.1          | 0.3   | 0.2   | 0     | 0     |
| NSGA+elite     | unbeaten     | 0            | 0     | 0.3   | 0     | 82.1  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |
| NSGA+a+e       | unbeaten     | 78.8         | 90.4  | 83.6  | 26.9  | 99.5  |
|                | beats all    | 0            | 1.0   | 0     | 0     | 0     |
| (1 + 1)-PAES   | unbeaten     | 68.0         | 89.8  | 71.7  | 68.9  | 94.9  |
|                | beats all    | 0            | 0     | 0     | 16.1  | 0     |
| (1 + 10)-PAES  | unbeaten     | 65.7         | 35.0  | 65.6  | 31.0  | 32.4  |
|                | beats all    | 0            | 1.0   | 0     | 0     | 0     |
| (1 + 50)-PAES  | unbeaten     | 45.1         | 30.4  | 72.4  | 0     | 32.5  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |
| (10 + 1)-PAES  | unbeaten     | 51.8         | 30.8  | 47.1  | 0     | 32.3  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |
| (10 + 10)-PAES | unbeaten     | 74.8         | 87.7  | 67.8  | 13.3  | 37.8  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |
| (10 + 50)-PAES | unbeaten     | 69.0         | 82.5  | 55.0  | 10.7  | 53.7  |
|                | beats all    | 0            | 0     | 0     | 0     | 0     |

Table 5.7: Pair-wise comparisons of all algorithms on the combined phenotype space for all problems.

case where exactly one member dominates the current solution, this will be accepted as the current solution of the next iteration. However, in all other cases, the acceptance is based upon the result of comparing each mutant with the archive of the previous iteration. Mutants are not compared one against the other. Any ties which occur are broken first with reference to the population in the mutants' grid locations and if this is inconclusive, randomly. This approach can lead to acceptance of a mutant which is dominated by one of the other mutants of its generation. In this case, some of the characteristic aggressiveness of  $(1+1)$ -PAES may be lost. The archive of the previous generation was used to balance the need for a static test of the current generation with computational parsimony. Comparing mutants against a constantly updated archive results either in an inaccurate assessment of their dominance rank, or requires tagging of those mutants which dominated the archive, scoring one, but were later dominated by another mutant, in order to correct their scores. Rather than add extra complexity, the option of using the archive of the previous generation was taken. It is unclear at the time of writing if this issue is, in fact the only factor (or most important factor) which affects the performance of  $(1+\lambda)$ -PAES but this is under investigation.

### $(\mu + \lambda)$ -PAES results

The population based variants,  $(10+1)$ ,  $(10+10)$  and  $(10+50)$  perform comparably with  $(1+1)$ -PAES on problems  $F_2$ ,  $F_3$  and  $F_4$ . On  $F_2$ ,  $(10+10)$ -PAES is superior to  $(1+1)$ . However, the population-based methods do not fare well on  $F_5$  or  $F_6$  and lack the consistently high performance of  $(1+1)$ -PAES. The use of a population does not, in general, improve the performance of the basic PAES algorithm, and adds considerable computational overhead (see Table 5.9). However, similar comments as those regarding the acceptance strategy used in  $(1+\lambda)$ -PAES apply equally here to  $(\mu + \lambda)$ -PAES.

On problems  $F_5$  and  $F_6$ , which are both multimodal, we might have expected a population to be beneficial to PAES. Indeed, to explain the poor performance, one may be tempted to blame the lack of a recombination operator. However, the reader should notice that on these two problems  $(1+1)$ -PAES (with no recombination) performs at a high level compared to both NPGA and NSGA, which both use a population and recombination.

### NPGA results

Turning now to the evolutionary algorithms, the first thing we notice is that without exception (not surprisingly), the archived versions consistently outperform the non-archived ones. Also, elitism is generally beneficial. The elitist technique employed in the NPGA is not so

successful, however, only enhancing the results in one of the test problems and degrading them considerably in the others.

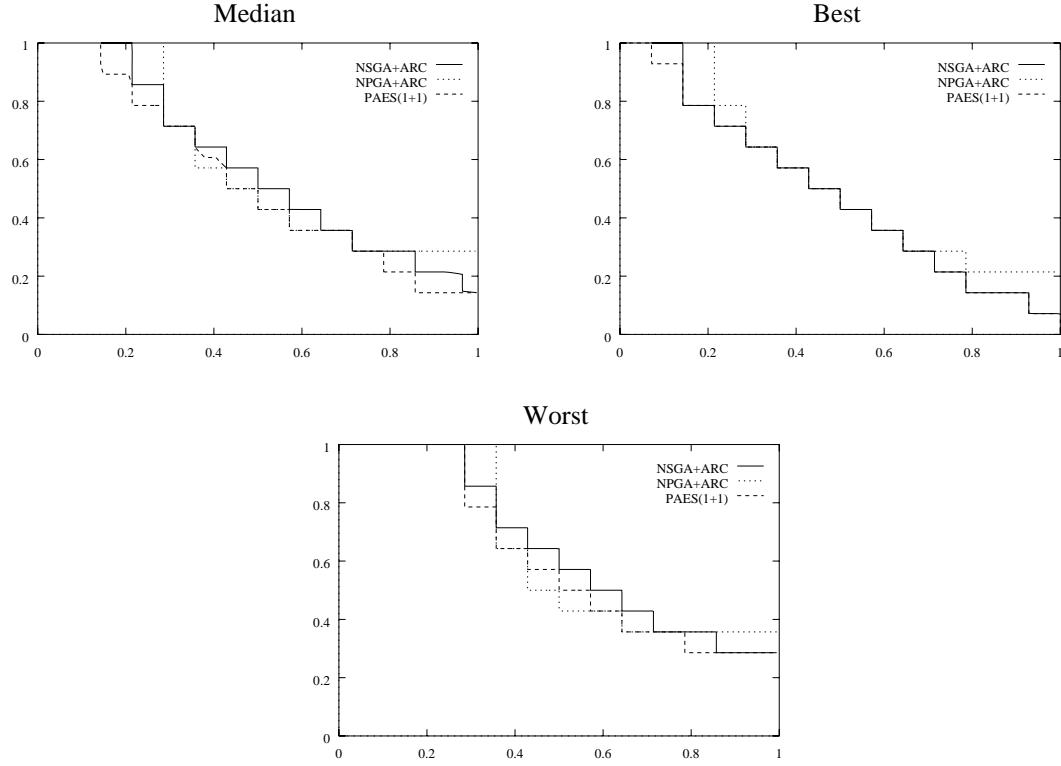
Overall, the NPGA with archiving does quite well in comparison to both (1+1)-PAES and the NSGA. It is superior to both of them on problem  $F_4$ . On  $F_6$ , the ADDMP, its performance is weak, however, and it does not perform as consistently well as either the NSGA with archiving and elitism or our baseline approach (1 + 1)-PAES. It is also the most difficult of the algorithms to use, requiring more parameters to be set, some of which can severely degrade performance if set incorrectly. Its computational complexity is low compared to either the population based PAES algorithms or the NSGAs because it does not have to explicitly assign fitness values to the population. However, (1 + 1)-PAES seems both a more consistent performer (see Table 5.8) and a faster algorithm on the results presented here.

### NSGA results

The NSGAs recursively sort the current population into two sets, the nondominated and the dominated. This approach gives a fairly accurate estimate of the dominance rank of each individual, encouraging selection to focus on the best members of the population. This is perhaps why the NSGAs, when coupled with the archiving of all nondominated solutions and elitism performs slightly better than the NPAGAs. It also employs a more accurate form of niching than the NPGA which approximates the niching process using equivalence class sharing.

The NSGA with archiving and elitism is ranked first on three of the five multiobjective test problems, when all algorithms are compared pair-wise on the overall combined space. Its lowest rank is on problem  $F_5$ , where its performance is quite poor in comparison to both the NPGA with archiving alone and those of some of the PAES algorithms. In fact it is nondominated on only 26.9% of the combined space. These results are summarized in Table 5.8.

The NSGA is computationally expensive compared with either the NPGA or the local search versions of PAES. Its average time complexity is greater than either, because it uses many comparisons to rank the current population and to calculate the niche count so that fitness values can be assigned. When the NSGA was timed on test problem  $F_5$  it was found to be the slowest algorithm here (see Table 5.9). Nonetheless, this overhead is unimportant in many applications where the evaluation of solutions is by far the most time-consuming process in the search for solutions, and reducing the number of evaluations is more important.


 Figure 5.1: Best, median and worst attainment surfaces found on  $F_5$ .

### Test function $F_5$ results

From Table 5.7 it appears that the algorithm which is unbeaten on the largest percentage of the space does not always also beat all with the highest percentage. On  $F_5$ , the  $k$ -optima problem, for instance,  $(1 + 1)$ -PAES is unbeaten on 68.9% of the combined phenotype space but only beats all on 16.1%. The NPGA with archiving, on the other hand, is unbeaten on less of the space but beats all on 31.1%. It would be interesting to see how these figures vary with the use of different confidence levels. In the case of problem  $F_5$ ,  $(1 + 1)$ -PAES beats all on 16% of the space because it has generated solutions at the edges of the range of optima, where other algorithms have failed to do so. The NPGA, by contrast, has a better distribution in the centre of the Pareto front.

For the time being, we indicate the difference in the distributions of points generated by the NPGA with archiving, and  $(1 + 1)$ -PAES by plotting graphs of the best, worst and median attainment surfaces of these algorithms on problem  $F_5$ . The best of the NSGAs is also included in the graphs which are shown in Figure 5.1<sup>3</sup>. The NSGA is also interesting because

<sup>3</sup>Note that all surfaces are orthogonal, although in the case of the median surfaces this is only apparent at

|               |                           | NPGA+a | NSGA+a+e | (1 + 1)-PAES |
|---------------|---------------------------|--------|----------|--------------|
| $F_2$         | rank                      | 4      | 2        | 7            |
|               | unbeaten                  | 75.1   | 78.8     | 68.0         |
| $F_3$         | rank                      | 5      | 1        | 2            |
|               | unbeaten                  | 66.6   | 90.4     | 89.8         |
| $F_4$         | rank                      | 1      | 3        | 4            |
|               | unbeaten                  | 88.4   | 83.6     | 71.7         |
| $F_5$         | rank                      | 2      | 5        | 1            |
|               | unbeaten                  | 51.9   | 26.9     | 68.9         |
| $F_6$         | rank                      | 7      | 1        | 2            |
|               | unbeaten                  | 37.7   | 99.5     | 94.9         |
| overall stats | worst rank                | 7      | 5        | 7            |
|               | sum of ranks              | 19     | 12       | 16           |
|               | worst coverage (unbeaten) | 37.7   | 26.9     | 68.0         |

Table 5.8: Summary statistics for best 3 optimizers.

| Algorithm    | Run times on SPARC Ultra 10 300MHz |              |
|--------------|------------------------------------|--------------|
|              | Mean (seconds)                     | SD (seconds) |
| (1 + 1)-PAES | 1.85                               | 0.0446       |
| (10+50)-PAES | 4.48                               | 0.0283       |
| NSGA         | 8.16                               | 0.0988       |
| NSGA+a+e     | 8.45                               | 0.0127       |
| NPGA         | 2.96                               | 0.0853       |
| NPGA+a+e     | 3.03                               | 0.0729       |

Table 5.9: Algorithm run times on test problem  $F_5$ .

although it appears to do relatively poorly from the statistical results, its best distribution is rather better than that of the NPGA. The best attainment surfaces show that (1 + 1)-PAES finds optima which extend the furthest towards the ends of the Pareto front. The NSGA is nearly as good, and the NPGA is least impressive on this measure. This is why it is beaten on approximately 49% of the space. The median, similarly, is not favourable for the NPGA in the most part, although it beats the other two algorithms in a small portion of the space near the centre. Finally, the plots of the worst attainment surface reveal why the NPGA beats all the other algorithms on such a large percentage of the space. Its worst attainment surface, again in the centre of the space, is significantly better than the other two algorithms. Returning to the comparison of pairs of algorithms on problem  $F_5$ , it can be seen that (1 + 1)-PAES had a better distribution than the NPGA with archiving on 48.2% of the space compared with 34.7% vice versa. This result seems to be borne out by the plots in Figure 5.1, and gives in this case a truer picture of the algorithm with the best coverage of the space than the 'beats all' statistic discussed above.

### 5.3 Assessment of PAES using Deb's test functions

In this section, the raw data sets from the comparative study [ZDT00] by Zitzler *et al.*<sup>4</sup> are taken and compared with the performance of (1+1)-PAES, on the same set of test functions. As far as possible, attempts have been made to recreate the same operating modes (i.e. number of evaluations, parameter settings etc.) for (1+1)-PAES as was set for each of the multiobjective EAs in the Zitzler *et al.* study, and where this was not possible or demonstrable, the parameters of PAES were varied to establish if any differences in performance could be explained by the different conditions. The aim of these experiments is to establish whether a local search optimizer can cope with the diverse set of test functions which are used in [ZDT00]. Furthermore, we wish to find out if, like the other MOEAs in that study, PAES can be placed at some position in a hierarchy of performance, independently of the test function, or whether it has any different areas of strength or weakness. Finally, a number of further runs are also described, in which some tuning of the local optimizer is carried out. A number of different mutation rates are tried, and the coding of the test functions is changed from binary to Gray. These experiments are carried out to explore the best parameter settings for PAES and to establish how the coding of problems may interact with the search algorithm.

In the next section the test functions used in [ZDT00] are reviewed. Section 5.3.2 provides high resolution.

---

<sup>4</sup>available from <http://www.tik.ee.ethz.ch/~zitzler/testdata.html>

details of the experiments that were carried out in [ZDT00], explains how these experiments have been replicated for testing PAES, giving the parameter settings used and noting where there are differences in experimental set up. The results of these experiments are tabulated, graphical results are presented, and some discussion is provided. A summary of our experiments and findings is then given. Finally, a concluding section presents a conjecture relating to our findings and indicates lines of possible future research.

### 5.3.1 Deb's test functions

In [Deb98], a procedure for creating tunable, multiobjective test functions with specific problem features was presented. In the paper, two tasks that any effective MOEA should perform are identified. These are:

1. Guide the search towards the global Pareto-optimal region, and
2. Maintain population diversity in the current nondominated front.

A number of problem features, such as multimodality, deception, convexity, which can cause difficulty in performing these two tasks, are considered. A general scheme for generating test functions is then proposed. The scheme enables an algorithm's response to each of the identified problem features to be isolated and investigated.

In this scheme, each test function consists itself of three functions  $f_1, g, h$  :

$$\begin{aligned}
 \text{Minimize} \quad \mathcal{T}(\mathbf{x}) &= (f_1(x_1), f_2(\mathbf{x})) \\
 \text{subject to} \quad f_2(\mathbf{x}) &= g(x_2, \dots, x_m)h(f_1(x_1), g(x_2, \dots, x_m)) \\
 \text{where} \quad \mathbf{x} &= (x_1, \dots, x_m)
 \end{aligned} \tag{5.7}$$

The function  $f_1$  is a function of  $x_1$  only,  $g$  is a function of the remaining  $m - 1$  variables, and  $h$  is dependent on  $f_1$  and  $g$ . Several example test functions are then given to illustrate how problems incorporating some of these features can be constructed.

Six such test functions, constructed by Deb, were later used in a study by Zitzler *et al.* [ZDT00], that compares the performance of eight different MOEAs. The six test functions (defined below) comprise  $\mathcal{T}_1$  with a convex Pareto front,  $\mathcal{T}_2$  with a non-convex Pareto front,  $\mathcal{T}_3$  having a discontinuous Pareto front,  $\mathcal{T}_4$  a multimodal problem with  $21^9$  local Pareto fronts,  $\mathcal{T}_5$  a deceptive problem, and  $\mathcal{T}_6$  having a non-uniformly distributed search space with solutions non-uniformly distributed along the Pareto front. Each of the functions is a two-objective minimization problem on  $m$  parameters. In five of the problems the parameters  $x_i$  are coded

as a binary string, and decoded such that  $x_i \in [0, 1]$ . The remaining function ( $\mathcal{T}_5$ ) also employs a binary chromosome but this time unitation is used to evaluate each of the parameters. The experiments presented in our study here, employ identical functions to those presented in [ZDT00] and are coded onto chromosomes using identical numbers of bits to represent each parameter.

The test function  $\mathcal{T}_1$  has a convex Pareto-optimal front and the solutions are distributed with a symmetric density about a maximum which is far above the Pareto-optimal front:

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \sum_{i=2}^m x_i \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \quad (5.8)$$

where  $m = 30$  and  $x_i \in [0, 1]$ . The Pareto-optimal front is formed with  $g(\mathbf{x}) = 1$ .

The test function  $\mathcal{T}_2$  is the non-convex counterpart to  $\mathcal{T}_1$ . Again, the solutions are distributed with a symmetric density about a maximum which is far above the Pareto-optimal front:

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \sum_{i=2}^m x_i \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \quad (5.9)$$

where  $m = 30$  and  $x_i \in [0, 1]$ . The Pareto-optimal front is formed with  $g(\mathbf{x}) = 1$ .

The test function  $\mathcal{T}_3$  represents the discreteness features; its Pareto-optimal front consists of several non-contiguous convex parts, and the search space is unbiased:

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \sum_{i=2}^m x_i \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g) \sin(10\pi f_1) \end{aligned} \quad (5.10)$$

where  $m = 30$  and  $x_i \in [0, 1]$ . The Pareto-optimal front is formed with  $g(\mathbf{x}) = 1$ . The introduction of the sine function in  $h$  causes discontinuity in the Pareto-optimal front. However, there is no discontinuity in the search space.

The test function  $\mathcal{T}_4$  contains  $21^9$  local Pareto-optimal fronts and therefore tests for the optimizer's ability to deal with multimodality:

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 10(m-1) + \sum_{i=2}^m (x_i^2 - 10 \cos(4\pi x_i)) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \quad (5.11)$$

where  $m = 10$ ,  $x_1 \in [0, 1]$ , and  $x_2, \dots, x_m \in [-5, 5]$ . The global Pareto-optimal front is formed with  $g(\mathbf{x}) = 1$ , the best local Pareto-optimal front with  $g(\mathbf{x}) = 1.25$ . Note that not all local Pareto-optimal fronts are distinguishable in the objective space.

The test function  $\mathcal{T}_5$  describes a deceptive problem and distinguishes itself from the other test problems in that  $x_i$  represents a binary string:

$$\begin{aligned} f_1(x_1) &= 1 + u(x_1) \\ g(x_2, \dots, x_m) &= \sum_{i=2}^m v(u(x_i)) \\ h(f_1, g) &= 1/f_1 \end{aligned} \tag{5.12}$$

where  $u(x_i)$  gives the number of ones in the bit vector  $x_i$  (unitation),

$$v(u(x_i)) = \begin{cases} 2 + u(x_i) & \text{if } u(x_i) < 5 \\ 1 & \text{if } u(x_i) = 5 \end{cases}$$

and  $m = 11$ ,  $x_1 \in \{0, 1\}^{30}$ , and  $x_2, \dots, x_m \in \{0, 1\}^5$ . The true Pareto-optimal front is formed with  $g(\mathbf{x}) = 10$ , while the best deceptive Pareto-optimal front is formed with  $g(\mathbf{x}) = 11$ . The global Pareto-optimal front as well as each of the local ones is convex.

The test function  $\mathcal{T}_6$  includes two difficulties caused by the non-uniformity of the search space: firstly, the Pareto-optimal solutions are non-uniformly distributed along the global Pareto front (the front is biased for solutions for which  $f_1(x)$  is near one); secondly, the density of the solutions is least near the Pareto-optimal front and highest away from the front:

$$\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_1) \sin^6(6\pi x_1) \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot (\sum_{i=2}^m x_i)^{0.25} \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{5.13}$$

where  $m = 10$  and  $x_i \in [0, 1]$ . The Pareto-optimal front is formed with  $g(\mathbf{x}) = 1$  and is convex.

### 5.3.2 Experimental results

#### Experimental method

In [ZDT00] eight algorithms are compared on the six Deb test functions. Each algorithm was executed 30 times on each test problem, and off-line a record of all nondominated solutions encountered was kept, and returned as the outcome of each run. For each pair of algorithms

$A, B$  and for each of thirty runs, a metric  $\mathcal{C}$  was used to compute the proportion of nondominated points in  $A$  covered by  $B$  and vice versa. The median and interquartile ranges of the resulting  $\mathcal{C}$  values were then plotted using box plots.

The results presented in the paper indicate that on four of the test functions,  $\mathcal{T}_1 - \mathcal{T}_3$  and  $\mathcal{T}_6$ , the Strength Pareto Evolutionary Algorithm (SPEA) [ZT98a], generates solution sets which consistently dominate *all* of the other algorithms tested. On test function  $\mathcal{T}_4$  SPEA is clearly superior to all other algorithms in terms of the median of the  $\mathcal{C}$  value although it no longer consistently beats two of the algorithms, namely NSGA [SD94], and SOEA (a single objective EA run 100 times with a different randomly chosen linear combination of the objectives). However, although SPEA is not consistently better than these two algorithms on this function, it does produce a better overall distribution of  $\mathcal{C}$  values. On test function  $\mathcal{T}_5$ , very similar performance can be observed between NSGA and SPEA, while SOEA actually does slightly better in terms of the median and the whole distribution of  $\mathcal{C}$  values achieved. To summarize these results, one can say that SPEA is the best algorithm overall and is only beaten on one test function,  $\mathcal{T}_5$ , by one algorithm, SOEA, which was given far more evaluations than any other algorithm (in fact 100 times as many).

Due to the dominance of SPEA alluded to above, PAES is initially tested only against the raw results achieved by SPEA, on each of the test functions. Where PAES does not exhibit clear superiority over SPEA on a particular test function, then further tests against the other algorithms in [ZDT00] are performed, again using the raw data available from the web-site given earlier. This should give an indication of the approximate rank of PAES compared with the other algorithms, without the need for explicitly testing against each of them.

The algorithms in [ZDT00] each employed a population of 100 solutions and were run for 250 generations on each of the test functions. The only exceptions to this were SOEA (using the results from 100 runs) and SPEA which has two populations, an internal population and an external population. Zitzler *et al.* chose to give SPEA an internal population of 80 and external population of 20 out of fairness to the other algorithms. This immediately gives us a problem when we wish to replicate the conditions given to the other algorithms for PAES. Since we would like to be sure of our results we try all of the alternative options and observe the effect of these different conditions. PAES has an internal population of just two since it uses a (1+1) or hillclimbing strategy. However, like SPEA it does have an external population, though it is used only as a comparison set for the purpose of ranking the current and candidate solutions, and is never used as a pool from which selection can occur. All of the algorithms in [ZDT00] have a total population of 100. Therefore, PAES could be given an external population limit of 98 so that it too has a total population of 100. This seems to be

### 5.3 Assessment of PAES using Deb's test functions

| Problem         | Algorithm version |              |              |              |
|-----------------|-------------------|--------------|--------------|--------------|
|                 | PAES_on-line      | PAES[98]     | PAES[20]     | PAES[98]Gray |
| $\mathcal{T}_1$ | [47.6, 7.5]       | [88.5, 3.6]  | [24.9, 3.6]  | [0, 97.1]    |
| $\mathcal{T}_2$ | [93.5, 0.8]       | [86.9, 3.6]  | [90.1, 1.1]  | [2.3, 48.5]  |
| $\mathcal{T}_3$ | [97.6, 1.0]       | [95.9, 2.1]  | [95.1, 2.6]  | [0.3, 77.5]  |
| $\mathcal{T}_4$ | [0.0, 100.0]      | [0.0, 100.0] | [0.0, 100.0] | [97.7, 0.1]  |
| $\mathcal{T}_5$ | [0.0, 100.0]      | [0.0, 100.0] | [0.0, 100.0] | —            |
| $\mathcal{T}_6$ | [100.0, 0.0]      | [100.0, 0.0] | [99.5, 0.0]  | [98.1, 0.0]  |

Table 5.10: PAES vs SPEA results from the AS2 metric on the six test functions. Four different operating modes of PAES were tested. The result top left means that on 47.6% of the sample lines the distribution of PAES\_on-line's attainment surfaces were significantly better than those of SPEA, and on 7.5% SPEA's distribution was significantly better. (On the remaining 54.9% the two distributions were not significantly different).

the fairest method of comparing these algorithms. However, SPEA was only given an external population of 20. Potentially, this could have hindered its performance (although this set up was chosen presumably to be the best compromise of internal and external population size given a maximum total population of 100) so we also compare PAES , using an external population of 20, with SPEA.

The number of evaluations carried out in [ZDT00] by each algorithm is 25000 ( $250 \times 100$ ). The total number of evaluations for all versions of PAES tested is also 25000. All of the algorithms in [ZDT00] were tested with respect to their off-line performance. Zitzler *et al.* note that this changes the relative performance of some algorithms compared to when on-line performance is considered. The performance of VEGA, for example, is improved greatly by considering its off-line performance. Thus, to demonstrate the on-line performance of PAES we include comparison of PAES with SPEA where PAES is only allowed to return the solutions stored in its external archive population at the end of each run, a maximum of 100, in addition to our off-line tests.

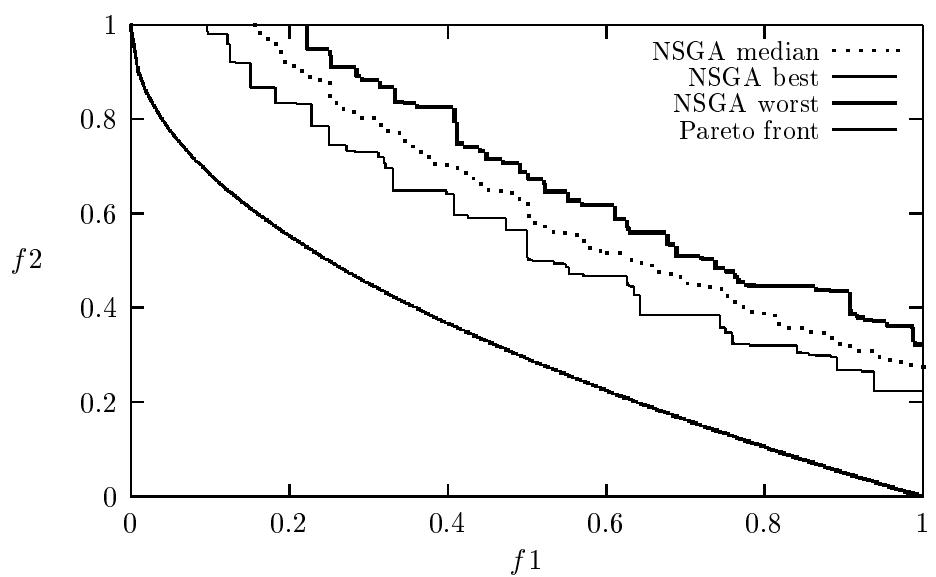


Figure 5.2: T1 NSGA.

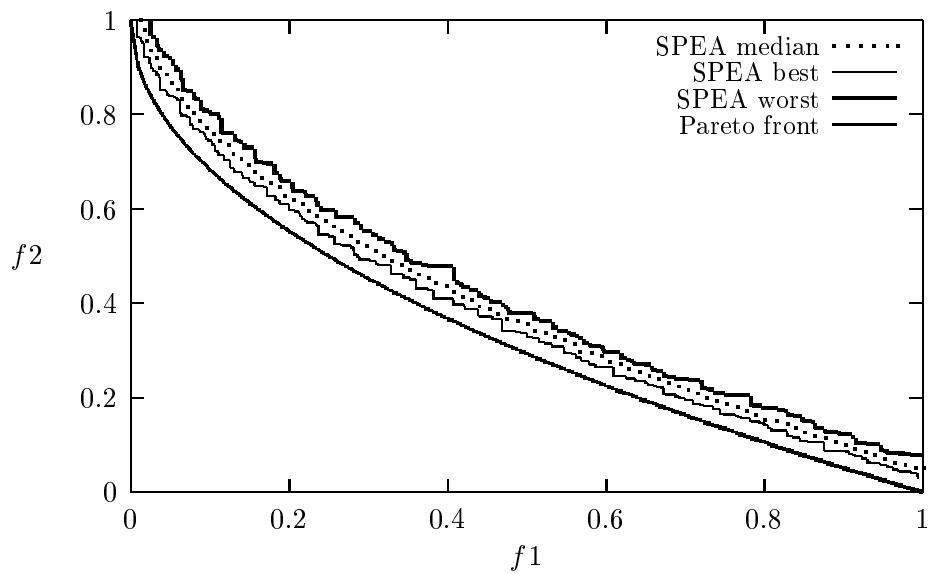


Figure 5.3: T1 SPEA.

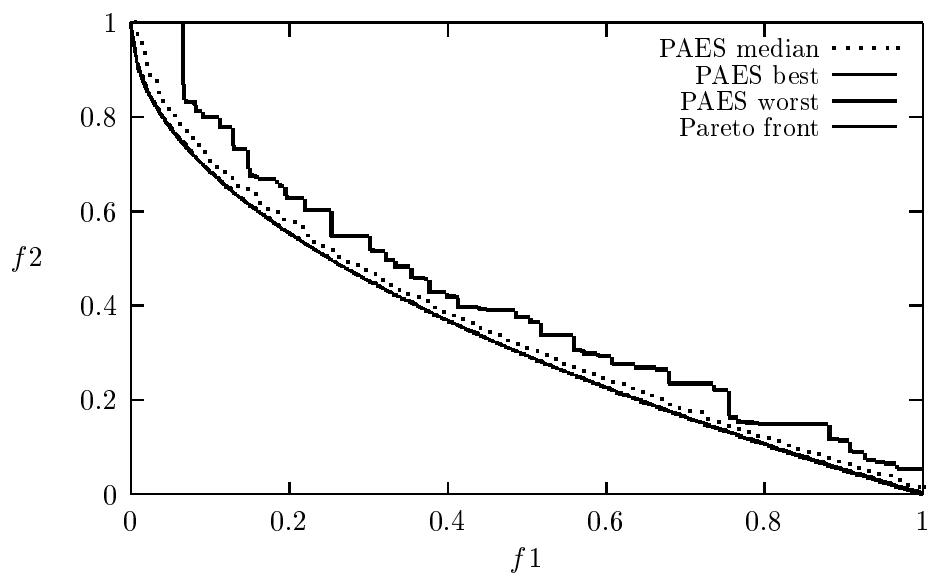


Figure 5.4: T1 PAES.

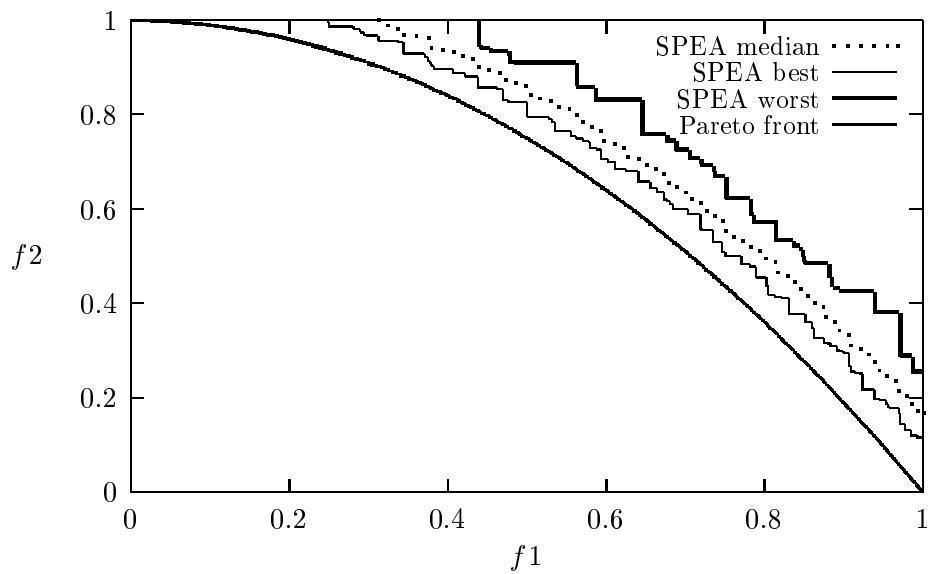


Figure 5.5: T2 SPEA.

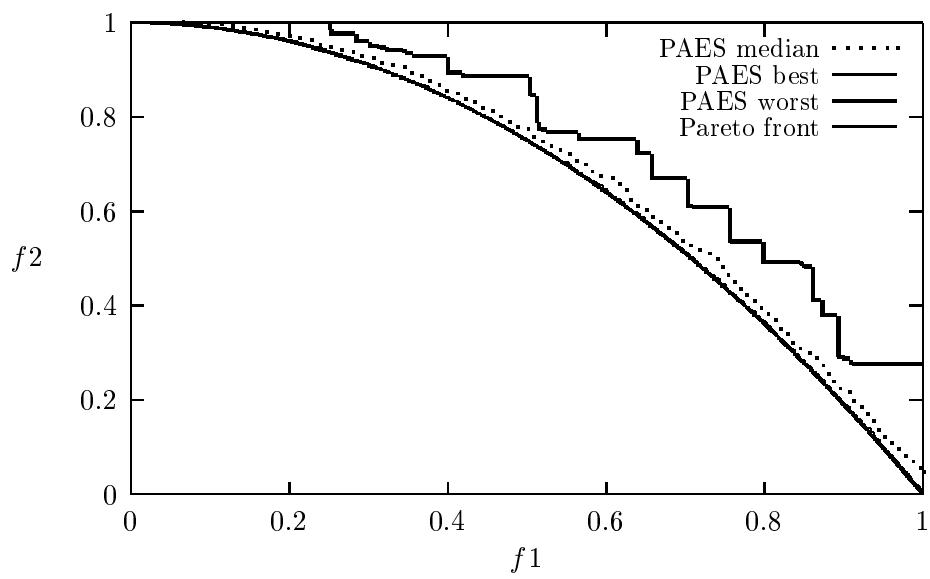


Figure 5.6: T2 PAES.

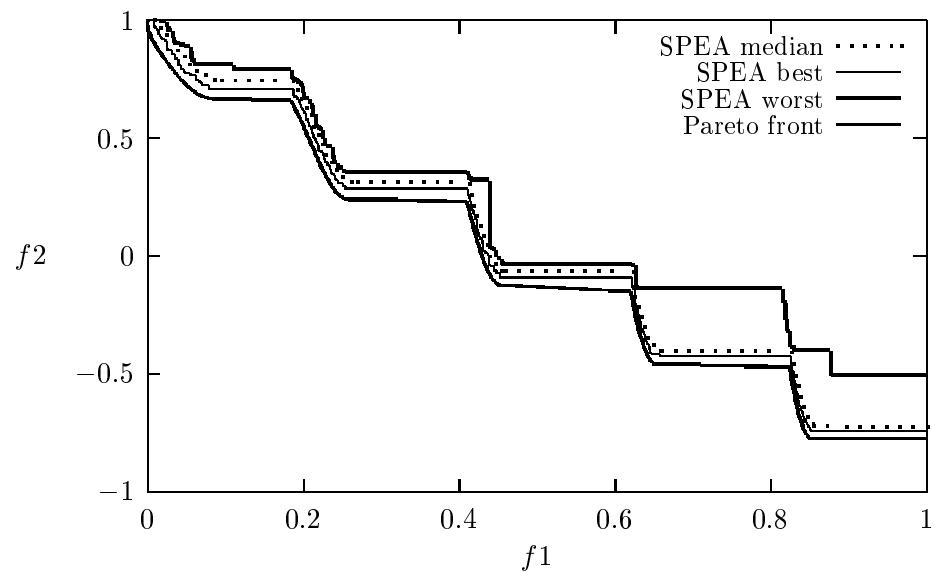


Figure 5.7: T3 SPEA.

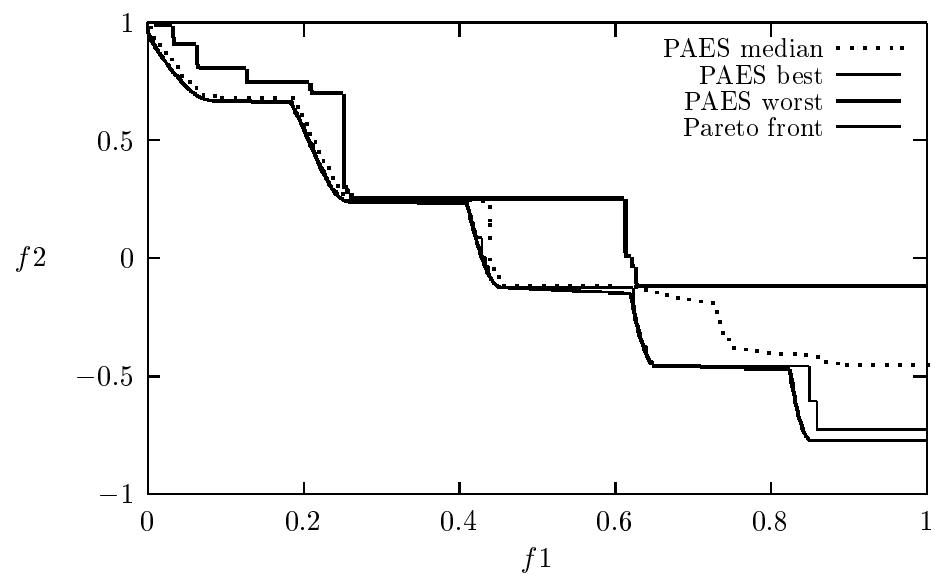


Figure 5.8: T3 PAES.

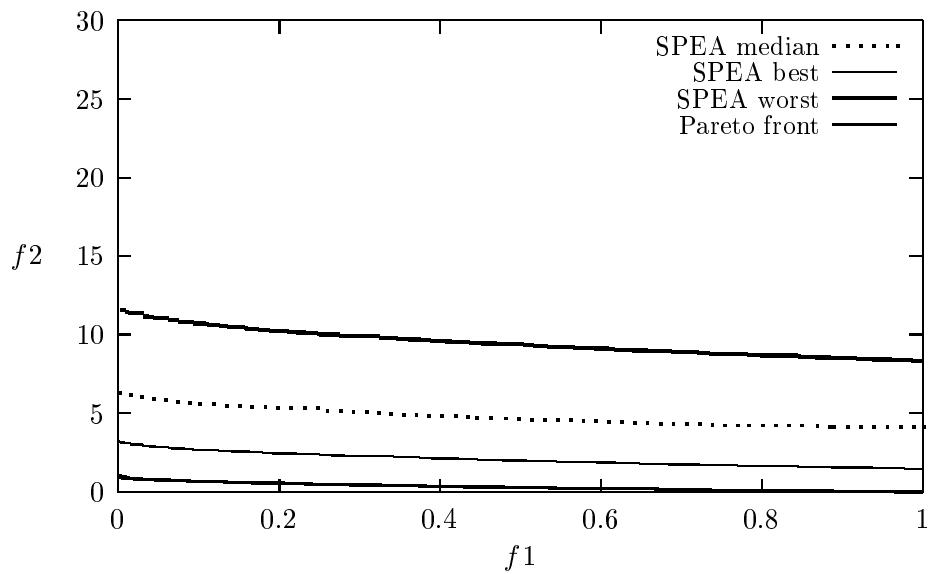


Figure 5.9: T4 SPEA.

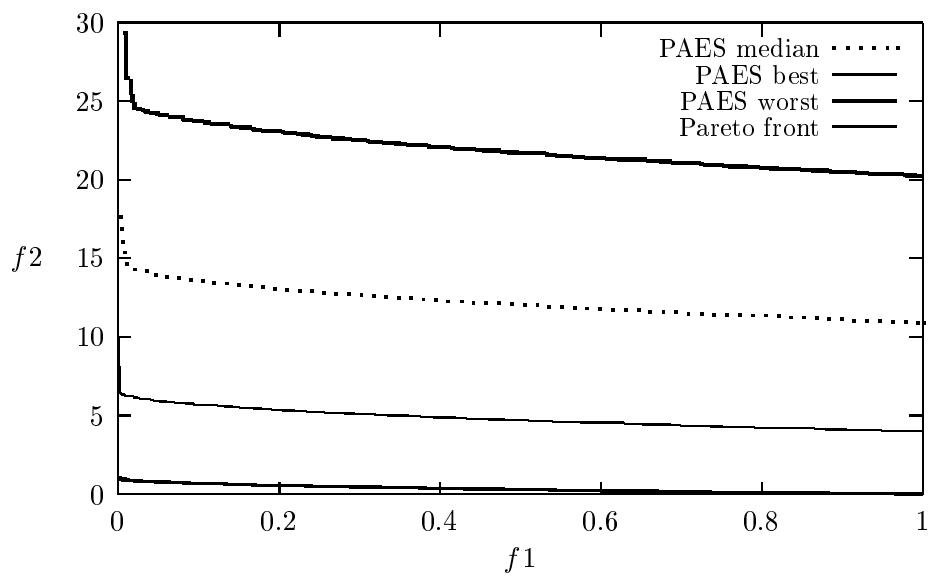


Figure 5.10: T4 PAES.

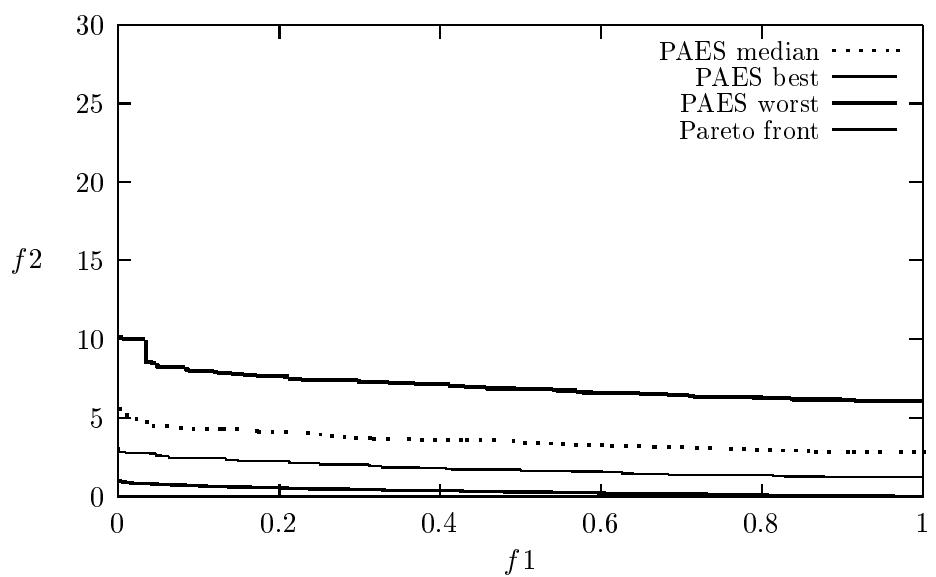


Figure 5.11: T4 PAES with a mutation rate of 8 bits per chromosome.

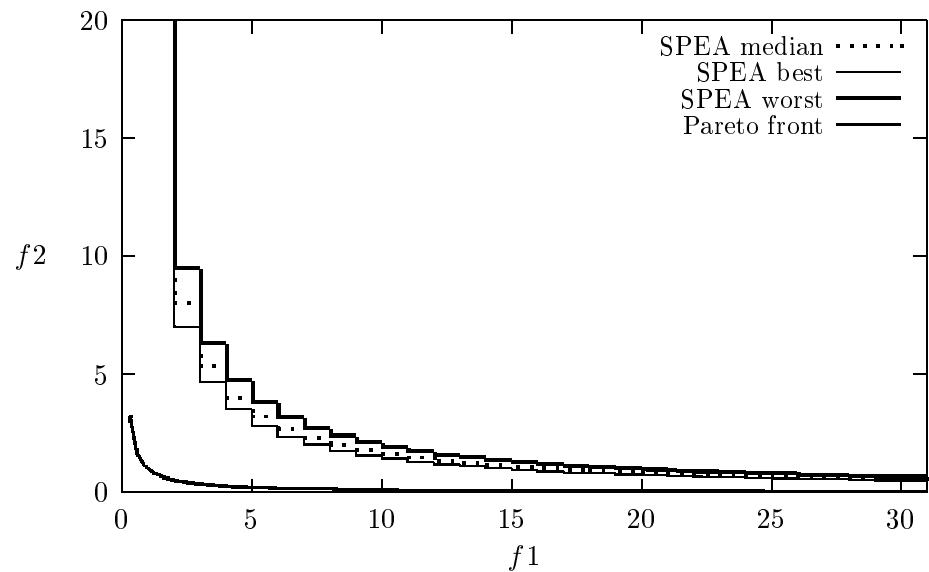


Figure 5.12: T5SPEA.

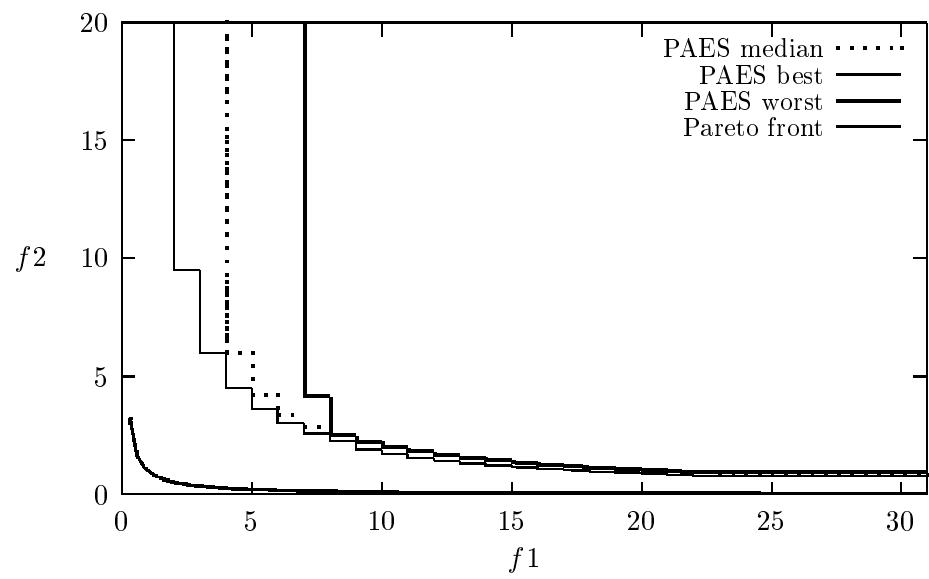


Figure 5.13: T5 PAES.

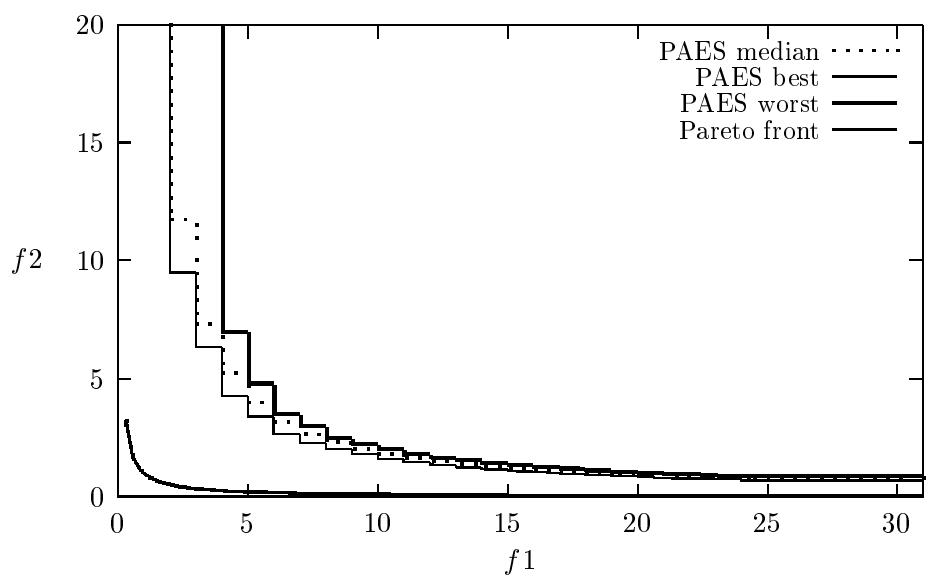


Figure 5.14: T5 PAES with a mutation rate of 4 bits per chromosome.

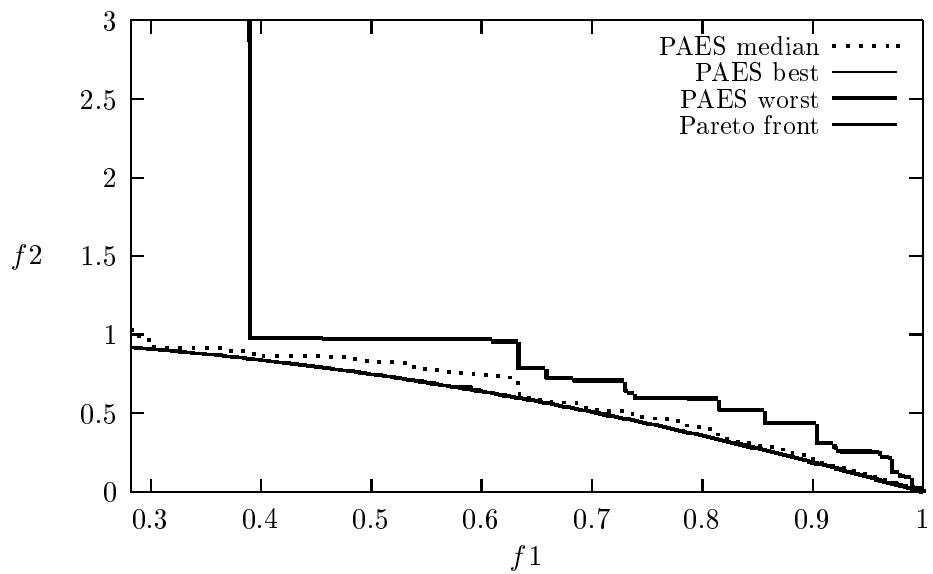


Figure 5.15: T6 PAES.

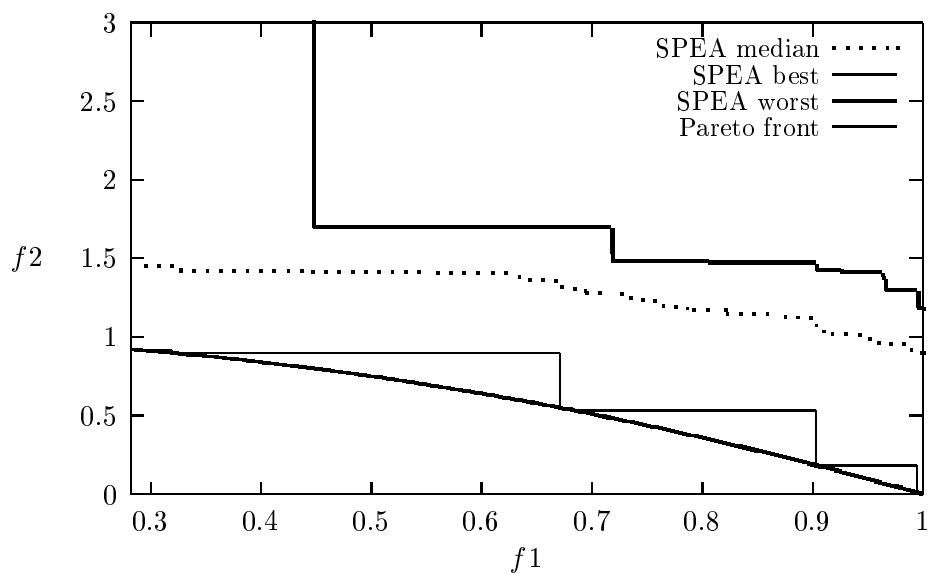


Figure 5.16: T6 SPEA.

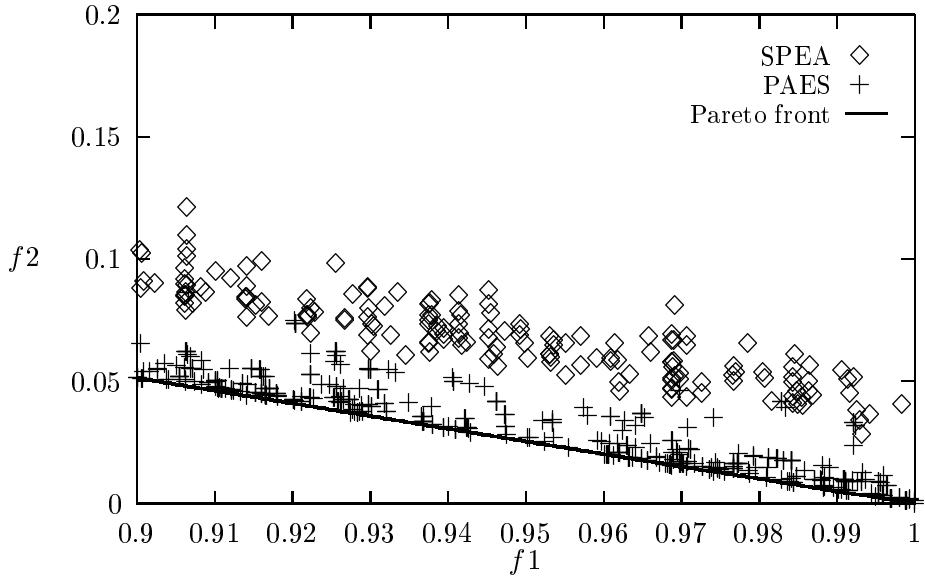


Figure 5.17: Solutions found from 30 runs of SPEA and PAES to test function  $\mathcal{T}_1$  with  $0.9 \leq f_1 \leq 1.0$ .

A mutation rate of 1% was used with each algorithm, on all problems in [ZDT00]. However, we choose to use a mutation rate (in our initial experiments) of 1 bit per chromosome because in previous experiments with PAES this setting has provided reasonable results. As the EAs in [ZDT00] all employ crossover, and mutation is only a secondary operator, we do not deem it necessary to choose the same mutation rate for PAES since its primary and only method of generating new points is through mutation. Nonetheless, we note that in this study we are unable to ‘tune’ the parameters of the MOEAs that we are using for comparison, and accept that this may affect the strength of our conclusions.

The number of subdivisions used in the adaptive grid archiving strategy of (1+1)-PAES was set to  $l = 5$ , giving 1024 grid regions. Although using this crowding method is not equivalent to the niching used by most of the EAs in [ZDT00], it is an integral part of the PAES algorithm and we submit that its use does not represent an unfair advantage, since we have previously shown that the crowding technique used by PAES is less computationally expensive than niching 4.4. In any case, the SPEA algorithm of Zitzler also uses a unique diversity maintenance technique: clustering.

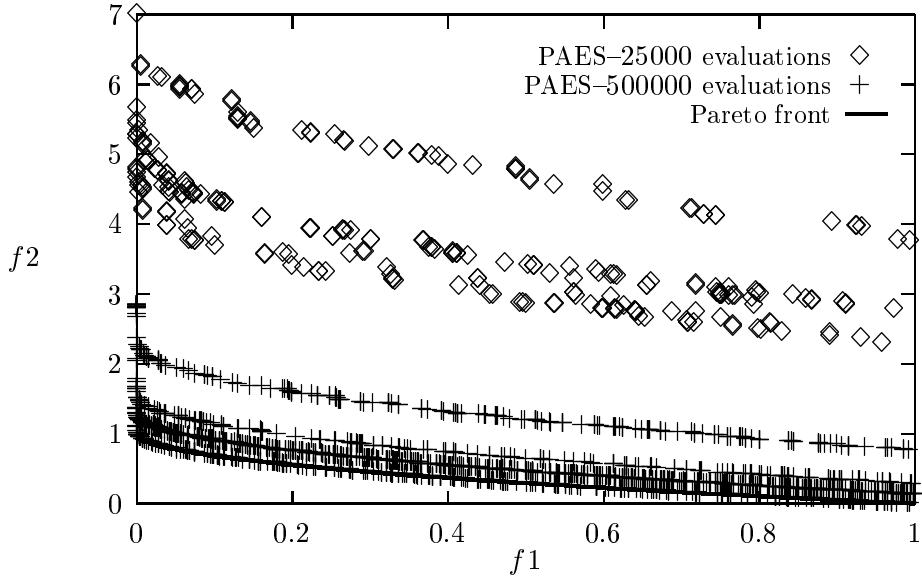


Figure 5.18: Solutions found to test function  $\mathcal{T}_4$  from 5 runs of PAES with 25000 evaluations per run and 500000 evaluations per run.

### Initial results

The results for these initial experiments are shown in Table 5.10. PAES<sub>on-line</sub> refers to runs in which the *on-line* performance of PAES was examined<sup>5</sup>. PAES[98] refers to PAES with an archive population of 98, and similarly for PAES[20]. PAES[98]Gray uses Gray coding of the parameters in place of binary. The results indicate that PAES outperforms SPEA on functions  $\mathcal{T}_1 - \mathcal{T}_3$  and  $\mathcal{T}_6$  regardless of its operating mode (with the exception of the Gray coded version). Using Gray codes adversely affects the performance of PAES on functions  $\mathcal{T}_1 - \mathcal{T}_3$  but causes PAES to outperform SPEA on function  $\mathcal{T}_4$ . Otherwise, PAES is outperformed by SPEA on problems  $\mathcal{T}_4$  and  $\mathcal{T}_5$ , the multimodal and deceptive problems respectively. Further investigation of these two problems is given below.

Unfortunately, the results in Table 5.10 do not indicate the *degree* to which PAES[98] and SPEA outperform each other on each sample line. Plots of their median, best and worst attainment surfaces (Figures 5.2–5.16), computed using the AS1 metric 3.12, give a clearer picture of the relative achievement of each algorithm. (Some of these plots relate to different mutation rates used in PAES, discussed below). On function  $\mathcal{T}_1$ , the results of the nearest competitor to SPEA in the Zitzler *et al.* study, NSGA, are also shown for comparison.

<sup>5</sup>The on-line version of PAES has an archive population of 100.

For completeness, Figure 5.17 shows all of the solutions returned by PAES[98] and SPEA over 30 runs on function  $\mathcal{T}_1$  over a small region of the front (for clarity only). This plot relates the attainment surface plots to a plot of points found, which the reader may be more familiar with.

### Mutation rate tuning

The results presented in the Table 5.10, and discussed above, were generated with no tuning of the mutation rate of PAES. On the multimodal test function  $\mathcal{T}_4$ , where SPEA was found to be superior to PAES using a mutation rate of 1 bit per chromosome, increasing the mutation rate was found to improve the performance of PAES. A mutation rate of 8 bits per chromosome or about 3% was found to give the following statistics [96.8, 0] when comparing PAES with SPEA, in favour of PAES (see also Figure 5.11).

On function  $\mathcal{T}_4$  a number of longer runs were also performed. The results of these show that PAES continues to find better solutions even after 500000 evaluations. A plot showing this is given in Figure 5.18, where 5 runs of PAES for 25000 evaluations are compared with 5 runs of PAES for 500000 evaluations. The mutation rate was 8 bits per chromosome or 2.67% in both sets of runs.

Improvements in performance were also found on  $\mathcal{T}_5$  with increased mutation rates. These are discussed in the next section.

On the other test functions, using a mutation rate of more than about 1% was found to be extremely detrimental to the performance of PAES, however. These results indicate that PAES is relatively sensitive to the mutation rate which is chosen and this can be problem dependent. Nonetheless, the relationship between the best mutation rate and the type of problem provides no surprises. Where the problem is unimodal, or at least not excessively rugged, a mutation rate of 1 bit per chromosome works well because large mutations are not needed to escape local optima, and this low mutation rate enables the evolving solutions to approach the Pareto front to a high accuracy. On the other hand, where the problem is multimodal and/or deceptive, higher mutation rates allow the exploration of a larger neighbourhood, giving a better chance of escape from solutions that would be local optima under a more restricted mutation operator. Thus, with minimal knowledge about the problem it may be possible to choose an appropriate mutation rate. When this knowledge is not available, a good strategy is to try out a mutation rate of 1 bit per chromosome, and to increase this by factors of two until performance drops off. Using this strategy, we were able to produce results better than those published for SPEA, on all but the deceptive function,  $\mathcal{T}_5$ , and using

only two or three trial runs of PAES.

### The deceptive function $\mathcal{T}_5$

Test function  $\mathcal{T}_5$  is a deceptive problem made up of 10 deceptive subproblems of 5 bits each, and a 30-bit non-deceptive problem. The 5-bit subproblems are all *fully deceptive* [Whi91] i.e. all schema of order four or less lead towards a sub-optimum which is the genotypic complement of the optimum. The performance of PAES on this problem compared to SPEA at a mutation rate of 1 bit chromosome (= 1.25%) is shown in Table 5.10, and indicates that the distribution achieved by SPEA is better than PAES in 100% of the space. However, the degree to which SPEA is better than PAES is not known. To investigate this, PAES was compared with the other algorithms tested in [ZDT00]. It was found to generate poorer distributions of solutions than SOEA, NSGA, HLGA [HL92] and VEGA but was slightly better than NPGA [HNG94] with a statistic of [39.1, 27.4]. We can conclude that with the initial mutation rate tried, PAES is significantly outperformed by SPEA on this test function, and does not compare favourably with other population-based EAs either. However, increasing the mutation rate to 4 bits per chromosome (=5.0%) increases the performance of PAES on this problem. At this mutation rate comparisons with VEGA and HLGA favour PAES - [100, 0.0] and [90.7, 0.0] respectively - while it is still beaten by NSGA and SPEA. With long runs of up to 5 million evaluations, the solutions found by PAES continue to improve, indicating that with this mutation rate at least, PAES does not converge prematurely. However, even with runs of this length PAES does not approach the performance of SPEA or NSGA using just 25000 evaluations; a statistic of [0.0, 37.9] resulted from comparing the solutions generated by 30 runs of PAES, for 5 million evaluations each, to the original data from SPEA running for just 25000 evaluations.

#### 5.3.3 Summary

In this study, a local search optimizer, PAES, has been compared with a modern, proven, population-based EA, SPEA, on a suite of six difficult test functions. With no tuning of the PAES algorithm to any of the test functions, and in several different operating modes, it was found that PAES generated solutions that were statistically significantly better than those generated by SPEA on four of the test functions. These functions are  $\mathcal{T}_1$  and  $\mathcal{T}_2$  with convex and non-convex Pareto fronts respectively,  $\mathcal{T}_3$  with a discontinuous Pareto-optimal front, and  $\mathcal{T}_6$  with both non-uniform density of the search-space and Pareto optimal solutions non-uniformly distributed along the global Pareto front; all 900 bit problems.

On the two remaining functions  $\mathcal{T}_4$  and  $\mathcal{T}_5$ , SPEA was found to generate solutions significantly better than those generated by PAES. These two test problems are difficult examples of multimodality and deception, respectively. The function  $\mathcal{T}_4$  has  $21^9$  local optima and consists of 300 bits, and  $\mathcal{T}_5$  has 10 fully deceptive 5-bit subproblems. Although, PAES was not competitive with SPEA on these two functions it was still better than NPGA and FFGA [FF93].

Furthermore, with tuning of the mutation rate of PAES, on  $\mathcal{T}_4$  the performance of SPEA was bettered. With longer runs at this mutation rate PAES was shown to evolve solutions to the global Pareto optimum. This result showed that on long runs, PAES has comparable performance to an elitist version of NSGA tested in [ZDT00]. On test function  $\mathcal{T}_5$  the performance of PAES was again improved with an increase in the mutation rate. However, the performance of SPEA on  $\mathcal{T}_5$  was not bettered even with runs of up to 5 million evaluations. Nonetheless, PAES does outperform all the other multiobjective EAs in [ZDT00] except for NSGA with the increased mutation rate, running for the same 25000 evaluations.

Experiments in which the mutation rate was changed showed PAES is sensitive to this parameter, and the best setting of it depends on problem type. However, the most appropriate mutation rate was found to conform to standard expectations of this: for highly rugged landscapes a high mutation rate is better, whereas for smoother landscapes a low mutation rate is more appropriate. Unfortunately, in this study it was not possible to experiment with the parameters of the other algorithms PAES was compared against. Therefore, we can't be sure that PAES would outperform tuned versions of SPEA and the other EAs on problems like  $\mathcal{T}_4$ , where it was initially worse before tuning of the mutation rate. However, on  $\mathcal{T}_4$ , we were able to replicate very similar levels of performance to experiments in [ZDT00] with long runs of NSGA using elitism and very large population sizes. This indicates that even in this highly multimodal function, PAES is probably competitive with most MOEAS, given the right mutation rate.

The use of Gray coding on function  $\mathcal{T}_4$  also caused a great improvement in the performance of PAES. Whitley [Whi99] has shown using NFL arguments that Gray coding induces fewer local optima than Binary over the set of all functions with fewer than  $2^L - 1$  optima. For a hillclimbing algorithm this may help substantially when the number of local optima is large, as in  $\mathcal{T}_4$ . However, when Gray was applied to the other problems in the test suite the performance of PAES was substantially reduced. In fact using Gray codes had much the same effect as increasing mutation rates. At present, the authors do not have a satisfactory explanation of this phenomenon, but simply note that when tackling multimodal functions, employing Gray coding and increasing mutation rates may be fruitful adjustments to experiment with.

### 5.3.4 Discussion

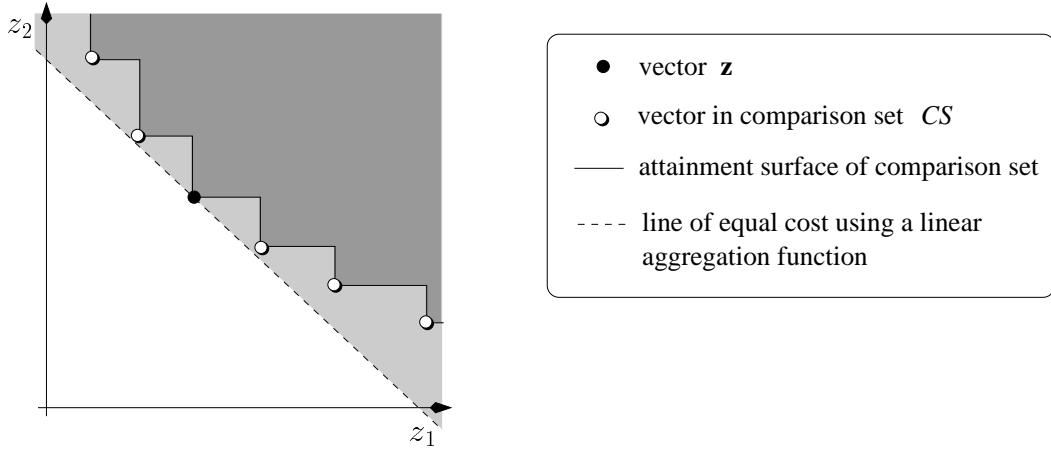


Figure 5.19: Illustrating how solutions of lower fitness using a linear aggregation of objectives may be of equal fitness when judged using a multiobjective approach with a comparison set of solutions.

The results summarised above lead us to the conclusion that tackling multiobjective problems, even those with specific features known to cause problems in proceeding to the Pareto front, or maintaining a diversity of solutions along that front, should not be the exclusive preserve of population-based evolutionary algorithms. In fact a simple hillclimbing or (1+1) selection strategy is sufficient to tackle even the difficult problems used in this section. Nonetheless, our conclusion cannot be as straight forward as this. It appears from our experiments that certain of the problems in the test suite did cause PAES more difficulty than some EAs, particularly SPEA. On others, PAES was substantially better. From this study it appears that PAES cannot be placed into the hierarchy put forward by Zitzler *et al.*, then. Rather, PAES seems to have difficulty with deceptive problems and perhaps multimodality. Other than this, it dominates all the algorithms in the Zitzler *et al.* study.

The empirical findings of this study suggest that local search may often prove a powerful heuristic to be used in multiobjective spaces. Furthermore, we postulate that assessing the quality of solutions using the notion of Pareto dominance may lead to fewer effective local optima than when fitness is assigned using a linear aggregated objective function. This effect, were it true, would mean that local-search Pareto optimizers would not be as susceptible to becoming trapped as standard local-search algorithms are, when applied to linearly-scalarized versions of multiobjective optimization problems. To see why this effect might be true, consider Figure 5.19. The diagram shows a solution  $\mathbf{z}$  in a two-objective space where each objective is to be minimized. A line  $l$  of equal cost under a linear aggregating function is

shown passing through  $\mathbf{z}$ . The region upper right of this line (shaded) are solutions of higher cost than  $\mathbf{z}$  under the aggregating function. In addition, a set  $CS$  of vectors which are nondominated with respect to  $\mathbf{z}$  and representing a comparison set against which candidate vectors are compared is also shown. The  $CS$  defines an attainment surface  $\mathcal{S}$  (shown with a solid line) which divides the space into two regions, the dominated region (darkly shaded) and the nondominated region. It can be seen that the ‘volume’ of solutions which are dominated with respect to  $CS$  (dark shaded region) is smaller than the volume of vectors that are costlier than  $\mathbf{z}$  under the objective aggregating function (total shaded region). So, if any of the neighbours of  $\mathbf{z}$  fell into the light-shaded portion of the shaded region, i.e. between  $l$  and  $S$ , and none of the neighbours of  $\mathbf{z}$  were cheaper than  $\mathbf{z}$  according to the aggregating function, then  $\mathbf{z}$  would be a local optimum under the aggregating function but not when judged using the multiobjective comparison set.

The above illustrates that using multiobjective comparison of solutions *can* lead to situations where there are fewer local optima. However, it is also possible to construct a counter-example in which there is less volume for solutions of equal or greater fitness under a multiobjective comparison set than under a linear aggregating function. To show statistically that, overall, fewer local optima are induced when multiobjective selection is employed than using objective aggregation will be a subject for future work. Nonetheless, we have already provided some strong empirical evidence for this effect [KWC01].

There are several other avenues for future research. It seems that PAES is sensitive to the mutation rate chosen and that the best setting is dependent on the type of problem. Some form of adaptive mutation rate may be examined to counteract this behaviour. The problems PAES exhibited with the deceptive function may indicate that large populations and crossover do have an important role to play in these kind of problems. This suggests that a memetic version of PAES that employs local search much of the time but which occasionally employs crossover, may also be a direction worth investigating.

## 5.4 Summary and conclusions

In this chapter, PAES was tested using two suites of test functions. On the first suite, PAES was compared against archiving and elitist versions of NPGA and NSGA. It performed favourably compared to these algorithms, and no significant weaknesses were found. In terms of the trade-off between solution quality and computation time, PAES was very strong. In terms of solution quality alone, PAES was, at worst, second amongst the algorithms tested, beaten only by the slowest of the other algorithms.

On the second test function suite, PAES was compared against the SPEA algorithm on six challenging and diverse test functions. Here, some weaknesses in the PAES algorithm were found. On the deceptive trap function  $\mathcal{T}_5$ , PAES could not compete with SPEA, and other MOEAs tested in the original study conducted by Zitzler *et al.*. However, this function is extremely biased towards recombination-based algorithms, since the bits were ordered on the string in such a way that recombination events can relatively easily combine the components together, particularly when 1-point crossover is used. With bit-reordering it is expected that the performance of PAES relative to the MOEAs would not be so poor. PAES also performed poorly on the highly multimodal function  $\mathcal{T}_4$ , using a default mutation rate of  $1/l$ . However, with a larger mutation rate PAES was able to compete successfully with the other MOEAs. Overall, the results from the second test function suite indicate fairly robust performance of PAES, and underline that it represents a strong baseline algorithm for multiobjective search.

No testing of PAES on problems with more than three objectives has been carried out in this chapter. In the next chapter, some four-objective knapsack problems are tackled with PAES, and these do not reveal that PAES has any difficulties with larger numbers of objectives. However, with five or more objectives, the efficiency and accuracy of PAES might be expected to deteriorate due to the need to have a very large archive. Few other Pareto MOEAs have been tested on problems of more than four objectives, however, and this remains an important area for future work.

Some variants of the standard (1+1)-PAES algorithm have also been proposed in this chapter. The population-based variants did not perform favourably compared to (1+1)-PAES, in general. The rules for updating the archive and performing selection were adapted from (1+1)-PAES to account for the multiple candidate solutions, but the new rules seem to be responsible for the poor performance of the population-based variants of PAES. The other variants of PAES, outlined in the previous chapter have not been tested here. This will be an important direction of future work.

In this chapter, the (1+1)-PAES algorithm has been extensively tested. Its level of performance, compared to modern MOEAs, has been shown to be sufficient for the purpose of baselining more sophisticated methods. Another of its potential uses, as a basis for designing hybrids of population-based and local-search algorithms, is the subject of the next chapter.

# Chapter 6

## The Memetic PAES Algorithm (M-PAES)

### 6.1 Overview

Memetic algorithms (MAs) combine the benefits of population-based EAs and local searchers in a single optimization method. The use of local search make MAs flexible with respect to incorporating problem-specific heuristics and specialized operators, while the population can exploit and recombine global information, and can help avoid suboptimal convergence. Overall, MAs, as a class, are very general-purpose and loosely-defined algorithms, although individually they tend to be highly tailored to the particular problem. It is this ‘tailor-ability’ that have given MAs the edge over other methods on many problems.

Of course, flexible methods, like MAs, should also perform well on Pareto optimization problems if they are tailored to the individual problem. Nonetheless, even a very flexible MA framework for Pareto optimization must specify how and which discovered solutions will be stored, and how the selection and acceptance functions (in the population-based and local-search phases, respectively) should operate. As with other multiobjective search methods, a central issue is whether to use Pareto dominance relations for evaluating solutions, or whether the components of the objective vector should be aggregated into a single value.

In this chapter, we discuss the implications of this design choice and suggest that an MA based on Pareto selection may exhibit quite different performance from recently proposed multiobjective MAs based on scalarization of objective values. With an eye to the Pareto local-search methods developed in Chapter 4, we specify a general framework for a Pareto

| Feature  | Pareto selection                              | Linear aggregation  |
|--|---|---|
| Efficient selection mechanism for GA populations | Proven  | Some evidence of poor performance   |
| Parallelizable                                   | Yes, but not completely independent processes | Yes   |
| Searches all directions simultaneously           | Yes   | No  |
| Able to find non-supported solutions             | Yes   | No  |
| Works with local search                          | With use of a comparison set as in PAES, only | Yes   |
| Range/scaling and reference point independent    | Yes   | No: requires the ranges to be equalized and the utopian point for effective use |

Table 6.1: Advantages and disadvantages of Pareto selection versus Linear aggregation.

MA. This framework is used to specify an algorithm, called memetic-PAES (M-PAES), which makes use of our adaptive grid archiving strategy, and incorporates a local-search subroutine based on (1+1)-PAES.

In the remainder of the chapter, M-PAES is tested against three modern, but fundamentally different MOEAs: the evolutionary algorithm, SPEA; the memetic algorithm, RD-MOGLS; and our baseline hillclimber approach, (1+1)-PAES. In these experiments, we employ non-specialized operators in M-PAES to verify its general performance and to ensure judicious comparison between it and the other algorithms. In the next chapter, we investigate the use of M-PAES again, for tackling a particular class of problems, but in the studies presented there we also design and exploit specialized encodings and associated operators.

## 6.2 Recent multiobjective memetic algorithms

Recently, two different (but similar) memetic algorithms for Pareto optimization, have been proposed; one by Ishibuchi and Murata [IM96], and the other by Jaszkiewicz [Jas98]. Both of these MAs use scalarizing selection (see Section 3.4.3) in both the reproduction and local-improvement search phases. In this section, we briefly consider how the use of scalarizing selection might affect the performance of these MAs, as compared with an MA using Pareto selection.

In both the proposed MAs, the weighting vectors, specifying the ‘direction’ of search in objective space, are chosen *at random*, at each iteration, without regard to the solutions that have already been found. This may be detrimental to obtaining an even distribution of points along the nondominated front because in many cases, it will not be sufficient to select weights in all directions with equal probability. Rather, because some areas of the Pareto front may be easier to obtain than others, the algorithm may waste time searching for solutions in areas where it has already found many solutions, while leaving other regions relatively unsampled. Even when the distribution of points in the true Pareto front is even, the MA may obtain a greater density of solutions in some areas than in some others, just via stochastic sampling effects. For these reasons, some form of adaptive scheme may be a better solution, and Pareto EAs that use niching mechanisms would certainly seem less susceptible to these problems.

Furthermore, these two MAs both perform local search on the offspring in the same direction as was used for selecting the parents. This unidirectional searching means that a solution generated during the local-search phase may be rejected because it scores poorly on the incumbent weighting vector, even though it dominates one or more of the best solutions already found, and may be an excellent solution when evaluated according to a *different* weighting vector. In Pareto selection methods, there is more efficiency in this regard because solutions are not usually discarded if they dominate solutions in the discovered nondominated front. Effectively, Pareto selection methods search in all directions at once. More *comparisons* between solutions are required in Pareto selection methods, but they may be more efficient in terms of the number of function evaluations required to obtain a given quality of non-dominated set. Whether it is preferable to have lower computational overhead or to reduce the number of function evaluations will depend on the overhead of evaluating solutions in the given application, but Pareto-selection MAs and MAs based on scalarizing selection will perform *differently* in this respect.

Further to the potential problems with the use of scalarizing selection employed in these two MAs, just outlined, there is also some empirical evidence that weighted aggregation methods are not as effective as Pareto selection, in population-based approaches. In [ZDT00], an elitist EA using linear weighted aggregation of objectives, SOEA, was compared with other elitist MOEAs, including SPEA. Even though the SOEA was run multiple times with different weight vectors, using many more function evaluations than SPEA, it did not achieve results of the same quality, except on one out of six test functions. Furthermore, an elitist version of Hajela and Lin’s weighted-sum GA, (*HLGA*\* [ZDT00]) performed less effectively than elitist MOEAs based on Pareto selection. Also, in work by Knowles *et al.* [KWC01], some evidence that Pareto selection-based methods were shown to be more effective than scalar selection methods, even on problems that only *naturally* possess one objective.

Another issue relates to non-convex Pareto fronts or regions of the Pareto front. Pareto methods are able to find non-supported efficient solutions where linear scalarizing selection methods tend to miss these solutions. Although non-linear (e.g. Tchebycheff) scalarization is possible, some experiments [BPH98] have found this to be less effective for search than linear scalarization in one application.

Scalarizing methods may yet have an advantage in parallel MA implementations, however, because local-search phases could be independently run on different processors with no need for communication between them. This can be achieved less easily with Pareto MAs based on using an archive as a comparison set (as in PAES) because the archive must be updated for each parallel search to be carried out. This requires communication to some shared memory, and so is not so readily parallelizable, or as efficient in this regard.

We give a summary of the possible advantages and disadvantages of Pareto selection compared with linear aggregation in Table 6.1. The table entries indicate that a multiobjective MA based on Pareto selection will have quite different features from the multiobjective MAs currently available, which are based on scalarizing selection. In particular, the proven ability of Pareto selection for use in GA populations warrants investigation in the context of a multiobjective MA.

### 6.3 Pareto memetic algorithm design

The algorithms presented in Chapter 4 facilitate the design of new Pareto memetic algorithms which rely entirely on Pareto selection. These MAs could use the procedures developed for PAES to perform local improvement of solutions, avoiding some of the problems associated with scalarizing selection. This method of carrying out local search could be naturally complemented by the use of elitist Pareto selection methods, as employed in some MOEAs, for selecting parents in the reproduction phases.

Additionally, it would be desirable for the MA's local-search phases to be somewhat independent so that they could be carried out in parallel. However, we would want these separate local-search phases to contribute to a global archive of solutions so that the best solutions from the whole run would be available at the end. We have already seen in Chapter 4 that an algorithm that uses multiple, independent Pareto selection-based local searches, and also returns a diverse nondominated set representing the best solutions from the whole run, requires two archives, as in the multi-start PAES algorithm.

Using these general ideas we formulate a rudimentary outline of a memetic algorithm for

Pareto optimization, following the general design of the local-search-based MA described in [Mos99]. This outline algorithm is shown in Figure 6.1.

The outline Pareto MA given in Figure 6.1 leaves a wealth of algorithm design choices unspecified. In particular, we can identify the following list of remaining design choices or free variables:

- what global stopping criteria are used;
- what stopping criteria are used for each local improvement procedure;
- what the local improvement procedure is, e.g. PAES, multi-start PAES, annealing PAES, etc., or some other Pareto local-search procedure;
- how the size of  $G$  and  $H$  and  $P$  should be controlled, and/or whether each should be of constant or variable size;
- what procedures should be used for archiving in  $G$  and  $H$ , and whether *all* solutions should be checked for entry into  $G$ ;
- whether the pool of parents should be restricted to individuals in  $P$ , or  $G$ , or parents should be chosen from  $P \cup G$ ;
- whether multi-parent recombination is allowed, and how this is controlled;
- whether mutation is included as a separate process from local improvement, as it is in the outline algorithm;
- whether mating restrictions are used;
- whether  $P$  is maintained as a mutually nondominated set;
- whether elitist selection and replacement are employed;
- how many new solutions are generated per ‘generation’, i.e. whether there is a generation gap;
- and whether the algorithm has multiple restarting capability.

Initially, we would like to put the following restrictions on these choices, to reduce the number of design variables, and to allow initial testing and comparison with MOEAs to be fairly and easily carried out.

---

**Algorithm:** Pareto MA Outline

---

*Data:*

$P$  is the population of solutions  
 $P'$  an intermediate population  
 $G$  is the global archive of solution vectors  
 $H$  is the local archive of solution vectors  
 $X_{par}$  is a set of ‘parent’ solutions  
 $\mathbf{x}^{off}$  is an ‘offspring’ solution  
 $\mathbf{x}'$  is a mutant solution

*Functions:*

$\text{Update}(X, \mathbf{x})$  updates the set  $X$  with  $\mathbf{x}$  according to some set of rules  
 $\text{Recombine}(X_{par})$  returns a solution vector from recombination of the vectors in  $X_{par}$   
 $\text{Reset}(H, \mathbf{x})$  resets  $H$  such that it contains a set of solutions from  $G \cup P$  that will help exert selection pressure on the solution  $\mathbf{x}$  to be improved using  $\text{PLS}(\mathbf{x}, H, G)$   
 $\text{PLS}(\mathbf{x}, H, G)$  is some Pareto local-search procedure for improving  $\mathbf{x}$  using and updating  $H$  to exert selection pressure, and also updating  $G$  with all solutions encountered

---

```
 $G \leftarrow \emptyset$ 
 $\text{Init}(P)$ 
 $P' \leftarrow \emptyset$ 
foreach  $\mathbf{x} \in P$ 
   $\text{Update}(G, \mathbf{x})$ 
   $\text{Reset}(H, \mathbf{x})$ 
   $\text{PLS}(\mathbf{x}, H, G)$ 
   $P' \leftarrow P' \cup \{\mathbf{x}\}$ 
 $P' \leftarrow P$ 
repeat
   $P' \leftarrow \emptyset$ 
  parfor  $j \leftarrow 1$  to  $\#\text{recombinations}$  do
    Select a set  $X_{par} \subseteq P \cup G$  to be parents
     $\mathbf{x}^{off} \leftarrow \text{Recombine}(X_{par})$ 
     $\text{Reset}(H, \mathbf{x}^{off})$ 
     $\text{PLS}(\mathbf{x}^{off}, H, G)$ 
     $P' \leftarrow P' \cup \{\mathbf{x}^{off}\}$ 
  endparfor
  parfor  $j \leftarrow 1$  to  $\#\text{mutations}$  do
    Select  $\mathbf{x} \in P$  for mutation
     $\mathbf{x}' \leftarrow \text{Mutate}(\mathbf{x})$ 
     $\text{Reset}(H, \mathbf{x}')$ 
     $\text{PLS}(\mathbf{x}', H, G)$ 
     $P' \leftarrow P' \cup \{\mathbf{x}'\}$ 
  endparfor
   $P \leftarrow \text{Selectfrom}(P \cup P')$ 
  if  $P$  has converged then  $P \leftarrow \text{Restart}(P)$ 
until termination condition is TRUE
```

---

Figure 6.1: An outline for a Pareto memetic algorithm with local archive  $H$ , global archive  $G$ , and population  $P$ . Notice that mutation is included in each generation, in addition to local improvement. Notice also the algorithm’s restart capability.

1. The overall stopping criterion should be the maximum number of function evaluations, allowing straightforward comparisons with other algorithms.
2. The adaptive grid archiving procedure developed for PAES should be used to control both  $G$  and  $H$ , and  $G$  and  $H$  should have a bounded capacity;
3.  $P$  should be of constant size.
4. Standard local search (mutation) and recombination operators only should be used to allow straightforward comparison with other MOEAs.
5. Mutation should not be included as a separate process from local search/improvement.
6. No restart facility should be included.
7. The mating pool of solutions should be  $P \cup G$ .
8. The number of newly generated solutions in each recombination phase should be  $|P|$  i.e., the recombination phase is generational.

With these restrictions we can specify the Pareto MA framework shown in Figure 6.2. The key defining features of the framework are: there are separate single-point local-search and population-based search phases; all comparisons to determine acceptance/rejection of solutions are Pareto comparisons; the archives  $G$  and  $H$  store mutually efficient solutions;  $H$  is initialized for each local search phase;  $H$  is used to determine the acceptance of local-search moves;  $G$  is updated for every new solution generated; in the population-based phase,  $G$  is used to determine acceptance of the offspring.

The method used to generate solutions in the local search phase can be any operator based on perturbing a *single* solution. The recombination method can be any operator based on generating solutions using information from two or more solutions. The initialization of the archive  $H$  can be achieved using any solution already stored in  $P$  or  $G$  and should encourage an effective local improvement of the current solution  $\mathbf{x}$ .

Elitism is inherent in the framework, and it can be controlled in several ways. Each time a solution is compared with either the local archive  $H$ , or the global archive  $G$ , to determine whether to accept it or not, a sample of solutions from the archive is used. By adjusting the size of the sample, the degree of elitism can be controlled. The acceptance functions themselves are not defined in the framework, either. What *is* specified is that all comparisons used to accept or reject solutions are Pareto comparisons.

The framework leaves the following design choices open to investigation.

1. Generate a population  $P$  of solutions using some initialization method.
2. Store the internally efficient solutions from  $P$  in a global archive  $G$ .
3. Repeat until a stopping criteria is satisfied:
  - (a) For each candidate solution  $\mathbf{x}$  in  $P$ :
    - i. Initialize a local archive of solutions  $H$ .
    - ii. Perform a local search from  $\mathbf{x}$ . At each step of the local search:  
Generate neighbour solution(s)  $\mathbf{x}^n$ ,  $n = 1..N$  from  $\mathbf{x}$ .  
Compare  $\mathbf{x}^n$  with one or more solutions from  $H$  to determine which (if any) of  $\mathbf{x}^n$  is accepted.  
Update current solution:  $\mathbf{x} \leftarrow$  accepted solution.  
Update  $H$  with  $\mathbf{x}^n$  if applicable.  
Update  $G$  with  $\mathbf{x}^n$  if applicable.
    - iii. Place improved solution  $\mathbf{x}$  back in  $P$ .
  - (b) Set new population  $P' \leftarrow \emptyset$
  - (c) Perform recombination of solutions in  $P$  and  $G$  to obtain new population  $P'$  by repeating the following until  $|P'| = |P|$ :
    - i. Select parents from  $P \cup G$ .
    - ii. Recombine to form offspring  $\mathbf{x}$ .
    - iii. Compare  $\mathbf{x}$  with one or more solutions from  $G$  to determine whether to accept  $\mathbf{x}$  ( $P' \leftarrow P' \cup \{\mathbf{x}\}$ ), or reject it.
    - iv. Update  $G$  with  $\mathbf{x}$  if applicable.
  - (d) Update population:  $P \leftarrow P'$ .
4. Return global archive  $G$  of unique mutually efficient solutions.

Figure 6.2: A Pareto MA framework.

1. What local-search procedure is used.
2. What stopping criteria should be used for controlling the local-search phase.
3. Whether/how elitist selection is incorporated into the recombination phase.
4. Whether mating restrictions should be used in the parent selection in the recombination phase.

It is expected that the multiobjective MA framework will be most effective when very good local-search heuristics are available for the particular multiobjective problem. However, to test its general applicability we will first use the framework to define a very generally applicable memetic algorithm: M-PAES, that does not incorporate any specialized operators, and is based on the simplest local-search algorithm developed in the previous chapter: (1+1)-PAES.

## 6.4 Development of M-PAES

Having restricted the first of the design choices given above, that is, to the use of a local-search procedure based on the simple (1+1)-PAES algorithm, there are three remaining design choices to make. The first of these to consider is the stopping criteria to be used for the PAES procedure. Previously, PAES has been terminated when a maximum number of function evaluations is reached. This simple criterion *could* be transferred to the PAES procedure but this choice would not be in the spirit of how the local improvement subroutine in a memetic algorithm is supposed to work. Ideally, the local improvement subroutine should be able to detect when a local optimum is reached and stop when this has occurred. However, the intention here is to use a standard mutation operator as the neighbourhood operator. Strictly, this means that *all* points in the search space can be visited from all other points (for example with a bit mutation rate of  $1/L$ ). Thus the local optimum is strictly the global optimum of the search space, and it would not be beneficial for the PAES procedure to wait until one of these were found. In any case, in general, it may not be possible to detect when a local (global) optimum has been found. These considerations mean that alternative criteria should be used.

Although true local optima may not exist given the use of a standard mutation operator, ‘approximate’ local optima can normally be monitored by observing whether the evaluation of the solutions visited stops improving for a ‘long’ time, or the rate of change of evaluation falls below a threshold value. Now, in PAES, evaluation is a vector so this is not so straightforward. However, the following means of detecting approximate local optima are proposed:

1. Count the number of consecutive PAES iterations that result in no changes to the (local) archive, and stop when this number reaches a threshold.
2. Count the number of consecutive PAES iterations that result in the candidate solution being rejected, and stop when this reaches a threshold.
3. Count the number of PAES iterations in which the candidate is dominated by the archive, resetting the counter to zero if the candidate is accepted as the new current solution. Stop when the count reaches a threshold.
4. Calculate the rate of change of the (local) archive, by measuring the *discard* rate from the archive. That is, the number of solutions that are discarded from the archive due to the entry of new candidate solutions, per iteration. This could be calculated as a moving average over a moving window of iterations. Initially, when the archive is not full the discard rate may be zero. However, once the archive becomes full the discard rate should increase, if new solutions are being archived. Once a local optima is reached the discard rate will fall again. The PAES procedure could be terminated if the average discard rate has previously exceeded a threshold, and then fallen below this threshold again. The discard rate should be used rather than the entry rate because this implicitly takes into account the quality of newly archived solutions. For if a very good solution is found, it may dominate several members of the archive, causing all of them to be discarded, and increasing the measured rate of change of the archive.

Some investigation of the first three of these different stopping criteria was carried out in initial development of the M-PAES algorithm. It was found that 3. was most effective. However, the use of this criterion in isolation, led to some poor behaviour. If the threshold is set too low, PAES does not search adequately before stopping. However, if it is set too high, the PAES procedure can run indefinitely, causing the advantages of using the population and recombination to be lost. Thus, the stopping criterion was found to work best in partnership with a maximum number of iterations parameter.

The fourth stopping criterion outlined above has not been investigated to date. This may be better than the other proposed criteria because it works with a (discrete) rate of change rather than a counter. However, it is complicated and requires two parameters to be set - the size of the moving window, and the threshold.

In the recombination phase, two further design choices must be made. First, is how elitism should be incorporated into the reproductive selection, if at all. That is, whether newly generated offspring should always be accepted in  $P'$  (and should eventually replace their parents in  $P$ ), or whether their entry into  $P'$  should be controlled by comparing the solutions

to their parents, and/or to solutions in  $G$ . This was investigated in the early development of M-PAES, and it was found that elitist reproductive selection was generally beneficial. The second design choice, is whether mating restrictions should be used. This is very much a problem-dependent choice. However, in M-PAES, mating restriction was tried on the 0/1 knapsack problem (see Section 6.6.1), and it was found to be detrimental to the performance of the algorithm. Therefore, we do not include it in the M-PAES algorithm, described below.

## 6.5 The M-PAES algorithm

The memetic-PAES algorithm (M-PAES) is shown in pseudocode in Figure 6.3. It is based on the local-search multiobjective algorithm, (1+1)-PAES, and the framework and design choices discussed above. The archiving of solutions in M-PAES is a little more complicated than in (1+1)-PAES. Recall that at the heart of PAES is a procedure for maintaining a finite sized archive of internally efficient solutions. The solutions in the archive are representative of the best nondominated solutions found by the algorithm as it searches the space. The solutions in the archive serve a dual purpose in (1+1)-PAES: as a memory of the solutions found during the run for presentation at the end; and as a comparison set to aid in estimating the dominance rank of new candidate solutions. In order that these same jobs are performed in M-PAES, two archives are required. This is because each local-search phase needs to be partially independent of the global search being performed by the algorithm as a whole. Thus we have a global archive  $G$  that maintains a finite set of internally efficient solutions found, and a local archive  $H$  that is used as the comparison set in each of the local-search phases. At the beginning of a local-search phase,  $H$  is cleared and filled with solutions from  $G$  which do not dominate the candidate solution  $\mathbf{x}$ . The archive  $H$  is then used as in (1+1)-PAES to improve  $\mathbf{x}$ , i.e.  $H$  is maintained and used as a comparison set, while  $G$  is continually updated but plays no part in the estimation of the quality of new solutions.

The PAES local-search procedure used by M-PAES to improve solutions in  $P$  is almost the same as the basic (1+1)-PAES algorithm. However, it differs in the way that termination of the procedure is determined. Termination may be invoked when either of two conditions are fulfilled: (1) If the maximum number of local-search moves  $l_{opt}$  is exceeded. (2) If the maximum number of local-search fails  $l_{fails}$  is exceeded. To achieve (2), the variable  $\#fails$ , initially zero, is incremented every time the mutant is dominated by the current solution. It is reset to zero every time a move occurs i.e. when the mutant is accepted as the new current solution. Hence,  $\#fails$  effectively counts the number of potentially detrimental moves between improving moves. If this number exceeds the threshold  $l_{fails}$ , the local search

---

**Algorithm: M-PAES**


---

*Data:*

$P = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{|P|}\}$  is the population of solutions  
 $\mathbf{x}^i$  is the  $i$ th member of  $P$   
 $G$  is the global archive of solution vectors  
 $H$  is the local archive of solution vectors  
 $t$  counts the number of objective function evaluations  
 $\#retrials$  counts the number of recombination trials  
 $recomb\_trials\_max$  is the maximum number of recombination trials

*Functions:*

$\text{Terminate}(t, G)$  returns TRUE if the stopping criterion of M-PAES is satisfied  
 $\text{Grpop}(\mathbf{x})$  returns the population of  $\mathbf{x}$ 's grid region  
 $\text{Update}(G, \mathbf{x})$  updates  $G$  with  $\mathbf{x}$  according to the adaptive grid archiving strategy  
 $\text{PAES}(\mathbf{x}, G, H, t, l\_fails, l\_opt)$  is a (1+1)-PAES subroutine shown in Figure 6.4  
 $\text{Recombine}(\mathbf{x}^1, \mathbf{x}^2)$  returns a solution vector from recombination of vectors  $\mathbf{x}^1$  and  $\mathbf{x}^2$   
 $\text{Rand}(A)$  returns a solution, uniformly randomly selected from the set of solutions,  $A$

---

```

 $t \leftarrow 0$                                      /* Initialization */
 $G \leftarrow \emptyset$ 
 $P \leftarrow \emptyset$ 
foreach( $i \in 1..|P|$ )
   $\mathbf{x} \leftarrow \text{Init}()$ 
   $P \leftarrow P \cup \{\mathbf{x}\}$ 
   $\text{Update}(G, \mathbf{x})$ 
   $t \leftarrow t + 1$ 
while( $\text{Terminate}(G, t) \neq \text{TRUE}$ )
  foreach( $i \in 1..|P|$ )                         /* Local-search phase */
     $H \leftarrow \emptyset$ 
     $H \leftarrow H \cup \{\mathbf{x} \mid \mathbf{x} \notin \mathbf{x}^i, \mathbf{x} \in G\}$ 
     $H \leftarrow H \cup \{\mathbf{x}^i\}$ 
     $\mathbf{x}^i \leftarrow \text{PAES}(\mathbf{x}^i, G, H, t, l\_fails, l\_opt)$ 
   $P' \leftarrow \emptyset$ 
  while ( $|P'| < |P|$ )                         /* Recombination phase */
     $\#retrials \leftarrow 0$ 
    do
       $\mathbf{x}^{p1} \leftarrow \text{Rand}(P \cup G); \mathbf{x}^{p2} \leftarrow \text{Rand}(P \cup G)$ 
       $\mathbf{x} \leftarrow \text{Recombine}(\mathbf{x}^{p1}, \mathbf{x}^{p2})$ 
       $\text{Update}(G, \mathbf{x})$ 
       $t \leftarrow t + 1$ 
       $\#retrials \leftarrow \#retrials + 1$ 
    while ((( $\mathbf{x} > G$ )  $\vee$  (( $\text{Grpop}(\mathbf{x}) > \text{Grpop}(\mathbf{x}^{p1})$ )  $\wedge$  ( $\text{Grpop}(\mathbf{x}) > \text{Grpop}(\mathbf{x}^{p2})$ )))  $\wedge$ 
      ( $\#retrials < recomb\_trials\_max$ ))
    if ( $\mathbf{x} > G$ )
      if ( $\text{Grpop}(\mathbf{x}^{p2}) < \text{Grpop}(\mathbf{x}^{p1})$ )
         $\mathbf{x}^{p1} \leftarrow \mathbf{x}^{p2}$ 
         $\mathbf{x} \leftarrow \mathbf{x}^{p1}$ 
       $P' \leftarrow P' \cup \{\mathbf{x}\}$ 
     $P \leftarrow P'$ 
  return  $G$ 

```

---

---

**Procedure:** PAES( $\mathbf{x}, G, H, tl\_fails, l\_opt$ )

---

*Data:*

$G$  is the global archive of solution vectors, passed as a parameter to the procedure  
 $H$  is the local archive of solution vectors, passed as a parameter to the procedure  
 $\mathbf{x}$  is the current solution vector, initially passed as a parameter to the procedure  
 $\mathbf{x}'$  is the mutant solution vector  
 $l\_fails$  is the maximum number of consecutive dominated mutants, a parameter  
 $l\_opt$  is the maximum number of local optimization iterations, a parameter  
 $\#fails$  counts the number of consecutive dominated mutants  
 $\#moves$  counts the number of local optimization iterations

*Functions:*

Terminate( $t, G$ ) returns TRUE if the stopping criterion of M-PAES is satisfied  
Mutate( $\mathbf{x}$ ) returns a neighbour of  $\mathbf{x}$   
Grpop( $\mathbf{x}$ ) returns the population of  $\mathbf{x}$ 's grid region  
ND( $X$ ) returns the nondominated vectors from a set  $X$   
Update( $G, \mathbf{x}$ ) updates  $G$  with  $\mathbf{x}$  according to the adaptive grid archiving strategy

---

```

#fails ← 0                                     /* Initialization */
#moves ← 0
while ((#fails < lfails) ∧ (#moves < lopt) ∧ (Terminate(t, G) ≠ TRUE))           /* Main Loop */
    t ← t + 1
    #moves ← #moves + 1
    x' ∈ X ← Mutate(x)
    if (x' < x)
        x ← x'; #fails ← 0
        H ← ND({x'} ∪ H)
        Update(G, x)
    else if (x' ∕ H)
        Update(G, x)
        if (x' < H)
            H ← ND({x'} ∪ H)
            x ← x'; #fails ← 0
        else if (x' ∼ H)
            if (|M_{t-1}| < arcsize)
                H ← {x'} ∪ H
            else if (x' increases the extent of the grid)
                H ← {x'} ∪ H \ {x^{cr}} where x^{cr} is one
                randomly selected solution from the
                set {x^i ∈ M_{t-1} | ∀x^j ∈ M_{t-1}, Grpop(x^i) ≥ Grpop(x^j)}
            else if (∃x^i ∈ M_{t-1} such that Grpop(x') < Grpop(x^i))
                H ← {x'} ∪ H \ {x^{cr}}
            if (Grpop(x') < Grpop(x))
                x ← x'; #fails ← 0
        else
            #fails ← #fails + 1
    return x                                     /* End procedure */

```

---

Figure 6.4: Pseudocode for procedure PAES( $\mathbf{x}, G, H, t, l\_fails, l\_opt$ ).

is stopped. The local-search procedure  $PAES(c, G, H)$  is shown in Figure 5.19.

In the recombination phase, parents are randomly selected from the union of the post local-search population, and the global archive. The resultant child is accepted only if it is nondominated with respect to the entire global archive, and it resides in a less crowded grid region than at least one of its parents. If it dominates any member of  $G$  it is naturally accepted too. However, solutions that are dominated by member(s) of  $G$ , or that reside in crowded regions are rejected. In this case two new parents are selected again and recombination is applied once more. The procedure is repeated until either a child is accepted or a threshold number of recombinations  $recomb\_trials\_max$  is exceeded. In the latter case, the current parent in the lesser crowded region joins the intermediate population  $P'$ . The recombination strategy is, as a whole, extremely elitist, following the general form of the (1+1)-PAES algorithm employed in the local-search phase. In the development of M-PAES, early versions did not have the facility of repeatedly rejecting children of recombination. However, we found that this weakened the effectiveness of the elitism inherent in (1+1)-PAES and so the recombination phase was made more stringent in later versions.

### 6.5.1 Convergence properties of M-PAES

The M-PAES algorithm uses a global archive  $G$  to store nondominated solutions for presentation at the end of the run. The archive is updated for *every* potential solution generated, using the adaptive grid archiving strategy described in Chapter 4. Thus, the same convergence properties exist for M-PAES as for PAES, given the same restrictions.

### 6.5.2 Parameter control in M-PAES

The M-PAES algorithm introduces a number of additional parameters that must be controlled or selected. The following procedure for controlling these parameters is recommended.

1. Decide on the number of final solutions required, or alternatively, the number of divisions in each objective that should be represented by a solution. Using the number of objectives,  $K$ , of the problem, the archive  $G$  can then be set to the appropriate capacity, with reference to Table 4.1. Alternatively, for comparison with the performance of an MOEA, set the archive size  $G$  to correspond to the number of solutions that can be returned by the MOEA. Set the size of  $H$  less than or equal to  $G$ . The default is to set  $H = G$ .

2. Set *max\_evals* according to the maximum time available for the search, or equal to other algorithms, if a performance comparison is being carried out.
3. Once *max\_evals* has been set,  $P$ ,  $l_{opt}$ ,  $l_{fails}$ , and *recomb\_trials\_max* must be set. These parameters cannot be set independently. In general, increasing  $l_{opt}$  decreases the number of generations, and thereby the number of recombinations applied, while increasing the length of local-search phases. If  $l_{opt}$  is increased then  $P$  must be decreased to allow a reasonable number of generations to occur. Similarly, the setting of  $l_{fails}$  and *recomb\_trials\_max* controls the amount of time spent on searching within each local search and recombination phase respectively. Setting these high implies that the number of generations will be low for a given  $P$ . Thus  $P$  should be reduced to keep the number of generations to a reasonable level. It is unwise to be dogmatic about what constitutes a reasonable minimum number of generations but one might expect that fewer than ten generations is unlikely to yield good results.

In order to further ‘tune’ the parameters,  $P$ ,  $l_{opt}$ ,  $l_{fails}$ , and *recomb\_trials\_max*, we may iterate step 3 of the procedure given above, to obtain better performance. This process must be necessarily guided by one’s intuitions because the space of parameter choices is too large for a thorough investigation. However, we have found that one can quickly ascertain whether making changes to these parameters makes a significant difference to performance or not, on a given problem instance. In our investigations outlined next, we found that ‘hillclimbing’ in the space of these parameters always led to reasonable performance within several trial runs, and after this, significant performance gains were not forthcoming. In this respect, M-PAES does not seem to be over-sensitive to the choice of these parameters. However, at present, we certainly do not suggest that we know a technique to set these parameters *a priori*, given a problem instance, or how to relate landscape measurements to the setting of parameters.

## 6.6 Initial study: M-PAES on the multiobjective 0/1 knapsack problem

### 6.6.1 Experimental method

The M-PAES algorithm is tested on a suite of multiobjective 0/1 knapsack problems. The problems are taken from a recent paper [ZT99] by Zitzler and Thiele, in which the general ability of their strength Pareto evolutionary algorithm (SPEA) was demonstrated. In [ZT99], the performance of SPEA on these problems was compared with eight other evolutionary

algorithms (EAs). Four of the algorithms, each a well-known multiobjective EA, as well as two versions of SPEA, were run 30 times with different random seeds on each of the problems, for 500 generations using the same population sizes<sup>1</sup>. The nondominated sets generated from each of the runs were used to make a statistical comparison of the algorithms tested.

Zitzler and Thiele's study found that SPEA is superior to each of the other MOEAs on all of the knapsack problems. However, also included in the set of eight algorithms tested, are two single-objective EAs that use weighted-sum aggregation of the objectives. The relative performance of SPEA and these algorithms is not clear-cut. Hence, in the first part of our comparative study we select as benchmarks, the data sets from the SPEA runs and those of the more powerful of the two single-objective algorithms, SO-5. The other algorithms are not considered. As additional comparators, we generated our own data sets for the (1+1)-PAES algorithm, and an enhanced setup of SPEA, on the knapsack problems.

The second part of the study is a comparison with the memetic algorithm, RD-MOGLS. The same problem instances are used, and once again we also compare performance with (1+1)-PAES acting as a benchmark indicating a basic, good level of performance. The parameter settings for all of the six algorithms that we compared are described in Section 6.6.3.

### Multiobjective 0/1 knapsack problems

An excellent general text on knapsack problems is [MT90]. In it, the standard 0/1 knapsack problem is described, including the history of methods for solving it, its numerous applications, and its links with fundamental integer programming problems. The multiobjective version of the problem is not described, but it has become a favourite problem in multiobjective combinatorial optimization (see [GF98]). The following text reproduced from [Zit99] defines the problem:

Given a set of  $n$  items and a set of  $k$  knapsacks, with

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i, \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i, \\ c_i &= \text{capacity of knapsack } i, \end{aligned}$$

---

<sup>1</sup>In the case of SPEA, an internal and an external population exist. The sizes of these were chosen to provide a fair comparison with the other MOEAs in the study.

find a vector  $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$ , such that the capacity constraints

$$e_i(\mathbf{x}) = \sum_{j=1}^n w_{i,j} \cdot x_j \leq c_i \quad (1 \leq i \leq k) \quad (6.1)$$

are satisfied and for which  $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$  is maximum, where

$$f_i(\mathbf{x}) = \sum_{j=1}^n p_{i,j} \cdot x_j \quad (6.2)$$

and  $x_j = 1$  if and only if item  $j$  is selected.

Zitzler and Thiele generated nine instances of the problem altogether, of differing combinations of size (number of items), and number of objectives (knapsacks). At the time of writing, the problems are available from an Internet web-site<sup>2</sup>. In the following experiments, we employ the same chromosome encoding and constraint handling techniques as described in [ZT99], and no additional heuristics for use with the knapsack problems are employed. This allows for a direct comparison between our results and those published by Zitzler and Thiele.

### 6.6.2 RD-MOGLS

The RD-MOGLS algorithm, described in [Jas98], is given in outline in Figure 6.5. The algorithm can be used with either weighted linear or weighted Tchebycheff scalarizing functions.

### 6.6.3 Parameter choices

The problem of setting parameters in comparative studies of algorithm performance is a serious one. Our philosophy in this study is that each algorithm be run with settings, found through some experimentation, to provide near-best performance for that algorithm. It is not possible, or even desirable, to use exactly the same parameter settings for each algorithm, as they differ considerably.

The study comprises two different sets of data, one where M-PAES is compared with the results from [ZT99], and the other, where we implemented an algorithm, RD-MOGLS, ran it, and collected the results ourselves. In the former, parameter control is obviously not possible, although we did supplement the results from [ZT99] with two further algorithms: an SPEA

---

<sup>2</sup><http://www.tik.ee.ethz.ch/~zitzler>

```

- Initialization - Repeat  $S$  times:
  - Generate a random weight vector  $\vec{\lambda}$ .
  - From a randomly generated solution  $x$ , perform local search,
    using a scalarizing function  $\mathcal{S}(\vec{z}, \vec{\lambda})$ , to obtain  $x'$ .
  - Add  $x'$  to the current set of solutions,  $CS$ .
  - Update the potentially efficient set  $PE$  with  $x'$ .
- Main Loop - Repeat until some stopping criterion is met:
  - Generate a random weight vector  $\vec{\lambda}$ .
  - Select the  $N \leq S$  best solutions from  $CS$ , using the measure
     $\mathcal{S}(\vec{z}, \vec{\lambda})$ , to form a temporary population  $TP$ .
  - Repeat  $N$  times:
    * Do crossover on a pair of uniformly randomly selected
      parents from  $TP$ .
    * Do local search from the offspring  $x$  to form  $x'$  using
       $\mathcal{S}(\vec{z}, \vec{\lambda})$ , and update  $PE$  with  $x'$ .
    * If  $x'$  is better than worst member of  $TP$ , then add to
       $CS$  and  $TP$ , deleting worst member of  $TP$ .
    * Update set  $PE$  with  $x'$ .

```

Figure 6.5: RD-MOGLS.

implementation of our own; and (1+1)-PAES. The parameters of these algorithms, like those of RD-MOGLS, were set empirically to give good performance.

To reduce some of the burden of testing different parameter settings, we do keep core parameters/conditions constant across those algorithms that we have control over. Further details of the parameter choices made for each of the algorithms tested are given below. The data from the setup of SPEA used in [ZT99] is referred to as SPEA(ZT). Our own setup of SPEA is labelled SPEA(KC).

### SO-5

The SO-5 data comes from one of two single-objective EAs used in [ZT99]. Unlike the other algorithms considered, these single-objective EAs were run 100 times per test problem, each run optimizing toward a different randomly chosen linear combination of the objectives. The resultant internally efficient solutions among all those generated in the runs form the tradeoff

front achieved by the algorithm. The two algorithms both employed equal population sizes to their multiobjective rivals, and differed only in that one (SO-1) was run for 100 generations, and the other (SO-5) was run for 500 generations in every single of the 100 runs used to form the nondominated front. Thus, in the case of SO-5, one hundred times as many function evaluations as in the other MOEAs in [ZT99] were performed in order to generate the (single) set of internally efficient solutions. Zitzler and Thiele did not perform the whole process thirty times to give thirty different data sets, but instead just used the same set repeatedly in the statistical analysis carried out. We follow this approach, using Zitzler and Thiele's data sets. Hence, where statistical information is given in relation to the SO-5 algorithm, it should be noted that, in fact only one data set for this algorithm is being used, in contrast to all the other algorithms in this study for which 30 runs were performed.

As with the other algorithms in [ZT99], one-point crossover was used. The mutation probability and crossover rate were fixed at 0.01 and 0.8 respectively, as for SPEA(ZT).

### **SPEA(ZT)**

The setup of SPEA is described fully in [ZT99]. The study was designed to show that SPEA could clearly outperform the other MOEAs tested, even with very conservative choices of parameters. Thus the authors kept the external population quite small - 1/5 of the population size of the other MOEAs in the study.

It is important to note that the data sets for SPEA(ZT) record the off-line performance of the algorithm. That is, all of the internally efficient solutions returned in a run were recorded. With the exception of SO-5 the other algorithms in this study are judged using the on-line performance. That is, only solutions stored in the external population (or archive) at the end of the run are recorded.

### **SPEA(KC)**

In [ZT99], the authors chose to run the algorithms for a fixed number of generations and increase population size with the size and number of objectives of the knapsack problem being tackled. To make direct comparison possible, we choose to use the same number of function evaluations as Zitzler and Thiele did, but do not deem it necessary to employ equal population sizes. In fact, the total number of evaluations *max\_evals* used by each of the algorithms in this study is the same for a given knapsack problem. Table 6.2 lists the value of *max\_evals* for each knapsack problem.

SPEA requires two population sizes,  $N$  and  $N'$  to be set. Zitzler and Thiele selected to use

$N = 4/5$  and  $N' = 1/4$  of the size of population used by the other GAs in their study. Experiments performed by us show that the performance of SPEA on these knapsack problems is improved significantly when population sizes of  $N = 1/5$  and  $N' = 4/5$  are used, for the same total number of function evaluations. This use of a larger elite population significantly increases the selection pressure, and as such this strategy may not be ideal for other applications. We find that in this application, however, where a form of indirect representation is used to repair infeasible solutions, a high selection pressure is beneficial. This may be because good building-block discovery is not likely to occur quickly enough (given the indirect encoding) for a low selection pressure to be preferable.

Our experiments also indicate that changing the crossover type from one-point to uniform improves the performance of SPEA on the knapsack problems, especially with the use of high selection pressure as discussed above. Thus, SPEA(KC) employs uniform crossover. A fixed per-bit mutation rate  $p_m = 0.01$  is used, as in [ZT99]. No experiments in which  $p_m$  was varied were undertaken by us. Since no other parameters need to be set for SPEA, we believe that SPEA(KC) is close to the best setup of SPEA possible for the problems tackled.

### (1+1)-PAES

With (1+1)-PAES, very few parameters must be set. The archive size was set equal to the external population size  $N'$  of SPEA, so that the same number of solutions is returned by each algorithm. Similarly, the number of evaluations is set in accordance with the total number performed by SPEA.

The mutation rate  $p_m$  was set to  $4/L$  (where  $L$  is the number of bits in the chromosome) for all problems. This setting follows our principle of using the best setting for the particular algorithm.

The number of bisections of the objective space  $l$  used in the adaptive grid algorithm for maintaining diversity was set according to the number of objectives of the problem. The values  $l = 5$ ,  $l = 4$ ,  $l = 3$  were used for the 2, 3, and 4 objective multiple knapsack problems, respectively.

### M-PAES

The total number of evaluations, number of bisections of objective space,  $l$ , and per-bit mutation rate used in M-PAES are as for (1+1)-PAES. The population size  $N$ , was set equal to the internal population of SPEA(KC). The two archives were sized equally, to match the external population of our setup of SPEA. Thus, the same number of solutions are returned by

## 6.6 Initial study: M-PAES on the multiobjective 0/1 knapsack problem

| Knapsack problem | Parameter     |                        |                            |                            |
|------------------|---------------|------------------------|----------------------------|----------------------------|
|                  | <i>lfails</i> | <i>l<sub>opt</sub></i> | <i>cr<sub>trials</sub></i> | <i>max<sub>evals</sub></i> |
| 2-250            | 20            | 100                    | 25                         | 75000                      |
| 2-500            | 20            | 100                    | 25                         | 100000                     |
| 2-750            | 20            | 100                    | 25                         | 125000                     |
| 3-250            | 20            | 50                     | 100                        | 100000                     |
| 3-500            | 5             | 20                     | 125                        | 125000                     |
| 3-750            | 20            | 50                     | 150                        | 150000                     |
| 4-250            | 20            | 50                     | 125                        | 125000                     |
| 4-500            | 20            | 50                     | 150                        | 150000                     |
| 4-750            | 5             | 20                     | 150                        | 175000                     |

Table 6.2: Parameter settings used in the M-PAES algorithm for the various multiple objective knapsack problems. The same total number of function evaluations *max<sub>evals</sub>*, shown for each problem, was used in M-PAES, SPEA(KC) and (1+1)-PAES.

SPEA(KC), M-PAES and (1+1)-PAES. The recombination operator was uniform crossover, in line with SPEA(KC), and the mutation rate was set to  $4/L$ , as with PAES.

In M-PAES, three more parameters must be set. These are the number of crossover trials, the maximum number of local moves *l<sub>opt</sub>*, and the maximum number of consecutive failing local moves *lfails*. Choices that give good general performance were found to be  $l_{opt} = 50$ ,  $l_{fails} = 20$ , and  $cr_{trials} = 25$ . However, it was found that increasing the number of crossover trials for the 3 and 4-objective problems increased performance further. A list of the best parameter selections found is given in Table 6.2. The results presented in Figure 6.6 and Table 6.4 are for these settings.

## RD-MOGLS

The parameters chosen for RD-MOGLS are given in Table 6.3. The initial and temporary population sizes were derived empirically from a few test runs of RD-MOGLS. The size of *PE* was set equal to the equivalent nondominated *archive*, used in M-PAES and (1+1)-PAES. There must also be a stopping criterion for each of the local-search phases. This detail is omitted in [Jas98], so we choose to end each local-search phase when *lfails* consecutive moves do not improve the solution. This is similar to the method used in M-PAES. Once again, *lfails* was set empirically. As with M-PAES, the mutation rate  $p_m = 4/L$ , and the recombination operator was uniform crossover.

To obtain solutions distributed across the whole Pareto front, RD-MOGLS selects a weight

## 6.6 Initial study: M-PAES on the multiobjective 0/1 knapsack problem

|                                      |     |
|--------------------------------------|-----|
| initial population size $S$ :        | 100 |
| temporary population size $N$ :      | 20  |
| efficient solution set size $ PE $ : | 100 |
| local-search fails $l\_fails$ :      | 5   |
| weight vector parameter $n$ :        | 100 |

Table 6.3: Parameter settings for RD-MOGLS.

| Instance | Algorithm |              |              |              |
|----------|-----------|--------------|--------------|--------------|
|          | 1+1-PAES  | SPEA(ZT)     | SPEA(KC)     | S0-5         |
| 2-250    | [87.2, 0] | [61.6, 24.5] | [28.1, 21.8] | [64.3, 27.5] |
| 2-500    | [100, 0]  | [100, 0]     | [80.9, 7.2]  | [92.8, 3.6]  |
| 2-750    | [100, 0]  | [100, 0]     | [95.5, 0]    | [100, 0]     |
| 3-250    | [100, 0]  | [69.1, 18.7] | [53.6, 26.6] | [44.4, 51.8] |
| 3-500    | [100, 0]  | [93.7, 0]    | [67.9, 20.8] | [74.5, 21.7] |
| 3-750    | [100, 0]  | [100, 0]     | [84.2, 3.9]  | [94.2, 3.6]  |
| 4-250    | [100, 0]  | [46.1, 31.6] | [32.9, 43.4] | [10.0, 85.9] |
| 4-500    | [100, 0]  | [92.5, 3.7]  | [63.3, 21.7] | [35.5, 56.6] |
| 4-750    | [100, 0]  | [100, 0]     | [82.9, 7.4]  | [71.7, 25.4] |

Table 6.4: The results of testing M-PAES against the algorithms shown, using our statistical techniques. The knapsack problems have 2, 3 or 4 objectives and 250, 500 or 750 items, as indicated.

vector at random at each step. We follow the same procedure used by Jaszkiewicz for doing this, which was put forward in [BPH98], and gives a maximally dispersed set of weight vectors. A random weight vector is generated in which each individual weight takes on one of the following values  $\{\frac{m}{n}, m = 0, \dots, n\}$ , where the fixed parameter  $n$  is any positive integer, and the value of  $m$  is uniformly randomly distributed in  $0, \dots, n$ . The number of unique weight vectors this results in is equal to:

$$\binom{k+n-1}{n} = {}^{k+n-1}C_n = \frac{(n+k-1)!}{n!(k-1)!}. \quad (6.3)$$

Following Jaszkiewicz, we use a value of  $n = 100$ , in all experiments, giving respectively 101, 5151, and 176,851 weight vectors for the 2, 3, and 4 objective problems.

## 6.6 Initial study: M-PAES on the multiobjective 0/1 knapsack problem

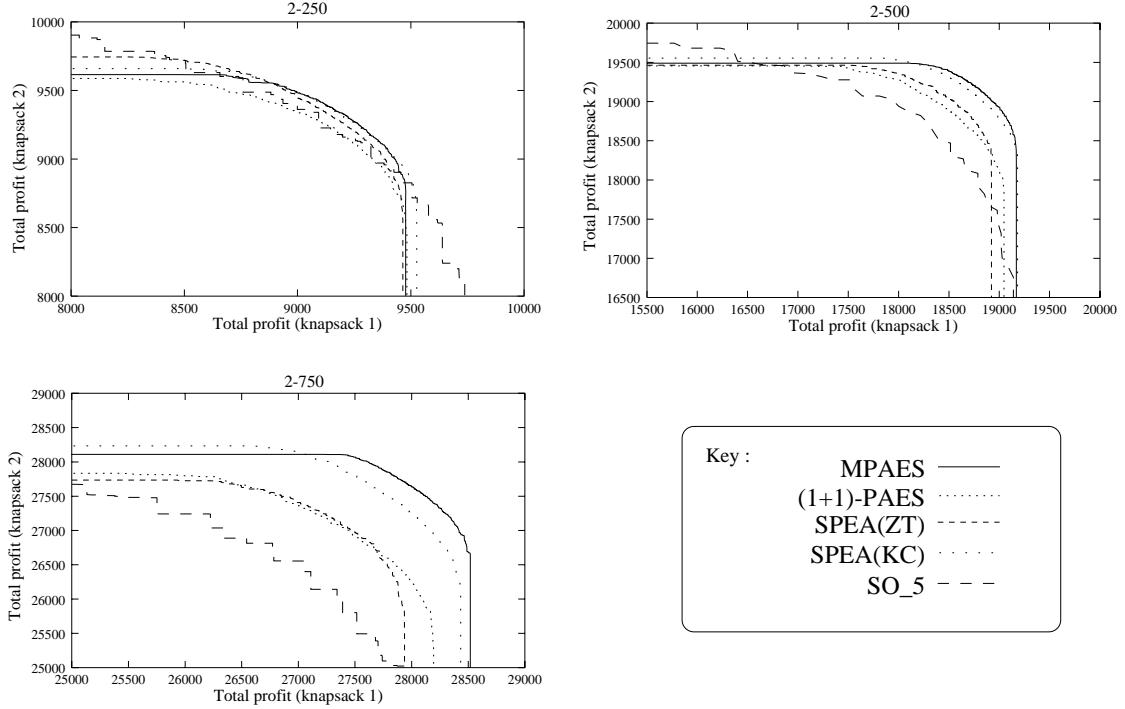


Figure 6.6: Median surfaces calculated from 30 runs of each algorithm, on the two-objective knapsack problems.

### 6.6.4 Results

#### Analysis: part 1

The results of the first part of the study, where M-PAES is compared with the algorithms in [ZT99] are shown in Figure 6.6 and Table 6.4. In Figure 6.6 the median surfaces generated by each algorithm are plotted for the two-objective problems. A number of observations can be made from these plots. First, our setup of (1+1)-PAES gives very similar levels of performance to the setup of SPEA used by Zitzler and Thiele on the three problems. This observation is in keeping with previous research where (1+1)-PAES was compared with SPEA (see Section 5.3). Second, in all cases SPEA(KC) outperforms SPEA(ZT). Third, M-PAES is very competitive with SPEA(KC) and clearly outperforms both (1+1)-PAES and SPEA(ZT). In the largest of the three problems, M-PAES generates a median surface that is clearly superior to the median surface of any of the other algorithms. On the smaller problems, M-PAES fails to generate solutions as far towards the extremes of the objective space as either SO-5 or SPEA(KC), but has generated a median surface that dominates these algorithms in the region where the two objectives trade off most rapidly with each other.

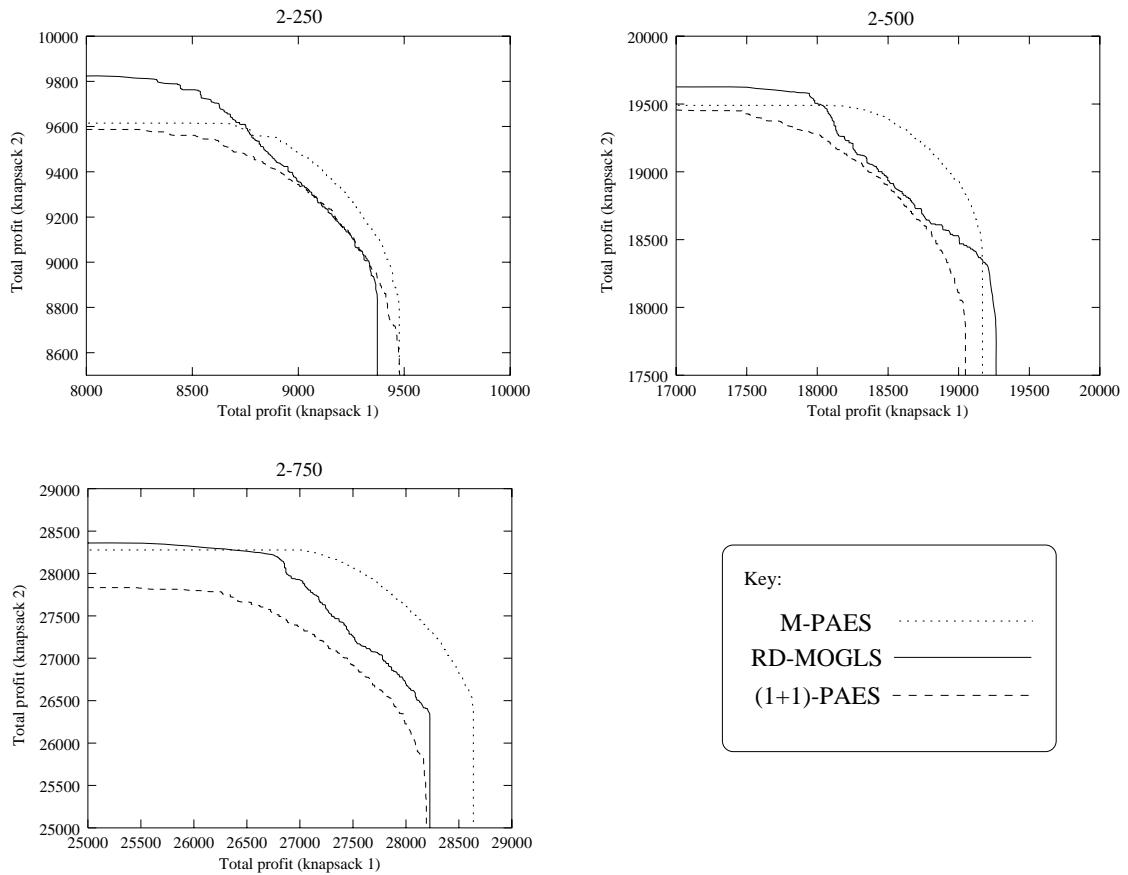


Figure 6.7: Median surfaces calculated from 30 runs of the three algorithms, M-PAES, RD-MOGLS, and (1+1)-PAES, on the 2-objective knapsack problems.

The statistical results for all the problems are summarised in Table 6.4. Comparisons between M-PAES and each of the algorithms are presented only. Thus, the statistic [100, 0] in the upper left entry in the table means that M-PAES gives a better distribution of surfaces over 100% of the combined nondominated front than (1+1)-PAES on the 250 item, 2 knapsack problem. Again, several observations from these results can be made. First, the first three rows of the table verify that M-PAES performs well on the two-objective problems, as suggested by the plots in Figure 6.6. Its relative performance increases as the number of items increases. This is true, not only on the 2-objective problem but on all the problems presented. However, as the number of objectives increases, the performance of SPEA(KC) and SO-5 increase relative to M-PAES, so that M-PAES is outperformed by SO-5 on the two smaller four-objective knapsack problems. SPEA(KC) only outperforms M-PAES on one problem, the smallest of the 4-objective knapsack problems, and only by a small margin, according to our statistical analysis.

### Analysis: part 2

Results collected from the second part of the study are shown in Figure 6.7 and Table 6.5. In all three of the plots in Figure 6.7, the median surface generated by RD-MOGLS extends beyond the surface generated by M-PAES in at least one of the objectives. However, the M-PAES algorithm generates a median surface that dominates the surface of RD-MOGLS over a larger portion of the tradeoff front. With increasing problem size (items), the M-PAES algorithm's performance improves relative to RD-MOGLS. Both M-PAES and the baseline algorithm (1+1)-PAES give smoother, more convex median surfaces than RD-MOGLS. Indeed, the plots of the nondominated fronts found by RD-MOGLS indicate that it has some trouble approaching the true Pareto front in the middle region where the 'compromise' solutions lie. Two explanations for this behaviour are proposed. First, RD-MOGLS is more successful than the Pareto methods at finding extreme vectors and consequently its search effort may not be so concentrated on the compromise region. Secondly, the RD-MOGLS algorithm uses a form of restricted mating whereby only the best solutions in a particular search direction ( $\lambda$ -vector) are recombined to form the offspring for local improvement in the same direction. Thus, there is less scope for recombining parts of good extremal solutions to form good compromise solutions. In this application it seems that this form of restricted mating is not beneficial. Indeed, it was trialled in M-PAES, and was found to degrade performance slightly.

The statistical results for all the problems are summarised in Table 6.4. Comparison is made between M-PAES and RD-MOGLS only. Thus, the statistic [65.4, 28.2] in the upper left entry

## 6.6 Initial study: M-PAES on the multiobjective 0/1 knapsack problem

| Knapsacks | Items        |           |           |
|-----------|--------------|-----------|-----------|
|           | 250          | 500       | 750       |
| 2         | [65.4, 28.2] | [49.4, 0] | [61.5, 0] |
| 3         | [72.7, 25.1] | [89.8, 0] | [100, 0]  |
| 4         | [100, 0]     | [100, 0]  | [100, 0]  |

Table 6.5: The results of testing M-PAES against RD-MOGLS using our statistical technique, AS2 (see Section 3.12) and with 30 independent runs of each algorithm. The knapsack problems have 2, 3 or 4 objectives and 250, 500 or 750 items, as indicated.

in the table means that M-PAES gives a better distribution of surfaces than RD-MOGLS over 65.4% of the tradeoff front, and vice-versa, RD-MOGLS gives a better distribution than M-PAES on 28.2%, on the 250 item, 2 knapsack problem. Again, several observations from these results can be made: The first row of the table verifies that M-PAES performs well on the two-objective problems, as suggested by the plots in Figure 6.6. Its relative performance increases as the number of items increases. This is true, not only on the 2-objective problem but also the 3-objective problems. Clearly, as the number of objectives increases, the relative performance of M-PAES increases compared to RD-MOGLS, with the results showing that the distribution of solutions found from the runs of M-PAES show statistically significant superiority to those of RD-MOGLS on 100% of the sampling lines used to probe the Pareto fronts.

### 6.6.5 Summary

The efficacy of M-PAES was verified on a set of nine multiobjective 0/1 knapsack problems. Two separate studies were conducted. In the first, the results from runs of M-PAES were compared with the results gathered in a published study by Zitzler and Thiele. The results sets from the study were supplemented by also running our own implementation of the strength Pareto evolutionary algorithm (SPEA), as well as (1+1)-PAES. The results indicate that M-PAES performs better than the local-search algorithm, (1+1)-PAES (on which it is based), on all of the problem instances. Compared with SPEA, the performance of M-PAES is similar, for a setup of each algorithm empirically derived to give near-best performance. Indeed, M-PAES appears to be superior on some problem instances, although comparison between these very different algorithms is difficult.

In the second study, M-PAES was compared with another multiobjective MA, the random directions multiple objective genetic local-search (RD-MOGLS) algorithm of Jaszkiewicz, using (1+1)-PAES as a baseline, once more. Both algorithms work well, generating results

that are better than (1+1)-PAES, produced. However, on the experiments carried out, M-PAES, was found to be superior overall. The RD-MOGLS algorithm seemed capable of generating solutions over a wider range in each of the objectives on the smaller 2-objective problems, but as the size of the problem and the number of objectives were increased, M-PAES exhibited superior performance according to our statistical measures.

On the negative side, the setting of parameters in the M-PAES algorithm is difficult at present. It appears to be more sensitive to them than SPEA is. Some investigation into mechanisms for controlling the parameters would be necessary for M-PAES to become a useful general purpose optimizer.

Nonetheless, the findings indicate that further investigation into the use of memetic algorithms in multiobjective combinatorial optimization is warranted. The M-PAES algorithm only represents one instance of the new memetic algorithm framework put forward, and it is anticipated that when good local-search heuristics are available for a particular problem, algorithms based on the framework will have an advantage over standard evolutionary approaches.

## **6.7 Assessment of M-PAES, (1+1)-PAES, and SPEA on the ADDMP**

The adaptive distributed database management problem (ADDMP) was described in Section 5.2.2, where a single instance was used as part of a test function suite. In this section, we use a number of further ADDMP instances as the basis for an empirical study.

### **6.7.1 ADDMP instances**

Instances of the ADDMP can occur in great variety. The numbers of clients and servers can range typically between 2 and 20, and the number of clients between 10 and several thousand. Access patterns can vary equally dramatically. For example, access to share price and similar financial databases may be very frequent with constantly changing global activity, and hence re-optimization of client/server access configurations may need to occur every few minutes. In other scenarios, re-optimization may only need to occur every few hours involving a small number of clients.

An ADDMP instance is defined by a number of arrays and parameters, defining server speeds, typical latencies in the underlying communications matrix, client access rates, and other

factors. The fitness function uses queuing theory to estimate the actual response times that will ensue given a particular client/server connection configuration. The reader is referred to [OC98] for a fuller description of the model; here, we need to say a little more about the client access rates.

A single ADDMP scenario is defined by a datafile which provides the information alluded to above, a key aspect of which is an array of access rates for each client. In the case of a globally used service, we can expect access rates to vary in a particular pattern over the course of 24 hours. Therefore, raw data for the ADDMP for the same database service will change with time in the way indicated in Table 6.6. Table 6.6 indicates the possible situation for two snapshots in time in a simple 5-node but globally distributed scenario. Access rates are in ‘queries per second’, and each client is typically expected to represent a LAN or WAN comprising several users of the database; hence, the rate for ‘client 1’ is actually the combined rate from clients accessing the database via users ‘centred’ at client 1.

| Clients:                   | 1   | 2   | 3   | 4   | 5   |
|----------------------------|-----|-----|-----|-----|-----|
| Access rates: scenario 1.1 | 1.5 | 0.4 | 0.1 | 0.4 | 0.6 |
| Access rates: scenario 1.2 | 0.1 | 0.4 | 0.6 | 1.5 | 0.4 |

Table 6.6: Access rates for two instances of the ‘same’ ADDMP, several hours apart.

Scenario 1.1 in Table 6.6 might refer to a situation in which it is about 3:00pm in the location of client 1, very late at night or early morning in the area of client 3, and around 9:00am in the area of client 5. Several hours later, in scenario 1.2, the access pattern changes to reflect the same pattern, but ‘phase-shifted’ as it is now 3:00pm (say) for client 4, and perhaps midnight for client 1.

In this study we tackle ADDMP scenarios involving 10, 20, and 40 clients, and in each case we consider 5 separate problems which reflect changes in access patterns over time in the way illustrated by Table 6.6. Thereby, we are comparing the quality of M-PAES and SPEA on the ADDMP over a wide but representative range of potential instances. We are interested particularly in the kind of ADDMP problem in which solutions must be calculated in a short time-span. Hence, in increasing order of the problem sizes, the maximum allowed number of evaluations is 500, 2000 and 5000, respectively.

We consider both 2-objective and 3-objective versions of each problem. In the 2 objective version, the objectives are the worst delay figure and the median delay figure. In the 3-objective version, the objectives are the 90% quartile delay figure (i.e.: 90% of clients will have a better delay figure than this), the 80% delay figure, and the median figure.

Finally, we need to note a point concerning the limitations of the fitness function model. It has been found to be robust over a wide range of ‘normal’ scenarios, however it can be expected to break down (and hence give inaccurate estimates of response times) for configurations which highly overload certain servers. We deal with this by building in a conservative cutoff of 1000 ms; that is, when the model estimates more than 1000ms for the worst client delay, we return the result ‘1000’ for all the objectives rather, forcing the point to be rejected. In this way, we avoid troubling the Pareto frontier with inaccurate phenotypes.

### 6.7.2 Parameter control and performance assessment

The parameter choices made for each algorithm were derived in the following way. First, it was found that uniform crossover was more effective than 1-point crossover on the ADDMP for both SPEA and M-PAES, from a small number of preliminary experiments, so this was used throughout. Some experiments with the SPEA algorithm found that it generally worked best with a crossover rate of 0.8, so this value was set for all further experiments. A mutation rate of  $1/L$  was chosen for all algorithms, and each uses flip mutation. That is, each gene is changed with probability  $1/L$ , to a uniformly randomly selected allele value in the allowed range, other than its current one. This was found to work well with all algorithms in initial experiments and was not further changed. The total population size was set to 100 for SPEA. This was split between 5/95 to 20/80 internal/external population. The actual choice for each problem was made after several preliminary runs. Similarly for M-PAES, the population was set between 5 and 20 while the archives  $G$  and  $H$  were set between 95 and 80, depending on problem. PAES was set up with an archive of 99 solutions. The other parameters of M-PAES —  $l\_fails$ ,  $l\_opt$ , and  $cr\_trials$  — were set using *ad hoc* experimental trials. The ranges of values used for all parameters are summarized in Fig. 6.8.

The performance assessment metrics employed are described in Section 3.12. Specifically, the metrics AS3, S1, S2 are used.

### 6.7.3 Results

In the following, all results are based on thirty independent runs of each algorithm on each problem instance. The first set of results (Fig. 6.9) was obtained using attainment surface sampling and our  $n$ -algorithm comparison metric. These results, taken on their own, seem to indicate that (1+1)-PAES is consistently difficult to beat, whereas SPEA and M-PAES are closely matched but not as consistently good as PAES. Results obtained from our other metrics, a key subset of which are presented below, are not always in agreement, however.

## 6.7 Assessment of M-PAES, (1+1)-PAES, and SPEA on the ADDMP

| parameter           | M-PAES         | SPEA           | (1+1)-PAES  |
|---------------------|----------------|----------------|-------------|
| $p_c$               | N/A            | <b>0.8</b>     | N/A         |
| crossover           | <b>uniform</b> | <b>uniform</b> | N/A         |
| $p_m$               | <b>1/L</b>     | <b>1/L</b>     | <b>1/L</b>  |
| mutation            | <b>flip</b>    | <b>flip</b>    | <b>flip</b> |
| internal population | 5–20           | 5–20           | <b>1</b>    |
| external population | 80–95          | 80–95          | <b>99</b>   |
| $l\_fails$          | 1–20           | N/A            | N/A         |
| $l\_opt$            | 2–100          | N/A            | N/A         |
| $cr\_trials$        | <b>5</b>       | N/A            | N/A         |
| $l$                 | 5 / 3          | N/A            | 5 / 3       |

Figure 6.8: Parameter settings for the three algorithms. Bold face indicates a fixed value in all experiments. The ranges of values used for the free parameters is shown. These were investigated on an *ad hoc* basis to provide ‘best’ performance. The two values shown for  $l$ , the number of bisection levels in the adaptive grid algorithm, refer to the values for the two and three objective problems, respectively.

The results of calculating  $S_\tau$  for three of the problems are shown in Fig. 6.10. First, notice that the values for the three algorithms are very close and that a large number of figures are significant. Nonetheless, the results here still provide extra information about the distribution of solutions found by the algorithms over multiple runs. On the first two of the 3-objective cases, two algorithms generate *exactly* the same total dominated region. In all probability this must indicate that the total set of non-dominated solutions found in each case is exactly the same. Interestingly, the total dominated region measure favours SPEA over M-PAES, although (1+1)-PAES is superior overall. Using our *unbeaten* and *beats all* statistics, M-PAES is ranked ahead of SPEA, with (1+1)-PAES in first place. This disagreement must indicate that SPEA tends to generate different solutions on different runs more often than M-PAES. Still, the total dominated region preserves the position of (1+1)-PAES as the most consistent algorithm, it winning on three of the six measurements, and being beaten only once.

In the next set of results presented (Fig. 6.11) the median attainment surface is first calculated for each algorithm. The size of the region dominated is then measured. Once again, using this measure alone could lead to different conclusions than if using it in conjunction with other measures. For example, on the two-objective version of problem 10-3, the rank order reported by this measure  $S_\mu$ , the  $S_\tau$  measure, and our *unbeaten* statistic are all different. Clearly the algorithms perform at very similar levels on this problem, but which is best? On the three objective version of the same problem, M-PAES and (1+1)-PAES are very similar with regard to  $S_\mu$ , whereas M-PAES is a poor third when considering the whole distribution

of attainment surfaces, using the Mann-Whitney  $U$  test.

Finally, the coverage differences,  $S_{A \setminus B}$  and  $S_{B \setminus A}$  were calculated for pairs of the algorithms, for each of the thirty runs. The median values of these differences are presented in Fig. 6.12. These results exhibit a high degree of agreement with the equivalent results in Fig. 6.9. According to the former, the attainment surface generated by (1+1)-PAES completely dominates the attainment surface found by the other two algorithms on at least 50% of the runs, on the three-objective versions of the two problems.

#### 6.7.4 Concluding remarks

In this study we have compared the performance of three multiobjective algorithms with respect to a suite of real-world problems related to the management of distributed databases. Several test metrics were employed to measure and compare algorithm performance over collections of solution sets found from several (30) runs. Three extensions to a metric based on the size of the dominated space were used for the first time. The results show that with such closely-matched algorithms, it is a very difficult matter to select the one that performs best. In fact, from the results presented, (1+1)-PAES seems to be the best performer all-round. However, results given by the different metrics indicated that the rank order of algorithms was certainly not independent of the test metric, on all problems. This shows the benefits of using a number of different metrics, when comparing algorithms: where a rank order is inconsistent across different metrics, the extra information provided can help to understand in what way one algorithm's approximations may differ from another's.

The performance of M-PAES on this problem is not as good as the much simpler (1+1)-PAES algorithm. Even with considerable tuning of the parameters it was not able to generate better results than PAES, in the same number of function evaluations. Similarly, SPEA did not outperform PAES, in general, although more detailed analysis of the full results reveals that SPEA was most consistent at providing an even distribution of solutions along the front, whereas (1+1)-PAES more often found very strong compromise solutions. These findings suggest that the ADDMP, with the current encoding, is not optimized effectively using recombination, in agreement with the findings of Oates [Oat00]. Overall, the dominance of the (1+1)-PAES algorithm, underlines the importance of baselining results against simpler algorithms. On the ADDMP, it appears that the (1+1)-PAES algorithm, which needs no parameter tuning, performs adequately compared to other more complicated algorithms, including M-PAES.

| ADDMP instance | statistic        | 2-objective |      |          | 3-objective |      |          |
|----------------|------------------|-------------|------|----------|-------------|------|----------|
|                |                  | M-PAES      | SPEA | 1+1-PAES | M-PAES      | SPEA | 1+1-PAES |
| 10-1           | <i>unbeaten</i>  | 93.8        | 83.1 | 99.2     | 98.0        | 81.3 | 100      |
|                | <i>beats all</i> | 0.8         | 0    | 5.0      | 0           | 0    | 2.0      |
| 10-2           | <i>unbeaten</i>  | 100         | 99.8 | 94.9     | 68.5        | 98.2 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0        | 0           | 0    | 1.8      |
| 10-3           | <i>unbeaten</i>  | 100         | 98.7 | 95.2     | 77.3        | 78.4 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0        | 0           | 0    | 8.3      |
| 10-4           | <i>unbeaten</i>  | 100         | 98.9 | 100      | 64.3        | 51.8 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0        | 0           | 0    | 34.1     |
| 10-5           | <i>unbeaten</i>  | 99.8        | 49.7 | 100      | 35.3        | 54.9 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0.2      | 0           | 0    | 16.3     |
| 20-1           | <i>unbeaten</i>  | 100         | 48.7 | 100      | 95.8        | 24.1 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0        | 0           | 0    | 3.0      |
| 20-2           | <i>unbeaten</i>  | 65.0        | 67.2 | 100      | 63.7        | 63.7 | 99.8     |
|                | <i>beats all</i> | 0           | 0    | 32.8     | 0           | 0    | 36.3     |
| 20-3           | <i>unbeaten</i>  | 100         | 0    | 100      | 98.9        | 67.0 | 100      |
|                | <i>beats all</i> | 0           | 0    | 0        | 0           | 0    | 1.1      |
| 20-4           | <i>unbeaten</i>  | 52.1        | 0.5  | 100      | 95.2        | 41.0 | 100      |
|                | <i>beats all</i> | 0           | 0    | 47.9     | 0           | 0    | 4.8      |
| 20-5           | <i>unbeaten</i>  | 49.9        | 50.0 | 100      | 23.8        | 24.7 | 100      |
|                | <i>beats all</i> | 0           | 0    | 50.0     | 0           | 0    | 74.1     |
| 40-1           | <i>unbeaten</i>  | 92.0        | 15.6 | 99.9     | 77.4        | 31.0 | 70.8     |
|                | <i>beats all</i> | 0.1         | 0    | 8.0      | 9.9         | 0    | 22.6     |
| 40-2           | <i>unbeaten</i>  | 69.3        | 7.9  | 93.4     | 76.4        | 26.2 | 87.6     |
|                | <i>beats all</i> | 4.6         | 0    | 30.7     | 0.5         | 0    | 23.6     |
| 40-3           | <i>unbeaten</i>  | 68.7        | 10.3 | 93.8     | 71.7        | 15.0 | 77.4     |
|                | <i>beats all</i> | 4.2         | 0    | 31.3     | 10.9        | 0    | 28.3     |
| 40-4           | <i>unbeaten</i>  | 100         | 11.2 | 99.5     | 78.2        | 0.1  | 93.1     |
|                | <i>beats all</i> | 0           | 0    | 0        | 6.9         | 0    | 21.8     |
| 40-5           | <i>unbeaten</i>  | 68.2        | 12.9 | 94.7     | 69.1        | 57.7 | 73.0     |
|                | <i>beats all</i> | 4.9         | 0    | 31.8     | 0           | 0    | 30.9     |

Figure 6.9: The AS3 metric results. They show the *unbeaten* and *beats all* statistics for the combined space inhabited by the solutions found. Two forms of the problem were investigated: The 2-objective case, where the median response time and the worst response time are minimized. And the 3-objective case, where the median response time, the response time bettered by 80% of requests, and the response time bettered by 90% of requests are minimized. The different problems are labelled by the number of nodes and the number of the scenario. E.g. The third scenario of the twenty node problem is labelled 20-3.

| ADDMP instance | 2-objective |         |          | 3-objective |           |           |
|----------------|-------------|---------|----------|-------------|-----------|-----------|
|                | M-PAES      | SPEA    | 1+1-PAES | M-PAES      | SPEA      | 1+1-PAES  |
| 10-3           | 0.33803     | 0.33776 | 0.33810  | 0.2703528   | 0.2703444 | 0.2703528 |
| 20-3           | 0.33580     | 0.34616 | 0.34070  | 0.1322440   | 0.1322490 | 0.1322490 |
| 40-3           | 0.27758     | 0.30131 | 0.30210  | 0.0498326   | 0.0498368 | 0.0498400 |

Figure 6.10: The values of the total dominated region,  $S_\tau$ , for three problem instances.

| ADDMP instance | 2-objective |          |          | 3-objective |          |          |
|----------------|-------------|----------|----------|-------------|----------|----------|
|                | M-PAES      | SPEA     | 1+1-PAES | M-PAES      | SPEA     | 1+1-PAES |
| 10-3           | 0.1551      | 0.1590   | 0.1482   | 0.09570     | 0.0867   | 0.0960   |
| 40-1           | 0.342969    | 0.345659 | 0.345649 | 0.024767    | 0.026677 | 0.024202 |

Figure 6.11: The values of the Median Attainment Region,  $S_\mu$ , for two problem instances.

| ADDMP instance | obj-<br>-jectives | A= M-PAES           |        | B= 1+1-PAES         |        | A= SPEA             |        | B= 1+1-PAES         |        | A= SPEA             |     | B= M-PAES           |     |        |        |
|----------------|-------------------|---------------------|--------|---------------------|--------|---------------------|--------|---------------------|--------|---------------------|-----|---------------------|-----|--------|--------|
|                |                   | $S_{A \setminus B}$ |        | $S_{B \setminus A}$ |        | $S_{A \setminus B}$ |        | $S_{B \setminus A}$ |        | $S_{A \setminus B}$ |     | $S_{B \setminus A}$ |     |        |        |
|                |                   | sc10-1              | 2      | 0.0327              | 0.0124 | 0.0384              | 0.0134 | 0.0353              | 0.0216 | sc10-1              | 3   | 0.0678              | 0.0 | 0.0161 | 0.0    |
| sc20-4         | 2                 | 0.0187              | 0.0161 | 0.0286              | 0.0153 | 0.0248              | 0.0118 | sc20-4              | 3      | 0.0026              | 0.0 | 0.0239              | 0.0 | 0.0084 | 0.0009 |

Figure 6.12: The median values of the coverage differences of alternate pairs of algorithms on two problem instances.

## 6.8 Summary

In this chapter we have presented a new memetic algorithm framework for multiobjective MAs. The framework specifies the use of Pareto selection in both local-search and reproduction phases of the MA. The potential advantages and disadvantages of this type of approach, compared to using scalarizing selection were discussed, and we noted that while the computational overhead *per* evaluation of a Pareto MA may be higher, it may be more a more efficient searcher, in terms of progress per evaluation.

Using procedures developed in Chapter 4, we used the MA framework to design an MA called M-PAES. This algorithm combines an elitist MOEA with a local improvement procedure based on (1+1)-PAES. The objective vectors discovered during a run of M-PAES are stored in a global archive which is updated using the adaptive grid archiving strategy developed for (1+1)-PAES.

We have only begun to test the performance of M-PAES and it is clear that we need to do a much more thorough investigation of how the setting of parameters might affect this. Nonetheless, the performance of M-PAES on the knapsack problems indicates that this approach may offer potential performance advantages over other methods, even where tailored heuristics are not utilized. In the next chapter we use M-PAES to tackle a problem where the use of specialized operators may offer a more significant advantage to MAs.

# Chapter 7

## Multi-criterion MST (mc-MST) Problems

### 7.1 Introduction

#### 7.1.1 Motivation

In the previous chapters a number of multiobjective optimization algorithms have been proposed and tested. These are general-purpose methods, which it is hoped will achieve reasonable search performance across a broad range of problems. However, in order for even better performance to be obtained in any real application, it would be necessary to tailor these algorithms specifically to the problem at hand. This tailoring may take different forms, for example, specialist initialization procedures that ‘seed’ the population with good solutions; tailored encodings of the solution space that aid search and discourage constraint violation; and specialist operators that facilitate both exploration and exploitation, can all be used. Indeed, even a whole heuristic technique may be embedded into these algorithms.

In this chapter, we use the evolutionary and memetic algorithms developed earlier in the thesis to tackle a combinatorial problem which, in one form or another, has been of major significance to the fields of operational research and combinatorial optimization: the minimum spanning tree (MST) problem. In order to improve the performance of our algorithms on this problem, we develop and use specialist operators, encodings, constraint-handling techniques, and initialization procedures. After some testing and comparison of our methods, we finally present a number of benchmark results for a diverse suite of instances of the multi-criterion MST problem. To provide some context for these results, we begin, in this introduction,

with a short review of the MST problem. We review the various EA and other approaches to *NP*-hard versions of the problem, and focusing particularly on the degree-constrained and multi-criterion MST problems, that we tackle in later sections.

### 7.1.2 The minimum spanning tree problem

The minimum spanning tree (MST) problem requires us to find a minimum cost set of edges to connect up a set of vertices within a weighted graph. The problem is the simplest and one of the most central models in the field of combinatorial optimization, and it has inspired the development of many other problem domains in discrete optimization. Its applications are varied, and are both practical and theoretical [Nes97]. Most importantly, the MST problem arises in many systems, such as highways, computer networks, telephone lines, and television cables, where we need to design a simplest network (spanning tree) that will connect — at minimal total cost — geographically dispersed system components so that they can communicate with each other [Pri57].

A lesser known application of the MST is clustering: if an MST is found between a set of vectors (points) representing data in some metric space, and the  $m$  longest edges are then removed, the data will be grouped into  $m$  clusters. This technique has it uses in many fields of study wherever classification, taxonomy, and visualization of data, are important. For example, Jones *et al.* used the minimal spanning tree to statistically analyze fungal spore spatial patterns [JLM96], while Barrow *et al.* have used the problem in the analysis of astronomical data [BBS85].

Other practical applications of MST problems include minimizing message-passing in distributed systems and making bitmaps for compressing large files, according to [ADG00]. Theoretically, the MST is important because it may be used to approximate solutions to other combinatorial problems such as the travelling salesman problem (TSP) [Kru56] and steiner tree. Furthermore, the MST problem is partly responsible for the development of the greedy algorithm and complexity measures [Nes97], and it has led to the development of new data structures and many algorithms [GGST86].

Fortunately, for such an important problem, the MST, without additional constraints, is solvable in polynomial time. The most popular MST algorithms are those of Prim [Pri57] and Kruskal [Kru56], but recently, it has been discovered that in the pre-computer age these and other procedures had already been invented [Nes97]. In particular, Jarník [Jar30] had already formulated Prim's algorithm by 1930, while Boruvka [Bor26] had devised an efficient approach by 1926. Reappraisal of the latter has led to a resurgence of interest in the methods

for solving the MST, with Boruvka's algorithm becoming the basis of several near-linear time algorithms [KKT95, Kin95].

### 7.1.3 *NP*-hard MST problems

Although the basic MST problem is in  $P$ , the addition of one or more constraints often transforms it into an  $NP$ -complete problem. In fact, thirteen different  $NP$ -complete problems based on spanning trees are listed in Garey and Johnson [GJ79], of which six relate to bounding (optimizing) the tree weight. Naturally, the existence of so many closely-related problems of importance to network design has led to the proposal of many heuristic and metaheuristic approaches.

Up to 1999, finding network trees had been the subject of 23 papers, according to Sinclair's review of the use of evolutionary algorithms for telecommunications problems [Sin99]. In particular, Gen and colleagues have used GAs to tackle a variety of network design-related problems based on  $NP$ -hard minimum spanning trees, including the quadratic MST, the degree-constrained MST, the leaf-constrained MST, and the multi-criterion MST [Gen00]. Gen has also used a GA approach for 'solving' the multi-objective fixed charge transportation problem and a local area network (LAN) design problem, based on finding constrained MSTs [Gen00]. In addition, Elbaum and Sidi also used spanning trees in a LAN design problem involving the simultaneous optimization of criteria relating to traffic locality and balancing, average and maximum delay and network cost [ES96]. And Palmer and Kershenbaum [PK97] considered the importance of encodings to GA approaches for optimizing trees, and proposed the node and leaf-biased encoding which was later reviewed by Gaube and Rothlauf [GR01]. More recently, Li and Bouchebaba [LB99] and Li [Li00] have proposed genetic algorithm encodings for the optimal communication spanning tree problem [GJ79].

### 7.1.4 Approaches to the $d$ -MST problem

In the following, we review some of the many different approaches that have been taken to the degree-constrained MST ( $d$ -MST) problem, one of the most practically important of the  $NP$ -hard MST problems. This review will provide a basis for designing operators, encodings and heuristics for tackling the less well-researched multi-criterion MST problem.

In a network or graph, the *degree* of a vertex is the number of edges that are incident to it. In many real-world network applications, the vertices are subject to a degree-constraint, limiting the number of edges that can be connected to one vertex. For example, in a communications

network, many exchanges or switches can only be physically connected to a limited number of linking wires. Also, when designing a network for maximum reliability, introducing a degree-constraint limits the damage that may be caused by a single exchange failure. Unlike the MST, the  $d$ -MST of a graph cannot, in general, be found using a polynomial time algorithm. In fact, for a given rational number  $R \geq 1$ , finding a spanning tree of maximum degree at most  $d$ , and of total weight at most  $R$  times that of the optimal solution, is  $NP$ -hard [RMR<sup>+</sup>93].

Several different heuristic approaches for solving the  $d$ -MST problem have been taken in the literature. Narula and Ho [NH80] investigated three methods, one of which (a branch and bound algorithm) is guaranteed to converge to a globally optimal solution. In their methods, good upper bounds are generated by a modification to Prim's algorithm [WW90]; the resulting algorithm, we will refer to as *d-Prim's*. Savelsbergh and Volgenant [SV85] also employed a branch and bound technique with improved heuristics leading to considerably faster run-times than those achieved by Narula and Ho. Random, non-Euclidean graphs were found to be far less tractable than Euclidean graphs, however, with the CPU time needed to solve standard instances rising from a mean of 0.2 seconds ( $SD = 0.7s$ ) for  $n = 30$  to a mean of 74.2 seconds for  $n = 70$  ( $SD = 369.5s$ ), where  $n$  is the number of vertices in the graphs. Khuller *et al* [KRY96] show that for an arbitrary collection of points in the plane there exists a degree-3 spanning tree of at most 1.5 times the MST and a degree-4 spanning tree of at most 1.25 times the MST. They also give algorithms that compute these trees in  $O(n)$  time, given an MST as part of the input. They also extend these results to Euclidean points in  $n$ -dimensional space. Fekete *et al* [FKK<sup>+</sup>97] report a network flow technique which improves upon these results, working again in linear time. They do not consider the more difficult non-Euclidean case, however. Recent research by Boldon *et al* [BDK95] describes a 'dual simplex' approach, based on Prim's algorithm, which produces good results on random, non-Euclidean graphs where the underlying MST has degree up to twenty and the degree-constraint is set to four.

Algorithms for the  $d$ -MST are often designed with Euclidean graphs in mind, and/or tested only on Euclidean graphs. However, although the cost function defined on links can sometimes be closely related to Euclidean distance, this relationship is often confounded in practice. For example, where link costs are defined to be communications costs between nodes, physical distance can be a very minor factor in comparison to others such as the type, quality, maintainability, speed, and corporate provider of the link in question. In general, therefore, it is valuable to test and compare algorithms for the  $d$ -MST on non-Euclidean graphs. In such cases, the algorithms of Khuller *et al* and Fekete *et al* are not applicable, while Savelsbergh and Volgenant's are cumbersome.

### A comparison of methods

In 1999, an empirical comparison of algorithms for the  $d$ -MST was carried out by Krishnamoorthy *et al.* [KES99]. In it they discuss the advantages of different encoding techniques for trees, including simple, direct representations of the edges (arcs in Krishnamoorthy *et al.*'s terminology); predecessor trees [Li00]; Prüfer numbering [ZG99, Prü18]; and a combined Prüfer/permuation-based encoding that they introduce for the first time. The performance of a range of heuristics including genetic algorithms, simulated annealing, and a hybrid constructive technique, as well as exact algorithms based on branch and bound, and Lagrangian relaxation, are compared.

Four types of problem are described and used in the paper. The first are 2-d Euclidean problems, some instances with points scattered uniformly randomly in a square, and others taken from TSPlib. Second, there are Euclidean problems in a higher dimension,  $d > 2$ , also with a uniformly random distribution of vertices, obtained from Volgenant. Third, there are 'structured' (STR) problems where the nodes are in tight, well-separated clusters in a high dimensional Euclidean space. Fourth, are structured problems that are particularly hard (and contrived) using non-Euclidean distances. These SHRD problems are constructed by assigning lengths of  $l$  to all edges incident to the first node,  $2l$  for the second node (bar the one connecting it to node 1) and so on. Afterwards, all of these weights are perturbed by adding a uniform random deviate in  $[0.05l, 0.9l]$  to all the edge weights.

From a large number of experiments, the authors present the average solution quality and computation time on instances of differing size, constraint setting, and Euclidean dimension (where applicable), on all of the problem types. They find the exact methods — Lagrangian relaxation and branch and bound — to be most effective (both in terms of final solution quality, and CPU time) for the simpler unstructured problems in Euclidean spaces. However, for the structured problems the GA using a Prüfer coding is best, and for the more contrived structured hard problems (SHRD) both their simulated annealing algorithm, and more particularly their randomized constructive approach, PSS (similar to Raidl and Jusltrom's weight coding), are best.

### The randomized primal method

In [KC00a], the primal method of Narula and Ho [NH80],  $d$ -Prim, which constructs a low-weight degree-constrained spanning tree, was adapted into a decoder encoding for use in general-purpose optimization algorithms. This method, called RPM (Randomized Primal Method) was employed in three different search algorithms: multi-start hillclimbing, sim-

ulated annealing and a genetic algorithm. The quality of solutions found by these search techniques were compared with each other, and also with those achieved by the dual simplex algorithm of Boldon *et al* [BDK95] and the upper bound generated by Narula and Ho's *d*-Prim's method. These comparisons were made on a variety of randomly generated *d*-MST problems of from 50 to 250 nodes. Some of the random graphs used for testing were created to be particularly difficult and misleading in order to emphasize differences in algorithm performance.

Results showed that a genetic algorithm was able to take advantage of RPM most effectively in terms of solution cost returned. It was found that the genetic algorithm employing RPM consistently finds better solutions than both BF2 (the dual simplex approach of Boldon *et al*) [BDK95] and *d*-Prim's (the upper bound technique of Narula and Ho) [NH80], and the advantage is most pronounced on the more challenging graphs.

The genetic algorithm with RPM was also compared with the Prüfer number encoded GA proposed by Zhou and Gen [ZG97]. It is able to find the optimal solution to a small 9-node problem, used by Zhou and Gen, in far fewer evaluations and far more reliably than the GA using a Prüfer number encoding.

### Weighted coding GA

Shortly after the paper of Knowles and Corne, Raidl and Julstrom [RJ00] proposed an approach to the *d*-MST using a weighted coding. Like the approach of Knowles and Corne, the weighted coding uses *d*-Prim's algorithm to build potential solutions to the problem. However, in a similar manner to Boldon's dual simplex approach, this is achieved by temporarily biasing the graph's edge costs. The biases applied are encoded for by the chromosomes in the population. This decoding algorithm enforces the degree constraint, so that all chromosomes represent valid solutions and there is no need to discard, repair, or penalize invalid chromosomes. On the set of graphs put forward in [KC00a], whose unconstrained minimum spanning trees are of high degree, a genetic algorithm using the weighted coding identified degree-constrained minimum spanning trees that were shorter than those found by either *d*-Prim's, the dual-simplex approach of Boldon *et al.*, or the RPM GA of Knowles and Corne.

### An efficient EA using a direct encoding

In a further paper by Raidl [Rai00], a direct representation and specialized initialization, mutation, and crossover operators were presented. The initialization is based on a perturba-

tion of Kruskal's classic algorithm, and works in  $O(|V| \log |V|)$  time. The other operators are designed to operate efficiently, in just  $O(|V|)$  time, giving an algorithm that can be run effectively even on very large graphs. The direct representation and operators are also designed to exhibit very good locality while also never generating infeasible or constraint-violating spanning trees. Using efficient data structures, all of these operations can be performed in  $O(|V|)$  time.

Raidl tests his evolutionary algorithm on two sets of test problems ranging in size from 15 vertices to 500. The first set of problems is taken from Krishnamoorthy *et al.* [KES99], described above. Comparison is made between Raidl and Julstrom's weight-coded EA, the new direct-coded EA, and the algorithms tested by Krishnamoorthy *et al.*. The results show that on the smaller problems, up to 25 nodes, the direct-coded EA is best (of all algorithms) in terms of solution quality, while the weight-coded EA is best (of all algorithms) on the largest 30 node problem. It is also clear from the run-times reported that the direct-coded EA has linear time operators, whereas the weight-coded EA has not.

The second set of problems are from Knowles [KC00a], with some additional equivalently structured problems of larger size (up to 500 vertices). The direct-coded and weight-coded EA are compared once more, but this time results are reported against Knowles and Corne's EA with RPM encoding, and the dual simplex approach of Boldon *et al.* [BDK95]. In all problems the direct-coded EA of Raidl finds the best solutions. Once again, its linear time complexity is also demonstrated.

### Variable neighbourhood search

Finally, a variable neighbourhood search (VNS) algorithm for the  $d$ -MST was recently proposed by Ribeiro and Souza [RS01]. Beginning with a random feasible tree, the VNS algorithm tries to improve it at each of increasing orders of neighbourhood moves. It does this by producing a low-order neighbour of the current solution, applying a local descent procedure to it, and checking if this is better than the original tree. If it is, then the process is repeated at the same neighbourhood order. However, if the locally improved mutant of the current solution is worse, then the current solution is retained and a neighbour at the next order up is generated and locally improved. The algorithm stops when this whole process has been repeated  $seqs$  times where  $seqs$  is a parameter. The local improvement procedure is itself a variable neighbourhood method too. It uses a local search procedure which begins with single edge exchanges and then goes on to use both two- and three- edge exchanges when exchanges at the previous level fail to produce an improvement. Both the two- and three- edge exchange operations, work by first making exchanges that violate a degree-constraint

and then restoring feasibility with the final exchange.

The reported results of the VNS are very impressive for the STR and SHRD problems of Krishnamoorthy and on the simpler problems of Zhou and Gen [ZG97], far outperforming the range of methods used by Krishnamoorthy (in computation time and solution quality found), and the genetic algorithm of Zhou and Gen.

### 7.1.5 The mc-MST problem

In the multi-criterion MST (mc-MST) problem, a vector of weights is defined for each edge, and the problem is to find all Pareto optimal (or efficient) spanning trees (see Figure 7.1). This is *NP*-hard [Ehr00] even in its unconstrained form. The mc-MST is commonly found in network-design oriented applications. For example, one set of weights may be associated with financial cost, while another may be associated with link reliability.

Recently, there has been much interest in the mc-MST in the operations research literature, e.g. [Ehr00, HR94, RASG98]. But, there has, to our knowledge, been only one EA previously proposed for the mc-MST: a Prüfer-encoded multiobjective GA proposed by Zhou and Gen [ZG99], which uses Srinivas and Deb's Non-Dominated Sorting method [SD95] and a Prüfer based encoding. They apply their method to randomly generated bi-objective mc-MSTs ranging from 10 nodes to 50 nodes in size. An important element of their paper is a comparison between their EA and the results from an enumerative method (which is meant to find all true Pareto optimal solutions) which they propose in the same paper. Although the enumerative method is of course infeasible for large problems, comparisons of their EA with the enumerative method were meant to give an idea of how much of the true Pareto front the EA finds on small problems. However, in trying to replicate their enumerative method we found that it is flawed, since it neither generates all Pareto optimal points, nor guarantees that those points that it does generate are Pareto optimal. A proof of this result is given in Appendix D and published in [KC01b].

Furthermore, the problems Zhou and Gen used to test their genetic algorithm were very simple and could be much better solved using good exact methods [RASG98] (smaller problems), or heuristic approximation methods [EK97, HR94] (larger problems). In fact, we show in this chapter that a method which simply scalarizes the weights on each edge and applies Prim's algorithm to the resulting single-objective MST can obtain a better nondominated front than Zhou and Gen's GA, on some mc-MST instances, when the former is iterated for different scalarizing weight vectors.

Nonetheless, some instances of the mc-MST may not be easily or efficiently solved by such

exact or heuristic methods. One example is when the number of objectives is greater than two. It is then relatively straightforward for a multiobjective EA to be applied, whereas some exact methods [RASG98] and heuristics have only been developed for the bi-objective case. Furthermore, when other constraints need to be incorporated then this may be achieved relatively easily in an evolutionary algorithm, using methods developed for other problems, whereas this can be troublesome in some of the pure heuristic approaches. Unfortunately, these more difficult mc-MST problems are not generally available and have not yet been considered in other algorithm studies.

In order to aid in the further development of good evolutionary algorithms and other meta-heuristics for a diverse variety of mc-MST instances, it would be useful to have a collection of parameterized problem generators to provide benchmark problems that could be used in comparative studies. In this chapter we present such a collection of simple problem generators. These can provide problems with different features, controlled by the generator's parameters. In particular, we can generate graphs with correlated and anti-correlated weight components (which critically affect the shape of the Pareto front, and therefore the applicability of different methods), problems with large regions that have no solutions on the convex hull of the Pareto front, and problems which are especially difficult to solve when a degree constraint is additionally imposed. All of the generators can generate problems of different sizes, with a differing degree of sparsity, and with any number of objectives (except for the concave graph generator).

These problem generators may also be used for testing the general strengths and weaknesses of multiobjective EAs, as part of a wider test problem suite. Much progress has been made recently in improving the variety, difficulty, and actual use of, test problem suites by researchers in the evolutionary multiobjective optimization (EMO) community. However, there is still a shortage of parameterized combinatorial problems, constrained problems, and problems with large numbers of objectives. The generators proposed here can provide problems with all these features, either in isolation or together in one instance.

### 7.1.6 Organization of this chapter

In the next section we formally define the mc-MST and *mcd*-MST problem. We then describe a simple iterative constructive approach which can find one or more supported efficient solutions to the problem in polynomial time. This algorithm is later used as a baseline for measuring the efficacy of the EA approaches. In Section 7.4, three alternative representations (and associated operators) are described, including the Prüfer encoding favoured by Zhou and Gen, and two of our own based on encodings for the *d*-MST, described above.

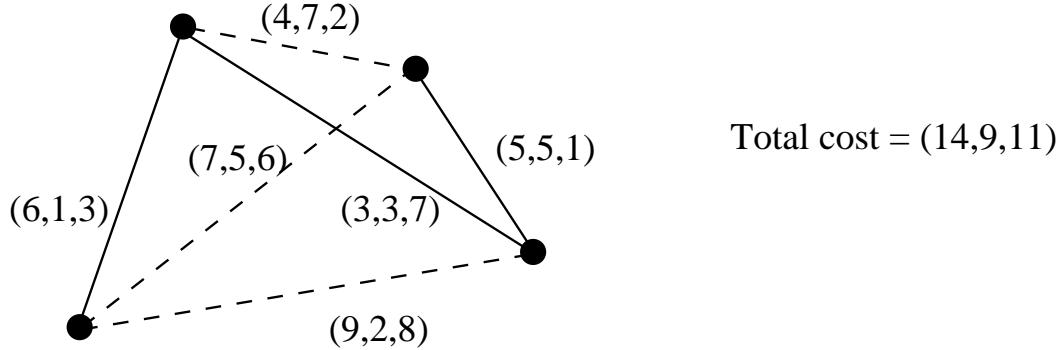


Figure 7.1: The mc-MST problem. Each edge in the graph has a vector of costs associated with it, for example: construction, maintenance, and delay, in a telecommunications network design scenario. The problem is to find all spanning trees in the graph that “minimize” the vector of summed costs. This is a Pareto optimization problem. It is *NP-hard* even without constraints because there may be exponentially many efficient spanning trees. An efficient tree is shown (solid lines) together with its associated cost vector.

Section 7.5 describes a set of problem generators for producing benchmark instances of the mc-MST problem of various types, with and without constraints.

Our experiments are in two parts, described in Section 7.7. Initially, we test the Prüfer encoding of Zhou and Gen on a set of simple random weight mc-MST instances without a degree constraint. This places our second set of experiments in context. In the second part, we use our two proposed representations (and associated operators) to tackle 15 benchmark problem instances with degree constraints. The performance of PAES, an EA called AESSEA, and M-PAES are compared. The results of both sets of experiments are described in Section 7.8 and the chapter is concluded in Section 7.9.

## 7.2 The mc-MST problem

A spanning tree of an undirected, connected graph,  $G = (V, E)$ , is a subgraph  $T = (V, E_T)$ , with  $E_T \subset E$ , such that  $T$  contains all vertices in  $V$  and connects them with exactly  $|V| - 1$  edges, so that there are no cycles. If  $G$  is complete, then the set  $S$  of spanning trees  $T$  of  $G$  has  $|S| = |V|^{V-2}$  members. If each edge  $(i, j) \in E$  has  $K > 1$  associated non-negative real numbers, representing  $K$  attributes defined on it and denoted with  $\mathbf{w}_{i,j} = (w_{i,j}^1, w_{i,j}^2, \dots, w_{i,j}^K)$ , then the mc-MST problem may be defined as:

$$\begin{aligned}
 \text{"minimize" } \mathbf{W} &= (W^1, W^2, \dots, W^K) \\
 \text{with } W^k &= \sum_{(i,j) \in E_T} w_{i,j}^k, \quad k \in 1..K
 \end{aligned} \tag{7.1}$$

where the term ‘minimize’ is in quotation marks to indicate that it may not be possible to find a single solution that is minimal on all the components of  $\mathbf{W}$ . Instead, one is required to find a set of spanning trees  $S^* \subset S$ , called the Pareto optimal set, with the property that:

$$\forall T^* \in S^*, \forall T \in S, T \prec T^* \tag{7.2}$$

where  $T \prec T^* \iff \forall k \in 1..K, W^k \leq W^{k*} \wedge \exists k \in 1..K, W^k < W^{k*}$ . The expression  $T \prec T^*$  is read as  $T$  *dominates*  $T^*$ , and solutions in the Pareto optimal set are also known as efficient or admissible solutions.

If there is, in addition, a constraint  $d$  on the maximum vertex degree in the spanning tree, then the problem is called the multi-criterion degree-constrained minimum spanning tree (mcd-MST) problem.

### 7.3 Finding the supported efficient solutions

We now describe an efficient way of finding at least a subset of the true Pareto optimal solutions to an instance of the mc-MST. First we recall Prim’s classic algorithm for the single objective MST, and then we indicate how this can be embedded in a slightly more sophisticated procedure for the multiobjective case.

Prim’s algorithm uses two sets of vertices,  $C$  (connected) and  $U$  (unconnected). Initially,  $C$  contains a single, arbitrarily chosen vertex, while  $U$  contains all of the remaining vertices. The algorithm proceeds by moving vertices from  $U$  to  $C$  one at a time until  $U$  is empty. Each such move is associated with the addition of a specific edge to the growing tree. When  $U$  is empty, the tree is complete. The edge chosen at each step is one of minimal weight among those for which the following is true: it connects vertex  $u \in U$  to  $c \in C$ , and its addition will not introduce a cycle. Clearly, having decided to add edge  $(u, c)$ , vertex  $u$  is the one moved from  $U$  to  $C$ .

This procedure quickly and efficiently solves the single objective MST. A simple modification, in which we add an extra obvious feasibility test when moving a vertex between the two sets, enables it to build solutions to the  $d$ -MST, but in that case it is an approximate heuristic

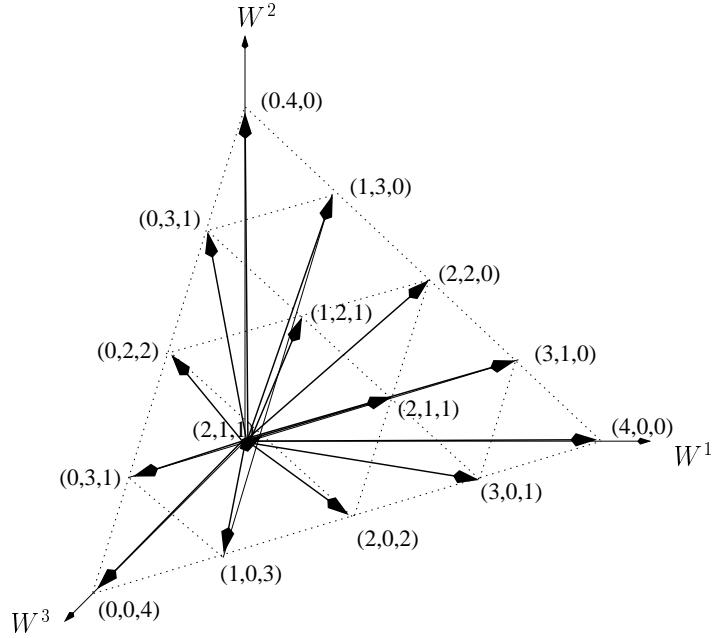


Figure 7.2: A set of evenly distributed  $\lambda$  vectors can be generated by using every normalized scalarizing vector,  $\lambda$ , with components equal to  $l/s$ ,  $l = 0..s$  where  $s$  is a parameter controlling the number of different vectors that will be generated. This gives  $s+K-1 \choose K-1$  different scalarizing vectors.

rather than an exact method [NH80].

In preparation for the multiobjective version of Prim's, we first observe that a true Pareto optimal solutions to the mc-MST problem can be found by simply using Prim's algorithm (or any algorithm for the single-objective MST), by substituting the weight vector at each vertex with a weighted sum. This may be achieved since, for any scalarizing vector  $\lambda = (\lambda^1, \lambda^2, \dots, \lambda^K)$  with  $\sum_{k=1}^K \lambda^k = 1$ , and  $\lambda_k \geq 0$ ,  $k = 1..K$ , the following holds:

$$\sum_{(i,j) \in E_T} \lambda \cdot \mathbf{w}_{i,j} = \sum_{k=1}^K \lambda^k \left( \sum_{(i,j) \in E_T} w_{i,j}^k \right). \quad (7.3)$$

A tree which optimizes the term on the left will be a solution to the MST for the single-objective problem defined by the scalarization, and hence easily found by Prim's algorithm. But the rewrite of this term on the right reveals that this tree must also be on the true Pareto front of the corresponding mc-MST. If not, a tree exists which must improve one or more of the inner summed weight terms, contradicting the assumption that this sum was already minimal.

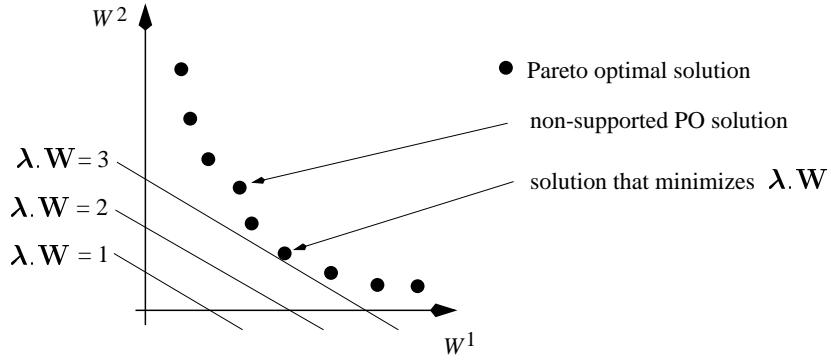


Figure 7.3: Supported and non-supported solutions. The supported Pareto optimal solutions are minima of  $\lambda \cdot \mathbf{W}$  with a particular  $\lambda$ . The non-supported Pareto optimal solutions are *not* minima of  $\lambda \cdot \mathbf{W}$  for *any*  $\lambda$ .

So, we can find an optimal solution to the mc-MST by replacing the vector weights defined on each edge in  $G$  by a scalar weight  $b$  formed by taking the inner product of  $\lambda$  and  $\mathbf{w}$ ,  $b = \lambda \cdot \mathbf{w}$ , and then finding a spanning tree that minimizes the sum of the scalarized edge weights.

The algorithm, mc-Prim, shown in Figure 7.4 iteratively changes the scalarizing vector  $\lambda$  and uses Prim's algorithm to find a set of optimal solutions to an mc-MST. At each iteration of the main loop, a new scalarizing vector  $\lambda$  is generated by the function `nextvector( $\lambda$ )`, which successively generates every normalized scalarizing vector,  $\lambda$ , with components equal to  $l/s, l = 0..s$  where  $s$  is a parameter controlling the number of different vectors that will be generated. This gives  ${}^s C_K$  different scalarizing vectors (see Figure 7.2). For each different  $\lambda$  vector, Prim's algorithm is applied. This begins with the selection of a random vertex from  $V$  using the function `rand( $V$ )`, and at each subsequent step a minimum weight edge, that connects a vertex in the connected list to one in the unconnected list, is added to  $E_T$ .

For a large number of different scalarizing vectors (in our experiments we set  $s = 1000$ , giving 1001 different  $\lambda$  vectors), mc-Prim may generate an approximation to  $S^*$ , the set of Pareto optimal spanning trees, that is satisfactory in many cases. However, in general mc-Prim cannot usually discover the complete set  $S^*$  because it can only find solutions on the convex hull of the true Pareto front — the so-called *supported* efficient solutions (see Figure 7.3). Such trees are Pareto optimal, but not minima of any single-objective scalarization.

---

**Algorithm: *mc-Prim***

---

*Data:*

$U$  is the unconnected set of vertices  
 $C$  is the connected set of vertices  
 $E_T$  is the edge-set of the spanning tree  $T$   
 $S$  is the set of nondominated spanning trees

---

```
 $S \leftarrow \emptyset$ 
 $i \leftarrow 0$ 
while ( $i < {}^s C_K$ ) /* Main Loop */
   $\lambda \leftarrow \text{nextvector}(\lambda)$ 
   $E_T \leftarrow \emptyset$ 
   $U \leftarrow V$ 
   $C \leftarrow \emptyset$ 
   $v \leftarrow \text{rand}(V)$  /* Begin Prim's Algorithm */
   $C \leftarrow C \cup \{v\}$ 
   $U \leftarrow U \setminus \{v\}$ 
  while ( $|C| < |V|$ )
    select an edge  $(u, v)$  with  $u \in C, v \in U$  so that
     $\forall i \in C, j \in U \sum_{k=1}^K \lambda^k \cdot w_{u,v}^k \leq \sum_{k=1}^K \lambda^k \cdot w_{i,j}^k$ 
     $E_T \leftarrow E_T \cup \{(u, v)\}$ 
     $C \leftarrow C \cup \{v\}$ 
     $U \leftarrow U \setminus \{v\}$ 
   $S \leftarrow S \cup T \leftarrow (V, E_T)$ 
   $i \leftarrow i + 1$ 
return ( $S$ ) /* Termination */
```

---

Figure 7.4: mc-Prim.

## 7.4 Representations and operators

We make use of three distinct representation for use in the evolutionary and memetic algorithms, each with associated operators specific to it. These are described below. The first, RPM, is a decoder representation, initially proposed by us for tackling the  $d$ -MST problem, and adapted here for the mc-MST. The second is the Prüfer encoding favoured by some researchers for tackling spanning tree problems. The third employs a direct encoding of the tree edges. This method employs the RPM for initialization.

### 7.4.1 The randomized primal method

The randomized primal method (RPM) was first proposed by Knowles and Corne [KC00a] for tackling the  $d$ -MST problem. It has the advantages that it only generates feasible spanning trees which do not violate the required degree constraint, and it uses edge weight information to strongly bias the choice of edges towards lighter ones. It can also be used with standard crossover and mutation operators, making it relatively simple to implement. On the negative side, Raidl and Julstrom [RJ00] and Raidl [Rai00] have obtained better performance with a weighted coding and with a direct representation and associated (specialized) operators, respectively. Nonetheless, RPM is worth investigating alongside the methods of Raidl, on the mc-MST problem, as it has exhibited good performance previously.

RPM employs a tabular chromosome of length  $n$  and depth  $d - 1$ , where  $n$  is the number of vertices in the presented graph and  $d$  is the required degree constraint. It decodes the chromosome to iteratively construct a degree-constrained spanning tree  $T$  with vertex set  $V$  and edgeset  $E_T$ , initially empty. Initially all vertices are in a set  $U$  (unconnected) and the set of connected vertices  $C$  is empty. The decoder proceeds as follows:

First step : Move an arbitrary vertex<sup>1</sup> from  $U$  to  $C$ .

General step : Add an edge  $e$  to  $E_T$  which joins a vertex  $v$  in  $C$  whose current degree  $d_v < d$ , to a vertex  $w$  in  $U$ . Move  $w$  from  $U$  to  $C$  and increment the current degrees,  $d_v$  and  $d_w$  of  $v$  and  $w$ .

The particular edge chosen in the General step above is dependent upon the edge weights in the underlying graph and the values stored in the tabular chromosome. The general step is implemented in the following way:

---

<sup>1</sup>For example, use vertex 1.

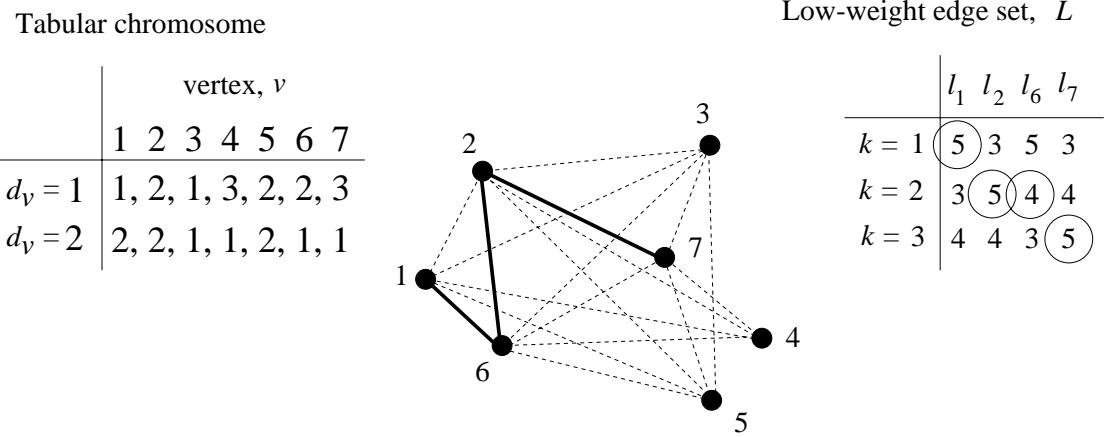


Figure 7.5: A schematic of the randomized primal method. The 7-vertex tree under construction has a degree constraint of  $d = 3$ . The tabular chromosome thus has length 7 and depth 2. The sorted lists,  $l_v$ , for each connected vertex form a table, shown on the right. The entries are the vertices  $w$  which the respective connected vertex  $v$  could feasibly be connected to, listed in weight order, lightest at the top. The circled entries are the ones encoded for by the tabular chromosome. These circled entries form the low-weight edge set  $L$ , from which the lightest edge is chosen to join the tree. In this case, the edge 7-5 is the shortest of the circled entries, assuming weights are Euclidean distance.

1. For each vertex  $v$  in  $C$  whose current degree  $d_v$  is less than  $d$ , construct a sorted list  $l_v$  containing each of the vertex's incident edges which join it to vertices in  $U$ , in descending order of weight.
2. For each list  $l_v$  look up the allele value  $a_{(v,d_v)}$  stored in the tabular chromosome in position  $(v, d_v)$ . Place the  $k$ th lightest edge from list  $l_v$  into a low-weight edge set  $L$ , where  $k = \min(a_{(v,d_v)}, |l_v|)$ .
3. Select the lowest weighted edge in  $L$ .

RPM builds the same tree as the  $d$ -Prim algorithm if every gene in the chromosome has allele value 1. Larger allele values in the chromosome will lead to heavier edges being included in the constructed tree. In our work on the  $d$ -MST problem [KC00a], chromosomes were initialized using a negative exponential distribution giving a strong bias towards lower allele values. The mutation operator was applied to each gene with probability 1%, and sets it to an allele value drawn randomly from the same distribution as in the initialization phase. Standard uniform crossover was employed.

To adapt the RPM for use in mc-MST and mcd-MST problems the edge weight vectors can

be scalarized using a scalarizing weight vector  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_K)$ . Each chromosome is then decoded into a spanning tree using the RPM with the vector of edge weights replaced by a set of scalar edge weights,  $b_{i,j}$ , produced by taking the inner product of the scalarizing vector  $\boldsymbol{\lambda}$  and  $\mathbf{w}$ :  $\forall (i,j) \in E, b_{i,j} = \boldsymbol{\lambda} \cdot \mathbf{w}_{i,j}$ . The scalarizing vector forms part of the chromosome, and is initialized from a uniformly random distribution such that  $\sum_{k=1}^K \lambda_k = 1$  and  $\forall k \in 1..K, \lambda_k \geq 0$ .

The other genes in the chromosome are initialized with a strong bias towards low values, and hence towards choosing lower cost edges. Specifically, each gene is set to an allele value  $q = \lfloor e^{xp \log(|V|)} \rfloor$ , where  $x$  is a uniformly distributed random number in  $[0, 1)$ , and  $p$  is a parameter that controls how strong is the bias towards lower allele values. We set  $p = 2$  in our experiments.

The mutation operator for RPM (`mutate_RPM`) works as follows. Each gene is mutated with some small probability  $p_m$ . For the tabular part of the chromosome, the gene's allele value is set to a new value drawn from the same distribution as at initialization. The gene representing the scalarizing vector is made up of  $K$  component values,  $\lambda^1, \lambda^2, \dots, \lambda^K$ , with  $\sum_{k=1}^K \lambda^k = 1$ . To mutate this gene a Gaussian deviate with mean zero and standard deviation of 0.05 is added to each component, and the component values are then re-normalized so that they sum to one.

### 7.4.2 The Prüfer encoding

Cayley [Cay89] proved that there are  $n^{n-2}$  distinct labeled trees for a complete graph with  $n$  vertices. By providing a constructive proof of this, Prüfer established a bijective mapping between the set of distinct labeled trees and the set of all combinations of  $n - 2$   $n$ -ary digits from  $n$  [Prü18]. The details of the encoding are fairly well known and can be found in [ZG99], for example. Using this encoding, a chromosome is simply a string of  $n - 2$  digits, where each can vary independently between 1 and  $n$ . Following [ZG99], our implementation of a Prüfer encoding EA uses single-gene random mutation (a gene is chosen at random, and changed to any new integer between 1 and  $n$ ), and uniform crossover [Sys89].

### 7.4.3 Direct tree encoding and operators with RPM initialization

In order to tackle the  $d$ -MST problem, Raidl [Rai00] used a direct encoding of a spanning tree, and associated genetic operators. In the direct encoding, a spanning tree is explicitly represented by the list of edges which comprise it; Raidl calls this the ‘edge-set’ representa-

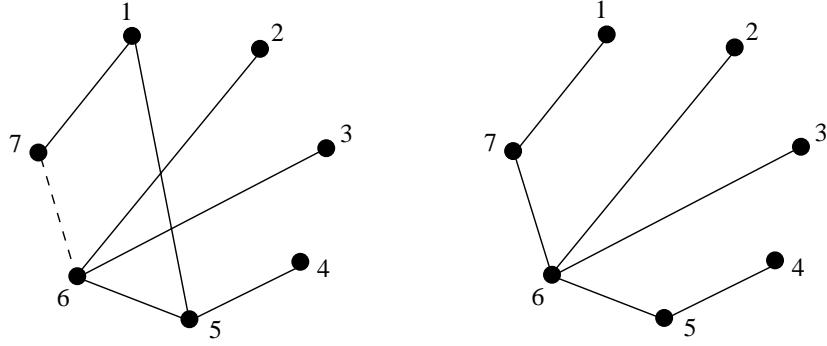


Figure 7.6: Edge mutation: choose an edge to add into the tree where this choice is biased towards cheaper edges; remove a random edge from the cycle created. The edge mutation operator works in  $O(n)$  time where  $n$  is the number of vertices in the graph.

tion. In Raidl's approach to the  $d$ -MST problem, he used a procedure based on Kruskal's algorithm [Kru56] to generate initial feasible trees which respect the degree constraint.

Raidl's 'edge mutation' operator simply replaces one of the existing edges of a tree with a new edge, all the time respecting feasibility constraints. In operation, it works as follows. First, an edge currently not in the tree is chosen to insert; this choice is biased towards lower cost edges. Naturally, this will temporarily introduce a cycle. A random choice among the edges in that cycle is then made (excluding the new edge), and the selected edge is removed from the tree.

Raidl's 'edge crossover' operator is constructed on the principle of inheriting as many edges as possible from two parent trees. Initially, a child is set to contain the intersection of its parents' edge-sets, and hence will almost always be a forest of unconnected non-spanning trees. Next, remaining edges (from those which were in one parent but not both) are successively selected, and included in the tree if such inclusion is feasible. In order to bias this step towards considering the lower cost edges, Raidl uses binary tournament selection. At the end of this process, if the child is not yet a spanning tree, edges between its components are chosen randomly until the required  $d$ -ST is produced.

To adapt the edge-set encoding and associated operators for use on the mc-MST problem, we use a scalarizing vector  $\lambda$ , which is encoded as a gene on the chromosome, to scalarize the vectors of edge weights. The mutation and crossover operators are then adapted to take into account this extra gene. The initialization procedure we use employs the RPM encoding, described above, with the same parameters for setting the initial chromosome values and scalarizing vector.

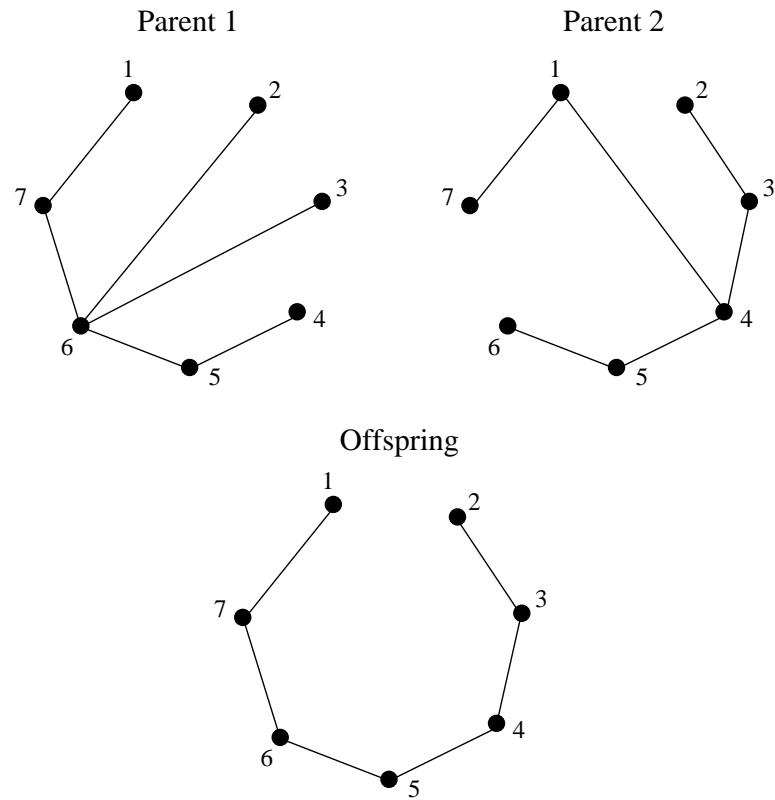


Figure 7.7: Edge crossover. To construct an offspring from two parents: 1. include all the edges that appear in *both* parents; 2. add in edges that appear in either parent but not both, adding in cheaper edges first (this is achieved using binary tournament selection of edges); 3. if the edges do not form a tree after step 2, add in further edges, randomly.

After the initialization procedure, the direct tree encoding only is used. In our adapted edge crossover, the offspring inherits the scalarizing vector  $\lambda$  from parent  $\mathbf{a}$ . Then the scalarized weights  $b_{i,j}$  are used in the tournament selection of edges used in the edge union operation of the heuristic edge crossover.

Adapting the edge mutation operator is a little more complicated. In Raidl's operator, the edges are sorted in order of weight (once at the beginning of the algorithm) and the heuristic mutation operator uses this sorted list to bias the selection of an edge to be inserted, towards cheaper edges. In the mc-MST problem, it is not possible to order the edges by weight, so this method cannot be applied directly. However, if the vector weights of edges are replaced by scalar edge weights  $c_{i,j}$ , produced by taking the inner product of the edge weight vectors with a scalarizing vector  $\mu$ , with  $\sum_{k=1}^K \mu_k = 1$  and  $\forall k \in 1..K, \mu_k \geq 0$ :  $\forall (i,j) \in E, c_{i,j} = \mu \cdot \mathbf{w}_{i,j}$ , then the edges can be sorted in increasing order of  $c$ , and the bias can be applied using this sorted list. We could use the scalarizing vector  $\lambda$  that is associated with each solution vector, putting  $\mu = \lambda$ , to generate all the  $c_{i,j}$ , but we do not wish to have to re-sort the edges for each different  $\lambda$  vector, as this would strongly compromise the computational efficiency of the mutation operator. To get around this, we pre-sort the edges for a small sample of different, evenly distributed  $\mu$  vectors at the beginning of the algorithm. In the mutation operator itself, we first mutate the  $\lambda$  vector of the current solution vector by adding a Gaussian random deviate of mean zero and standard deviation  $\sigma = 0.05$  to each of the components of  $\lambda$  and then re-normalizing it. Then, when choosing an edge to insert into the current tree, the sorted edge list that was generated using a  $\mu$  vector which most closely matches the updated  $\lambda$  vector of the solution undergoing mutation is used. In our experiments, 5 different representative  $\mu$  vectors:  $(0,1), (0.25,0.75), (0.5,0.5), (0.75,0.25), (1,0)$  are defined, so that just 5 edge-sorts must be performed (just once) at the beginning of the algorithm. Thus, our adaptions of Raidl's operators preserve their linear time complexity whilst also maintaining the heuristic nature of their biasing edge choices towards lower-weighted edges.

## 7.5 Benchmark problem generators

We propose generators for the following types of non-Euclidean problems:

**Random:** Random (uncorrelated) integer or real number weighted graphs;

**Correlated:** Random correlated real number weighted graphs;

**Anti-Correlated:** Random anti-correlated real number weighted graphs;

**M-Correlated:** Correlated real number weighted graphs with high vertex degree in the underlying MST;

**Concave:** Real number weighted graphs that have a large concave region in their Pareto front.

All of the above can be generated as either sparse graphs, or complete graphs. We consider only complete graphs in this paper. Similarly, all but the concave graphs can be generated with an arbitrary number of objectives,  $K$ , although here we restrict our attention to 2-objective problems only.

### 7.5.1 Random

The random graph generator simply sets each component of each edge weight vector to a value drawn from a uniformly random distribution within some range,  $\mathcal{U}(\min, \max)$ . In this paper we do not consider graphs of this type but we have already shown (above and published in [KC01a]) that our AESSEA algorithm using a direct coding and specialized operators is superior to the AESSEA algorithm using a Prüfer encoding [Prü18, ZG99] on problems of this type. Here we restrict our attention to more difficult problem types.

### 7.5.2 Correlated and anti-correlated

The correlated (and anti-correlated) graphs are generated by using the algorithm given in Figure 7.10. The procedure takes the required correlation  $\alpha \in [-1, 1]$  as an argument and returns a weight vector of  $K$  weights where the first component is drawn from a uniform distribution, and all subsequent weights are either positively or negatively correlated with respect to the *first* component, and lie within the same range of values. Note that since the correlation exists between the first and each other component of the weight vector, a correlation of  $|\alpha|$  exists between all pairs of components  $w^k, w^l$   $k, l \in 1..K$ . The correlation between the components of a weight vector affects the shape of the associated Pareto front of the MST problem of the graph. A zero correlation gives a smooth, convex Pareto front with a fairly constantly varying gradient along its length. In contrast, a large positive correlation gives a convex Pareto front with more of a discontinuous change in the gradient. With a correlation approaching +1, the front becomes smaller until in the limit, it will only contain one optimal point. With a strong negative correlation, the front is convex but approaches a straight line or flat surface in objective space. Because of this there tend to be a large number of non-supported efficient solutions (those that do not lie on the convex hull of the Pareto

front). This shape of Pareto front might make it difficult for methods based on the use of scalarizing weight vectors to find a good approximation to the Pareto front since they tend to find it difficult to discover non-supported solutions, and also rely on a changing gradient in the Pareto front to find a good range of points on it.

### 7.5.3 M-correlated

The M-correlated graph generator is based on a graph generator developed by us [KC00a] for producing ‘misleading’ or M-graph problems for the standard (single-objective)  $d$ -MST problem, and combining this with the correlated graph generator described above. The M-correlated graphs are designed to be particularly difficult to solve when a low maximum vertex degree constraint must be satisfied.

In theory, the maximum vertex degree of a MST in a graph of random edge weights is  $|V| - 1$ . However, in practice, when reasonably large uniformly random weight graphs are generated, the maximum vertex degree of the graph’s MST rarely exceeds four or five. Due to this fact, some researchers [BDK95, KES99] have developed methods for generating biased random graphs where the graphs’ MSTs have a high maximum vertex degree. Knowles and Corne [KC00a] further developed the graph generator of Boldon *et al.* to bias the edge weights in such a way as to mislead any algorithm that greedily chooses edges of low weight in an attempt to grow a low-weight spanning tree. The M-graph generator, as it is called, requires four parameters to be set:

- $|V|$  the number of vertices in the graph;
- $f$  the number of vertices with large degree;
- $ld$  the lower bound on the degree of large-degree vertices; and
- $ud$  the upper bound on the degree of large-degree vertices,

with the constraints that  $f.ud < |V|$  and  $ld < ud$ . The generator has two main stages. The construction of an M-graph is illustrated in Figures 7.11–7.15. In the first stage a spanning tree that will be the MST of the graph is formed. In the second stage other edges are added to form a graph of the required density. The first stage begins by forming  $f$  different ‘star’ graphs with the degree of each star centre vertex chosen uniformly at random in  $[ld, ud]$  (Figure 7.11). These disconnected components are then connected by adding  $f - 1$  edges at random, to form a tree (Figure 7.12). The set of connected vertices,  $V_T$ , in the tree will

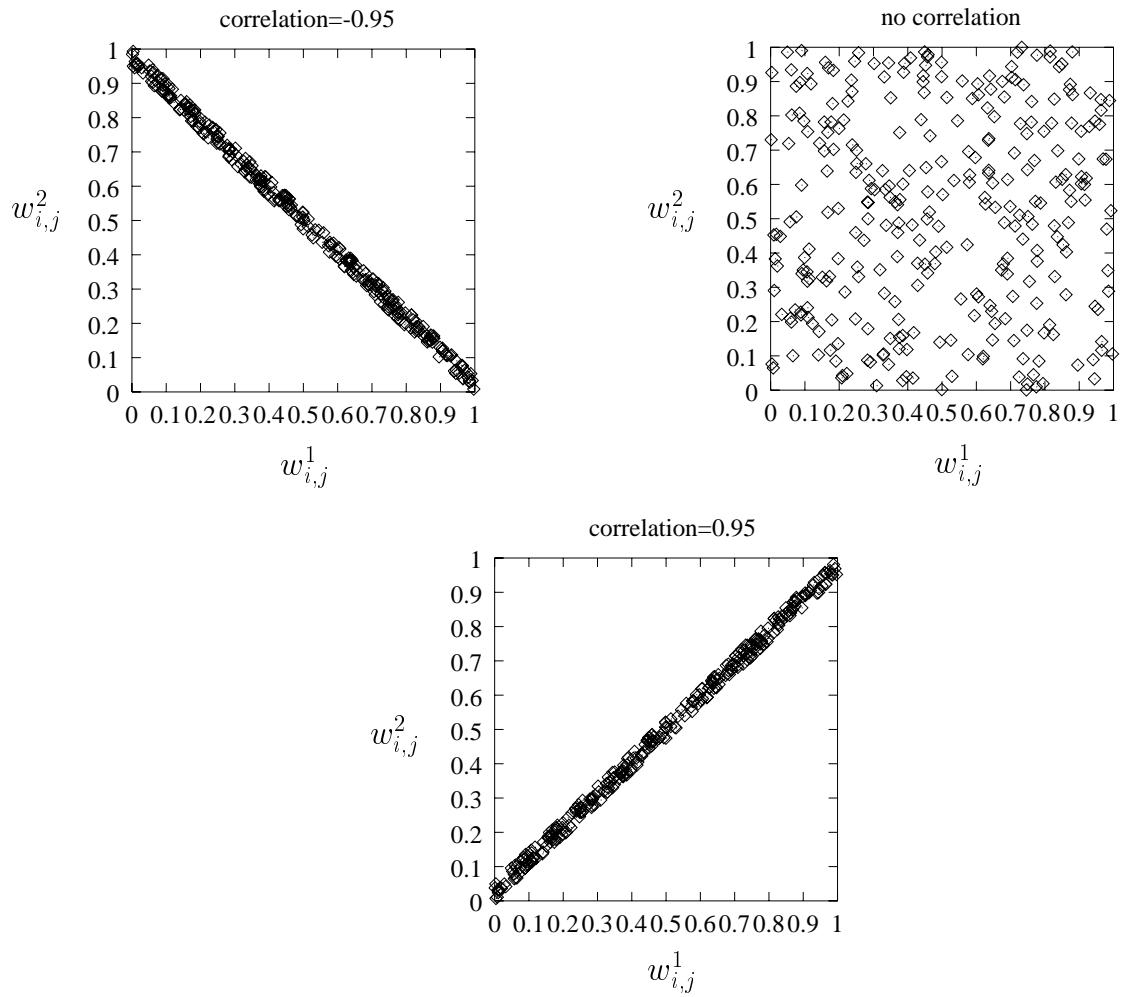


Figure 7.8: Plots showing the weight vectors assigned by the correlated-weights graph generator.

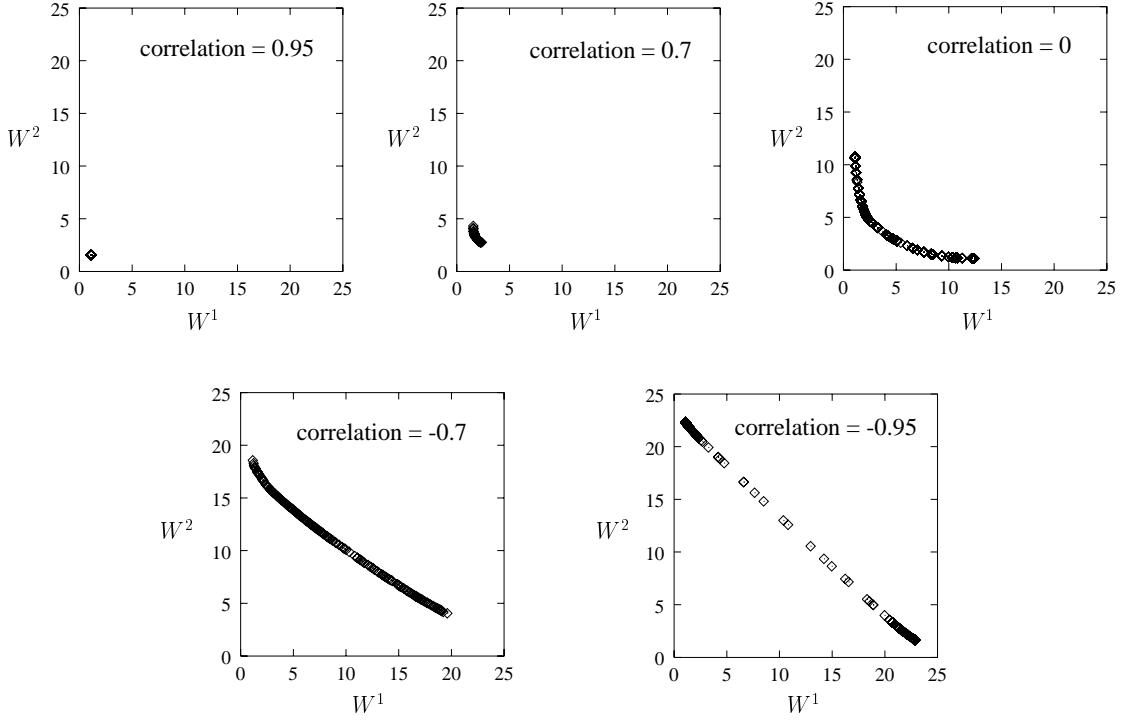


Figure 7.9: The effect of correlated weights on the shape of the Pareto front.

have fewer than  $|V|$  members, so that the tree is not spanning. To span the whole graph, additional vertices in  $V \setminus V_T$ , not in any of the stars, will be connected. However, first the edges in the (non-spanning) tree formed so far, are all assigned uniformly random weights in  $[0, \chi]$ . Next, all remaining unconnected vertices are connected to the tree by adding an edge between them and exactly one of the star centre vertices. The weight of these edges is assigned a uniformly random weight in  $[\phi, \chi]$  (Figure 7.13). Additional edges are then added between any pair of non-adjacent vertices, until the graph reaches the required density of connectedness (Figure 7.15). The weights assigned to these additional edges are uniformly random in  $[\chi, \omega]$  for any pair of vertices where both are members of  $V_T$ , and uniformly random in  $[\psi, \omega]$  for all other vertex pairs. If the weight parameters are set so that  $0 < \phi < \chi \ll \psi < \omega$  then the resulting graph will be a misleading graph, that is the graph's structure will successfully mislead algorithms that favour choosing low-weight edges. This can be understood by first noticing that the edges incident to the vertices in  $V \setminus V_T$  have two different ranges of values. Those edges whose other incident vertex is a star-centre have a low weight, whereas all others have a high weight. However, often a greedy-style algorithm will be forced to choose the higher weighted edge to connect these vertices because in earlier choices it will favour the edges that are incident to a star centre vertex (because these have the lowest range of weights

---

**Algorithm: *Gen\_correlated\_wts***

---

*Data:*  $\alpha \in [-1, 1]$  is the correlation, provided by the user  
 $\beta$  is the offset, calculated from  $\alpha$   
 $\gamma$  is the variation, calculated from  $\alpha$  and  $\beta$   
 $\mathcal{U}(min, max)$  is a uniformly distributed random deviate  $\in [min, max)$

---

```

if ( $\alpha \geq 0$ )
   $\beta \leftarrow 1/2(1 - \alpha)$ 
   $\gamma \leftarrow \beta$ 
else
   $\beta \leftarrow 1/2(1 + \alpha)$ 
   $\gamma \leftarrow \beta - \alpha$ 
foreach edge  $(i, j) \in E$ 
   $w_{i,j}^1 \leftarrow \mathcal{U}(0, 1)$ 
  foreach objective  $k \in 2..K$ 
     $w_{i,j}^k \leftarrow \alpha w_{i,j}^1 + \beta + \gamma \mathcal{U}(-1, 1)$ 

```

---

Figure 7.10: An algorithm for generating a graph with correlated weight vectors.

in the graph) thereby causing the star centre vertex to reach its maximum allowed degree, and so preventing the connection of it to one of the vertices in  $V \setminus V_T$ . Because  $\psi \gg \chi$  this will lead to a heavier graph, overall.

To make a multiobjective version of an M-graph, we use the M-graph procedure to set the weights of the first component of all the edge weight vectors. The others components of the edge weight vectors are then set using the correlation procedure outlined above. If a large positive correlation is used then the graph will be misleading in all of its components, and it will be difficult for a greedy approach to find a low-weight solution if the degree constraint is much lower than the parameter  $ud$ .

#### 7.5.4 Concave

The Concave problem generator can only be used to make 2-objective problems at present. It works by setting the edge weights of three ‘special’ vertices (labelled 1, 2, and 3) in such a way that a large concave region in the Pareto front will result. If we restrict all edge

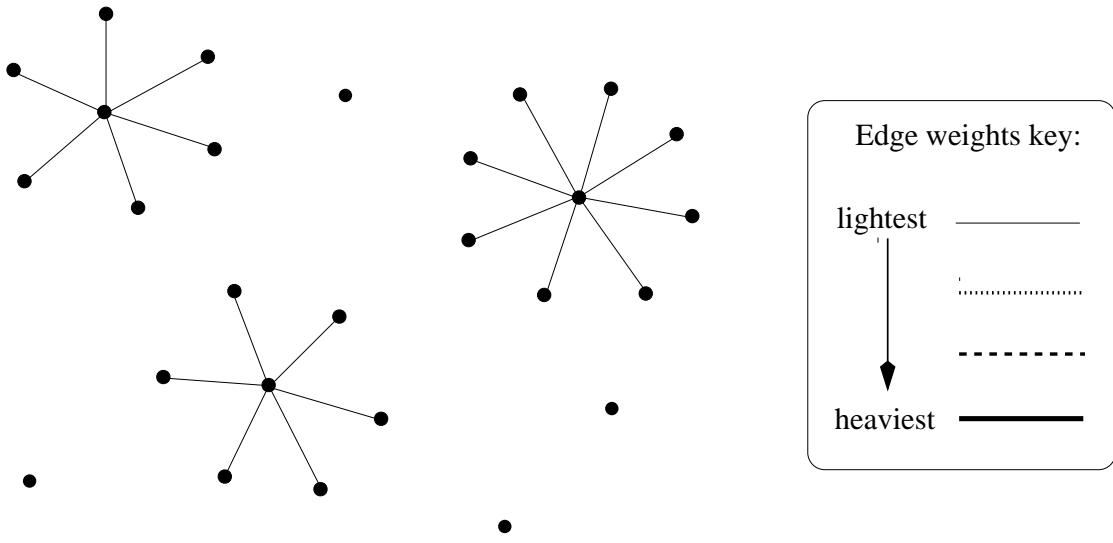


Figure 7.11: Constructing an  $M$  graph. Step 1: Form  $f$  star graphs, using the lightest edges, leaving some vertices unconnected.

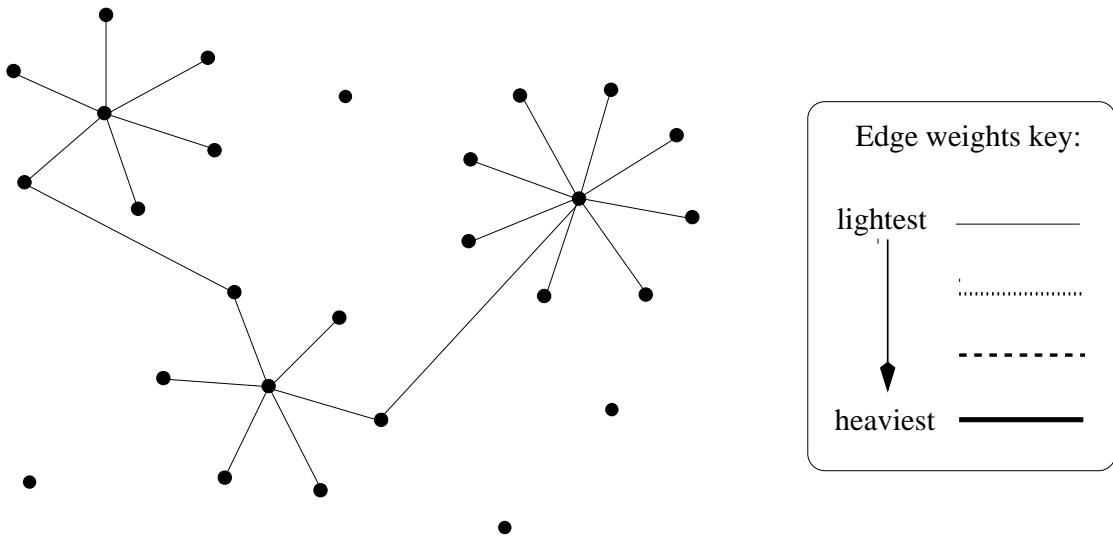


Figure 7.12: Constructing an  $M$  graph. Step 2: Add  $f - 1$  of the lightest edges between the stars, to form a tree.

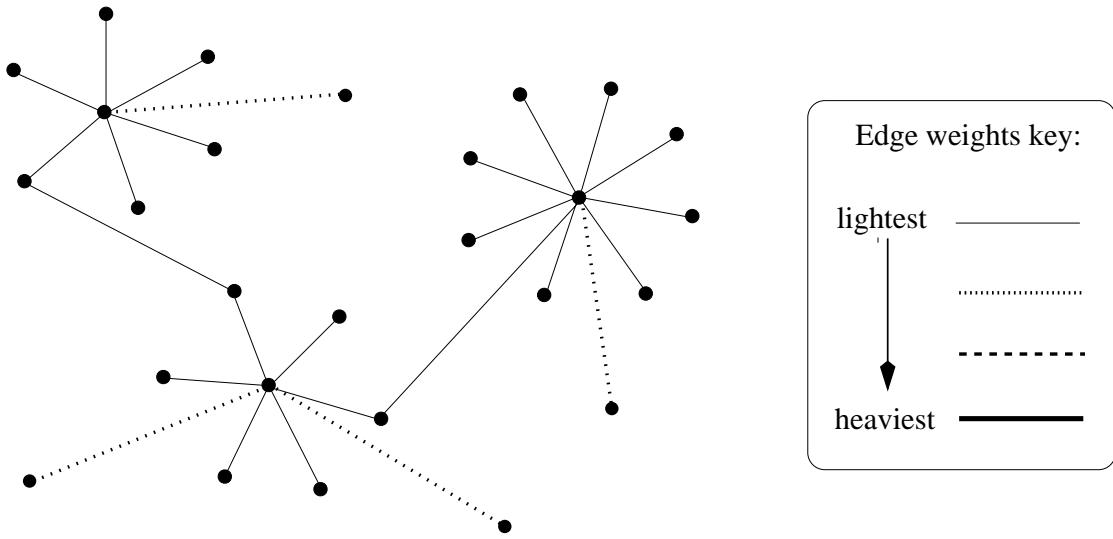


Figure 7.13: Constructing an  $M$  graph. Step 3: Connect up the unconnected vertices using the second lightest edges. The resulting graph forms the “underlying” MST.

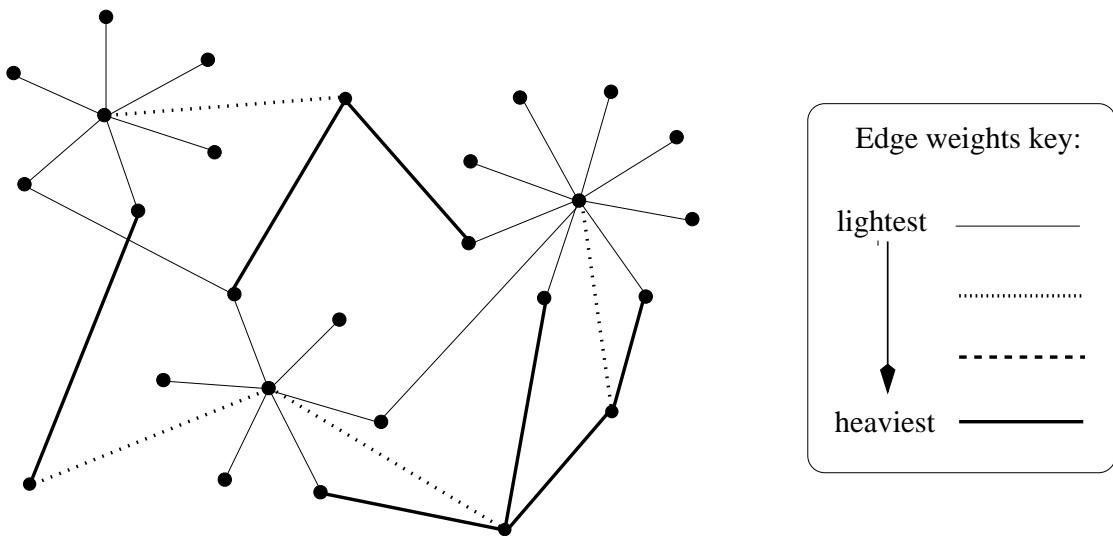


Figure 7.14: Constructing an  $M$  graph. Step 4: Add in edges of the heaviest weight, incident to the vertices *not* in the stars formed in step 1.

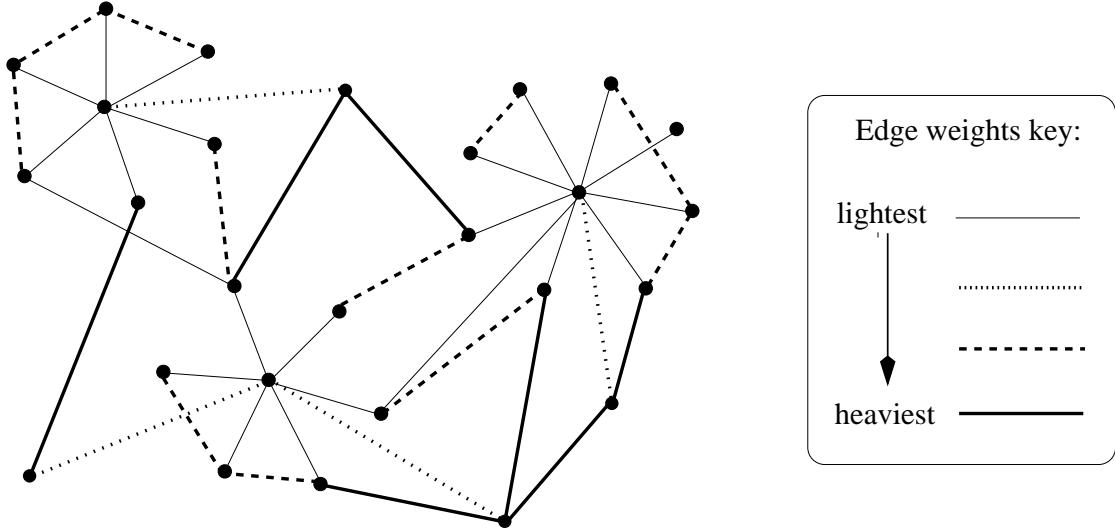


Figure 7.15: Constructing an  $M$  graph. Step 5: Add in further edges of medium weight until the graph reaches the required density of connectedness.

weights to lie in  $[0,1]$ , then the weights that can be used are the following:  $\mathbf{W}_{0,1} = (\zeta, \zeta)$ ,  $\mathbf{W}_{0,2} = (0, 1 - \zeta)$ , and  $\mathbf{W}_{1,2} = (1 - \zeta, 0)$ . All other edges are  $\mathbf{W}_{i,j} = (\mathcal{U}(\zeta, \eta), \mathcal{U}(\zeta, \eta))$  for  $i, j > 3$  and  $\mathbf{W}_{i,j} = (\mathcal{U}(1 - \zeta, 1), \mathcal{U}(1 - \zeta, 1))$  if  $i \oplus j \leq 3$ , with  $i, j \in V$ ,  $\zeta$  a small positive value of the order of  $1/|V|$ ,  $\zeta < \eta \ll 1 - \zeta$ , and  $\mathcal{U}(\min, \max)$  giving a uniformly random deviate in  $[\min, \max]$ .

An example of (an approximation of) the Pareto front of a concave graph problem is given in Figure 7.16. The graph has 25 vertices and the values of the parameters for generating it were:  $\zeta = 0.05$  and  $\eta = 0.2$ .

## 7.6 Problem instances

### 7.6.1 Simple random weight instances

Following [ZG99], we generate bi-objective MST instances using complete graphs of size 10, 20, 30, 40, and 50 vertices with uniformly randomly distributed edge weights  $w_{i,j}^1 \in [10, 100]$ ,  $w_{i,j}^2 \in [10, 50]$ ,  $\forall (i, j) \in E$ .

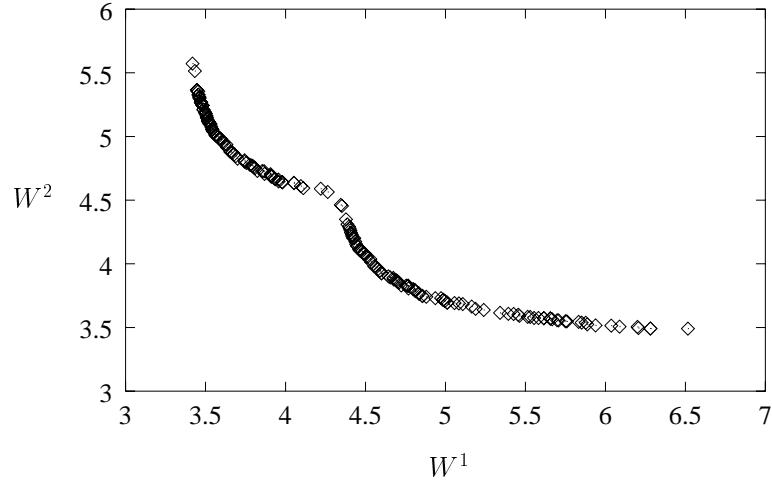


Figure 7.16: A plot showing nondominated points found by M-PAES on a run of a 25 vertex concave problem.

| Vertices | Instance type |   |     |      |
|----------|---------------|---|-----|------|
|          | AC            | C | M-C | Conc |
| 10       | 3             | 3 | 3,5 | 3    |
| 25       | 3             | 3 | 3,5 | 3    |
| 50       | 3             | 3 | 3,5 | 3    |

Table 7.1: The fifteen benchmark problem instances. Entries in the table are the degree constraints for each instance.

### 7.6.2 Benchmark problem instances

Three graphs for each of the instance types: correlated, anti-correlated, M-correlated, and concave were generated; one each at sizes of 10, 25, and 50 vertices, giving 12 graphs in all, from which 15 instances are created by setting degree constraints of 3 on all of the graphs, and an additional, lighter degree constraint of 5 on the three M-correlated graphs.

The correlations for the anti-correlated graphs, 10vAC, 25vAC, and 50vAC were set at -0.7. For the correlated graphs, 10vC, 25vC, and 50vC, the correlation was set at 0.7. For the M-correlated graphs the correlation was also set at 0.7, and the other parameters were  $f = 1, 2, 5$ ,  $ld = 6, 6, 7$ ,  $ud = 8, 10, 9$ , for the 10vM-C, 25vM-C, and 50vM-C, respectively. There is no correlation between the edge weight components in the concave graphs, and the other parameters used for generating these graphs were  $\zeta = 0.1, 0.05, 0.03$  and  $\eta = 0.25, 0.2, 0.125$  for the 10vConc, 25vConc and 50vConc graphs respectively.

## 7.7 Experiments

The experiments described in this section are in two parts. The first part concerns preliminary experiments on the simple random weight instances, designed to put the results of the second set of experiments in context of previous published work on the mc-MST problem. The second part uses the benchmark problem instances and compares the performance of PAES, M-PAES and a multiobjective EA, AESSEA, which is based on PAES.

In previous work on the mc-MST problem by Zhou and Gen [ZG99], the Prüfer encoding was used in a MOEA to give results on small random instances of the problem. The authors presented their results in terms of the number of true Pareto optimal solutions found, compared with the number in the true Pareto front, as determined using their enumeration algorithm. The reported results were surprisingly good, particularly as previous results using a Prüfer encoding on the  $d$ -MST have not been impressive [GJRR01]. Of course, the proof we have given in Appendix D regarding their enumeration algorithm indicates that the results reported may not be reliable. However, our preliminary experiments are aimed to test the results reported by Zhou and Gen so that our later results can be placed into context. To do this testing, we compare spanning trees found using a Prüfer encoding with those found using our own enumeration algorithm on the 10-vertex random weight graphs. We also compare the Prüfer encoding with our proposed direct encoding employing RPM initialization, described above. To make a comparison between the two encodings we test them in the same EA. The EA we use is called AESSEA and it is just a population-based variant of PAES. AESSEA, described in detail below, allows us to use crossover, as in the experiments carried out by Zhou and Gen. Finally, we compare both EAs with our iterated constructive approach, mc-Prim, on all the problem instances to judge the quality of the EA results on these simple problems.

In the second and main part of our experiments, the 15 benchmark problem instances are tackled. Once again, the performance of the evolutionary methods are compared with an enumeration algorithm (10-vertex instances only) and with our constructive iterative approach, mcd-Prim, to verify the efficacy of using EA methods on different instances of this problem. M-PAES and AESSEA are compared using the same encoding method — the direct/RPM encoding. These results are also compared with the baseline algorithm PAES which just uses the RPM encoding and operators. The results generated represent the first set of benchmark results for mc-MST problem instances with a degree constraint and with correlated and deceptive weights.

---

**Algorithm: Archived Elitist Steady-State EA**

---

*Data:*

$P$  is the population  
 $N$  is the archive of nondominated spanning trees  
 $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}$ , and  $\mathbf{x}$  are solution vectors

---

```
 $N \leftarrow \emptyset$                                 /* Initialization */  
foreach ( $\mathbf{x} \in P$ )  
    init( $\mathbf{x}$ )  
    evaluate( $\mathbf{x}$ )  
    archive( $\mathbf{x}$ )  
 $i \leftarrow 0$   
while ( $i < num\_evals$ )      /* Main Loop */  
    if rand()  $< p_c$   
         $\mathbf{a} \leftarrow \text{rand\_mem}(P)$   
         $\mathbf{b} \leftarrow \text{rand\_mem}(P)$   
         $\mathbf{c} \leftarrow \text{crossover}(\mathbf{a}, \mathbf{b})$   
    else  
         $\mathbf{a} \leftarrow \text{rand\_mem}(P)$   
         $\mathbf{c} \leftarrow \mathbf{a}$   
         $\mathbf{c} \leftarrow \text{mutate}(\mathbf{c})$   
        evaluate( $\mathbf{c}$ )  
        if ( $\mathbf{c} \prec \mathbf{a}$ )  
            archive( $\mathbf{c}$ )  
             $P \leftarrow P \cup \{\mathbf{c}\} \setminus \{\mathbf{a}\}$   
        else if ( $(\exists \mathbf{j} \in N \mid \mathbf{j} \prec \mathbf{a}) \wedge (\exists \mathbf{k} \in N \mid \mathbf{k} \prec \mathbf{c})$ )  
             $P \leftarrow P \cup \{\mathbf{c}\} \setminus \{\mathbf{a}\}$   
        else if ( $\neg \exists \mathbf{j} \in N \mid \mathbf{j} \prec \mathbf{c}$ )  
            if ( $(g\_pop(\mathbf{c}) < g\_pop(\mathbf{a})) \vee (\exists \mathbf{j} \in N \mid \mathbf{c} \prec \mathbf{j})$ )  
                archive( $\mathbf{c}$ )  
                 $P \leftarrow P \cup \{\mathbf{c}\} \setminus \{\mathbf{a}\}$   
 $i \leftarrow i + 1$   
return ( $N$ )                                /* Termination */
```

---

Figure 7.17: AESSEA.

### 7.7.1 AESSEA

An evolutionary algorithm, AESSEA, based closely on procedures already defined for the Pareto archived evolution strategy (PAES) is presented in Figure 7.17. AESSEA is a steady-state EA, that is, only one new solution is evaluated per ‘generation’. It keeps a set of nondominated solutions in an archive, and uses this set of solutions to estimate the quality of newly generated solutions. The algorithm is elitist in the sense that parents and offspring compete, but the overall selection pressure of the algorithm is not too strong since selection for mating is purely random, and offspring only replace one of their parents, rather than the weakest member of the population. Some preliminary testing of this algorithm and comparison with PESA [CK00] suggest that it is both an effective and computationally efficient, multiobjective EA.

In AESSEA, the function `rand()` returns a uniformly distributed deviate in  $[0, 1)$ , and the function `rand_mem( $P$ )` returns with uniform probability a member of the current population,  $P$ . The function `archive( $c$ )` updates the nondominated solutions archive  $N$  with  $c$ , that is  $c$  is added to  $N$  if it is nondominated with respect to the other solutions in  $N$ , and if  $c$  dominates any members of  $N$  the dominated members are removed. The archive  $N$  has a finite capacity  $arcsize$  and in the case where adding  $c$  to  $N$  would cause  $|N| > arcsize$ , a solution in the most crowded region in  $N$  is removed. This archiving strategy is exactly the same as in the PAES algorithm. The function `g_pop( $x$ )` returns the number of solutions in the same grid location in objective space as the solution  $x$ .

The functions, `init( $x$ )`, `evaluate( $x$ )`, `crossover( $a, b$ )`, and `mutate( $c$ )`, are to be defined depending on the application of the algorithm. In the following experiments, AESSEA+Direct/RPM uses the RPM initialization and the edge crossover and edge mutation described in Section 7.4.3 and AESSEA+Prüfer uses the operators and encoding described in Section 7.4.2.

### 7.7.2 Random weight instances

Three different instances at size 10 are generated, and on these instances the complete space of solutions is enumerated so that we obtain all Pareto optimal solutions. For these problems we then present results in terms of the proportion of the efficient set that is discovered by the genetic algorithms, over 30 independent runs. This is a similar method to that used by Zhou and Gen for verifying the effectiveness of their GA. However, we enumerate the set of efficient spanning trees using a straightforward technique where every possible spanning tree is generated using a Prüfer code, and then those that are dominated are discarded. Although better techniques exist [RASG98], we have been unable to obtain source code for

| Parameter    | AESSEA+Prüfer | AESSEA+D/RPM |
|--------------|---------------|--------------|
| $ P $        | 200           | 200          |
| $arcsize$    | 100           | 100          |
| $p_c$        | 0.2           | 0.2          |
| $p_m$        | $1/L$         | $1/L$        |
| $num\_evals$ | 100000        | 20000        |
| grid squares | 1024          | 1024         |

Table 7.2: Parameter settings for the two AESSEA algorithms.

these methods to date. Nonetheless, our enumeration does correctly guarantee to find all and only nondominated spanning trees, in contrast to the method used by Zhou and Gen (see Appendix D).

For the other sizes of network, just one instance at each size is used. Because of the larger size of these problem instances, we cannot use our enumeration method. Therefore, we instead compute a subset of the supported efficient solutions using our mc-Prim algorithm for 1001 different  $\lambda$  vectors from  $(0, 1), (0.001, 0.999), \dots, (0.999, 0.001), (1, 0)$ .

On the small problems, since we know the complete true Pareto front, we evaluate algorithm results by looking at the percentage of the true Pareto front found by the algorithm. We in fact look at the percentages of both the full Pareto front and its complete supported subset. For the larger problems, however, only a subset of the true supported Pareto front is available to us. In these cases we look at the size of the total discovered region. This is simply the area (since our problems are two-dimensional) contained by the approximation to the Pareto front which the algorithm discovers.

The parameters used in the AESSEA algorithms are given in Table 7.2. The mutation rate is quoted as  $1/L$ , where  $L$  is the length of the chromosome. For the Prüfer encoding  $L = |V| - 2$ . For the AESSEA+Direct/RPM, this parameter is not actually set but it refers to the fact that using Raidl's edge encoding and heuristic edge mutation, exactly one edge in the tree is changed per mutation, which is equivalent to a mutation rate of  $1/L$  on a direct edge encoding.

### 7.7.3 Benchmark instances with degree constraints

The parameters used for each of PAES, AESSEA, and M-PAES are given in Table 7.3. Each of the algorithms was given 30 independent runs on each of the 15 problems and the nondominated archive returned by each algorithm from each run was stored for statistical

| Parameter                               | PAES        | AESSEA         | M-PAES             |
|---|-------------|----------------|--------------------|
| population size, $ P $                  | 1           | 200            | 50                 |
| archive size, $arcsize$                 | 200         | 200            | $G = 200, H = 200$ |
| initialization method                   | RPM         | RPM            | RPM                |
| mutation type                           | mutate_RPM  | edge-mutation  | edge-mutation      |
| crossover type                          | —           | edge-crossover | edge-crossover     |
| # of function evaluations, $num\_evals$ | 20k/50k/50k | 20k/50k/50k    | 20k/50k/50k        |
| # of grid regions                       | 1024        | 1024           | 1024               |

Table 7.3: Parameter settings for the three algorithms; in addition  $l\_opt = 50$ ,  $l\_fails = 20$ ,  $cr\_trials = 10$  for M-PAES (see Section 6.5 for the meaning of these parameters). The three figures for number of evaluations relate to the three different problem sizes, 10, 25, and 50 vertices, respectively.

analysis, and comparison with the other non-EA approaches. For each problem, mcd-Prim was also run 5 times with a different start vertex, and the combined nondominated solution set was stored. On the ten-vertex problems we also use an enumerative procedure to give us the entire true Pareto front for comparison.

## 7.8 Results

### 7.8.1 Random weight instances

Table 7.5 shows the results on the 10-node problems. On the small problems, AESSEA+Direct/RPM finds all optima on all individual runs, and in only 20000 evaluations. The AESSEA+Prüfer method performs considerably worse, often finding no true Pareto optima at all. On the larger problems, with results summarised in Table 7.6, the discovered Pareto front of AESSEA+Prüfer is considerably smaller than that from either of the other methods.

Such findings are illustrated in Figure 7.18, clearly indicating how well AESSEA+Direct/RPM approximates the true Pareto front in comparison to AESSEA+Prüfer. As we note from similar figures (not displayed here) for the other problems, this difference is increasingly marked as problem size increases.

The runtime figures given in Table 7.4 seem to indicate that both EAs scale well with increasing problem size.

| Problem | AESSEA+Prüfer | AESSEA+D/RPM |
|---------|---------------|--------------|
| 20v1    | 19.7s         | 14.2s        |
| 50v1    | 49.6s         | 28.0s        |

Table 7.4: Run-times for the two algorithms on two different-sized problems. The algorithms were run on a Sun SPARC Ultra 5 300MHz with 256MB RAM. The timings are the mean of 10 runs. On each run, AESSEA+Prüfer performed 100000 function evaluations, and AESSEA+Direct/RPM performed 20000, as in the experiments reported.

| Problem<br>instance | AESSEA+Prüfer |         |             |           |                    | AESSEA+Direct/RPM |             |           |                    |                |
|---------------------|---------------|---------|-------------|-----------|--------------------|-------------------|-------------|-----------|--------------------|----------------|
|                     | $ Z_s^* $     | $ Z^* $ | $\% Z_s^* $ | $\% Z^* $ | $\overline{ Z^a }$ | $ Z^a  \sigma$    | $\% Z_s^* $ | $\% Z^* $ | $\overline{ Z^a }$ | $ Z^a  \sigma$ |
| 10v1                | 10            | 42      | 80          | 76        | 2.47               | 1.33              | 100         | 100       | 42.00              | 0.00           |
| 10v2                | 12            | 28      | 67          | 71        | 0.50               | 0.78              | 100         | 100       | 28.00              | 0.00           |
| 10v3                | 12            | 25      | 83          | 84        | 0.57               | 0.82              | 100         | 100       | 25.00              | 0.00           |

Table 7.5: Results on three 10-vertex mc-MST instances using the two different EAs.  $|Z_s^*|$  and  $|Z^*|$ , give the number of supported efficient points and the total number of efficient points on the *true* Pareto front, respectively, for each problem instance.  $\%|Z_s^*|$ , give the percentage of the supported efficient points that were found over the 30 runs *combined*, for each algorithm. Similarly,  $\%|Z^*|$  gives the percentage of the total number of efficient points that were found over the 30 runs *combined*, for each algorithm.  $\overline{|Z^a|}$ , and  $|Z^a| \sigma$ , give respectively the mean and standard deviation of the number of efficient points in the approximation sets generated by the algorithms *per* algorithm run, for each algorithm.

| Problem<br>instance | Upper limit of<br>bounding box |             | Size of the total discovered region |                   |        |
|---------------------|--------------------------------|-------------|-------------------------------------|-------------------|--------|
|                     | $ Z_{s,Pr}^* $                 | mc-Prim     | AESSEA+Prüfer                       | AESSEA+Direct/RPM |        |
| 20v1                | 25                             | 832 x 574   | 0.2949                              | 0.1488            | 0.2986 |
| 30v1                | 54                             | 1280 x 838  | 0.3780                              | 0.1249            | 0.3792 |
| 40v1                | 63                             | 1882 x 1181 | 0.4200                              | 0.1150            | 0.4196 |
| 50v1                | 85                             | 2425 x 1494 | 0.4473                              | 0.0943            | 0.4451 |

Table 7.6: The size of the total discovered region of the AESSEA algorithms and mc-Prim on four mc-MST problem instances of increasing size. The column,  $|Z_{s,Pr}^*|$ , gives the number of supported efficient solutions found by mc-Prim, using 1001 different  $\lambda$  vectors. The upper limit of the bounding box is determined, in each dimension, by taking the maximum value (over all algorithms) of any point on the combined Pareto front of each algorithm. The lower limit is taken to be zero in each dimension.

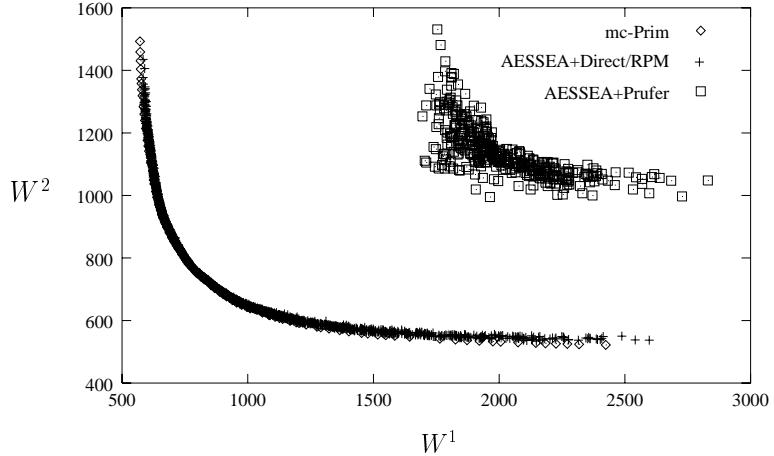


Figure 7.18: Combined discovered points found by the two versions of AESSEA and mc-Prim on a 50-vertex problem.

### 7.8.2 Benchmark test problem instances

The results of our first statistical analysis method are given in Table 7.7. The best results are shown in bold. It is clear from the table that M-PAES and AESSEA both using a direct coding and employing crossover do favourably compared to PAES using only the RPM decoder encoding, and this superiority is emphasized further as problem size increases. M-PAES and AESSEA using the same encoding are well-matched on most problems but M-PAES is clearly best overall on the set of problems considered. It is particularly strong on the largest of the M-correlated graphs, and the large concave graph problem.

The results of the second statistical analysis method are given in Table 7.8. The results paint a similar picture overall, although there are some interesting points to notice as well. First, mcd-Prim performs very well compared to the evolutionary algorithms on the correlated and anti-correlated problems, and is considerably faster (but see Figure 7.19 for further help visualizing the Pareto fronts discovered). However, on the M-correlated graphs it clearly struggles. This is as expected because on these problems the degree constraint has a real effect on the difficulty of finding good solutions. We can see that the EAs are managing to cope with this; observe the size of the region discovered by them compared to the enumeration method on the 10 vertex M-correlated problem for both  $d=3$  and  $d=5^2$ . This is further shown in a plot given in Figure 7.20. Finally, on the (larger) concave problems, it appears that mcd-Prim does better than the EAs but in fact is unable to find any solutions in the concave

<sup>2</sup>On 10vM-C-d3 it appears that PAES has a larger than possible median size. However, this is due to the sampling error of only using 500 sampling lines in measuring the attainment surfaces of this algorithm.

| Problem   | PAES           | AESSEA             | M-PAES             |
|-----------|----------------|--------------------|--------------------|
| 10vAC     | 32.7 (0)       | 97.6 (0)           | <b>100 (2.4)</b>   |
| 25vAC     | 0 (0)          | 13.3 (7.3)         | <b>92.7 (86.7)</b> |
| 50vAC     | 0 (0)          | 17.4 (15.7)        | <b>84.3 (82.6)</b> |
| 10vC      | 99.5 (0)       | <b>100 (0)</b>     | <b>100 (0)</b>     |
| 25vC      | 45.1 (0)       | <b>99.6 (10.8)</b> | 89.2 (0)           |
| 50vC      | 0 (0)          | 3.8 (0)            | <b>100 (96.2)</b>  |
| 10vM-C-d3 | 95.6 (0)       | <b>100 (0)</b>     | <b>100 (0)</b>     |
| 25vM-C-d3 | 0 (0)          | <b>99.3 (14.5)</b> | 85.5 (0.7)         |
| 50vM-C-d3 | 0 (0)          | 17.3 (2.5)         | <b>97.5 (82.7)</b> |
| 10vM-C-d5 | <b>100 (0)</b> | <b>100 (0)</b>     | <b>100 (0)</b>     |
| 25vM-C-d5 | 0 (0)          | <b>100 (3.8)</b>   | 96.2 (0)           |
| 50vM-C-d5 | 0 (0)          | 19.2 (0)           | <b>100 (81.8)</b>  |
| 10vConc   | 52.1 (0)       | <b>100 (0)</b>     | <b>100 (0)</b>     |
| 25vConc   | 9.2 (0.6)      | 68.5 (22.7)        | <b>74.6 (30.1)</b> |
| 50vConc   | 0 (0)          | 36 (3.7)           | <b>96.3 (64)</b>   |

Table 7.7: Unbeaten and (beats all) statistics for the three EAs on the full set of problems.

| Problem   | Total size |          | Median (Interquartile) size |                        |                        |
|-----------|------------|----------|-----------------------------|------------------------|------------------------|
|           | Enum       | mcd-Prim | PAES                        | AESSEA                 | M-PAES                 |
| 10vAC     | 21.3837    | 20.4108  | 21.2963 (0.2014)            | 21.3565 (0.0032)       | <b>21.357 (0.0044)</b> |
| 25vAC     |            | 246.256  | 239.248 (32.121)            | 245.175 (1.428)        | <b>245.5 (1.893)</b>   |
| 50vAC     |            | 1240.27  | 1040.02 (440.492)           | 1224.07 (7.6)          | <b>1225.67 (13.85)</b> |
| 10vC      | 36.0955    | 35.3919  | 35.7349 (0.0323)            | <b>35.7672 (0)</b>     | <b>35.7672 (0)</b>     |
| 25vC      |            | 363.233  | 362.409 (1.492)             | 363.329 (0.109)        | <b>363.386 (0.149)</b> |
| 50vC      |            | 1868.59  | 1853.01 (24)                | 1869.3 (3.52)          | <b>1873.29 (4.23)</b>  |
| 10vM-C-d3 | 26.8362    | 23.6895  | <b>26.8399 (0)</b>          | 26.8342 (0)            | 26.8342 (0)            |
| 25vM-C-d3 |            | 281.923  | 338.026 (7.135)             | 343.892 (0.55)         | <b>343.988 (0.005)</b> |
| 50vM-C-d3 |            | 1302.33  | 1454.86 (40.14)             | <b>1493.89 (3.17)</b>  | 1493.48 (8.9)          |
| 10vM-C-d5 | 40.0389    | 35.3428  | <b>40.0365 (0)</b>          | <b>40.0365 (0)</b>     | <b>40.0365 (0)</b>     |
| 25vM-C-d5 |            | 345.62   | 383.684 (2.801)             | <b>384.511 (0)</b>     | 384.439 (0)            |
| 50vM-C-d5 |            | 1496.8   | 1598.07 (26.44)             | 1615.49 (4.56)         | <b>1616.33 (7.79)</b>  |
| 10vConc   | 37.3255    | 37.0848  | 37.2362 (0.0061)            | <b>37.2367 (0)</b>     | <b>37.2367 (0)</b>     |
| 25vConc   |            | 334.694  | 332.901 (3.81)              | <b>334.644 (0.354)</b> | 334.491 (0.852)        |
| 50vConc   |            | 2122     | 2109.76 (19.39)             | <b>2118.39 (1.67)</b>  | 2114.7 (4.29)          |

Table 7.8: Size of the median and interquartile dominated regions for the different evolutionary algorithms, and the total combined size of the dominated region found using five runs of mcd-Prim. For the ten-vertex problems the true size of the dominated region is represented by the results of the Enumeration algorithm.

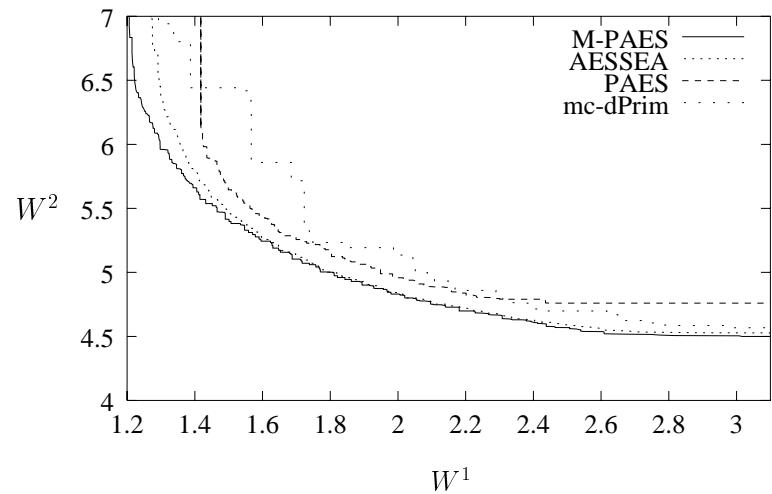


Figure 7.19: Median attainment surfaces on the 50 vertex correlated problem with a degree constraint of 3.

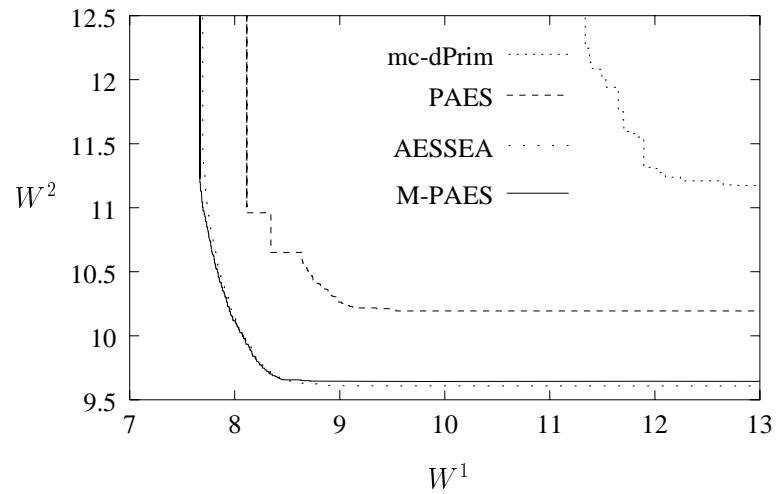


Figure 7.20: Median attainment surfaces on the 50 vertex M-correlated problem with a degree constraint of 3.

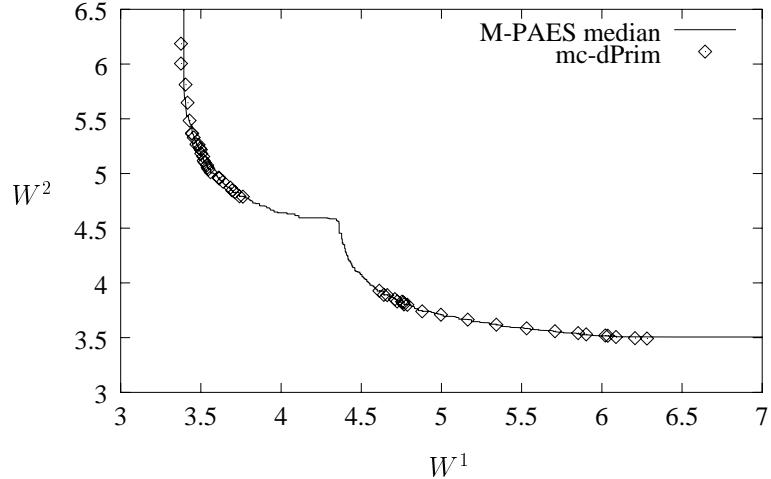


Figure 7.21: Nondominated points found from 5 runs of *mcd*-Prim, and the median attainment surface achieved by M-PAES. Note how M-PAES finds points in the concave region of the Pareto front.

region of the Pareto front. Its larger dominated region is due to it finding the very edge of the Pareto Front, which the EAs do not achieve on every run. A plot showing the median attainment surface for M-PAES, and the points found from 5 runs of *mcd*-Prim is given in Figure 7.21.

## 7.9 Conclusion

The results indicate that a direct encoding is more effective than a Prüfer-based encoding when using an EA to address the mc-MST. We should point out that our conclusions differ from those made by Zhou and Gen for their Prüfer GA, which seemed to show that their method was able to find high proportions of the true Pareto front on all sizes of problem, which is surprising in light of the results we find using the Prüfer encoding on large problems. We suggest the discrepancy arises from Zhou and Gen's use of an unreliable Pareto front enumeration method, as noted earlier. In addition to the results we report, reported theoretical notes provide additional evidence for direct encodings being a better choice than the Prüfer encoding for MST problems. For example, Ehrgott and Klamroth [EK97] noted that, in 50 random instances in which they generated a complete set of efficient spanning trees, the set was ergodic with respect to a single-edge exchange operator (although they prove that this will not always be true). A direct encoding such as Raidl's, which naturally allows for such operators (as indeed used here) seems thus well-designed for the problem, while the high non-locality of the Prüfer encoding seems ill-chosen in light of this. A further difficulty with

the Prüfer encoding is the fact that many real problems will have sparse graphs. That is, although the problem may still be very large in terms of allowed edges, most edges will not be allowable. This could be dealt with in Prim's algorithm, for example, by giving those edges massive weight (vectors), large enough to guarantee that they would never be chosen in tree construction. A far better approach, however, is simply not to include these edges in the edge array accessed by the algorithm. The latter natural and effective way to deal with sparse problems is trivial for both RPM and Raidl's direct encoding, but is unavailable to the Prüfer encoding. This is because it is impossible to control which edges are coded for by a Prüfer number, so the decoding process will typically yield infeasible trees (containing edges not members of the sparse graph).

It is also worth noting that mc-Prim's alone is capable of finding well-spread sets of true Pareto optima for the mc-MST much faster than an EA. However, mc-Prim's is incapable of finding unsupported Pareto optima, of which there may be many between any two neighbouring supported optima. For these reasons, mc-Prim's is able to display impressive results in Table 7.6, but will always fail to offer a fine-grained view of the tradeoff surface, such as would be desired in many applications. Also, note that although mc-Prim's is easily adaptable to degree-constrained problems, it certainly no longer guarantees to find true optima in those cases.

We suspect that many real-world problems, as well as having degree-constraints, are also quite sparse, and have a considerable number of unsupported solutions.

We have presented a number of graph generators that can produce a range of challenging graph types for the *mcd*-MST problem. We show that on some problems it may not be necessary to use an evolutionary algorithm or other metaheuristic technique for tackling these problems, because a simple, iterative approach — *mcd*-Prim — can provide very good results in a fraction of the time. However, we have also demonstrated that for certain problems with constraints that are difficult to meet, the evolutionary algorithms do exhibit superior performance. Furthermore, the evolutionary algorithms are able to find points in the non-supported regions of the Pareto front, as was clearly demonstrated using the concave graph generator. In real problems of interest to the telecommunications industry we suspect that the number and variety of constraints that must be met will necessitate the use of evolutionary algorithms similar to those investigated here.

We would also like to draw attention to the superiority of the memetic algorithm, M-PAES, on these *mcd*-MST problems. Although M-PAES has been shown to perform well on other problems, it seems that in this application, a local-search element is particularly useful. This may well be due to the observation made by Ehrgott and Klamroth [EK97] that from a

sample of 50 random instances of a random weight bi-objective MST problem, all of them were ergodic with respect to a single exchange operator (although they prove this will not always be true). In light of this, we predict that further advances in tackling these difficult constrained mc-MST problems will come from techniques that incorporate a strong local search element, as used in M-PAES.

# Chapter 8

## Conclusions

### 8.1 Summary

In this dissertation, we have argued that local-search methods may be a valuable, and somewhat overlooked, alternative to population-based approaches to general-purpose Pareto multiobjective optimization. We postulated that simple, baseline algorithms such as hillclimbers perform several useful roles in *single-objective* optimization: in verifying the expediency of more sophisticated approaches; in the development of new approaches, particularly hybrids; and in our theoretical understanding of problem types and search spaces, and how these relate to the efficacy of different optimization methods. From this perspective, we argued that developing a ‘Pareto hillclimber’ might similarly help to develop new *Pareto optimization* methods and theory. To support these ideas, we have devised, analysed, and tested several new Pareto optimization algorithms based on single-point local search. These algorithms, we argued, contribute to increasing the diversity of available methods for undertaking Pareto optimization, and also allow powerful hybrid approaches to be devised and exploited.

The first element of our argument concerned the status of local search methods, in the wider optimization community. We showed that local search methods such as hillclimbing, simulated annealing, and tabu search, are broadly recognized as valuable alternatives to evolutionary computation methods for searching ‘hard’ problem spaces, through a literature review presented in Chapter 2. In addition, we noted that local-search methods can sometimes be adapted to incorporate problem-specific heuristics, and partial evaluation of solutions, resulting in further improvements in efficacy and efficiency. Furthermore, memetic algorithms (MAs), which ‘improve’ solutions — often using local search — allow local search heuristics to be incorporated into a population-based recombinative search strategy. This gives these

hybrid approaches a significant advantage over standard genetic algorithms in many problem domains.

In the next element of the dissertation our attention turned to multiobjective optimization, and to the variety of approaches currently available. Chapter 3 reviewed the most popular and influential MOEAs, as well as some local-search based metaheuristics for multiobjective optimization. This survey indicated that MOEAs have converged somewhat towards two-parent recombinative algorithms using niching for diversity maintenance and Pareto ranking selection. Similarly, multiobjective local search algorithms, from the operations research community, also lack real diversity in some respects. Most of these local searchers use randomly selected utility functions, and a scalar acceptance function. We argued that the use of utility functions may not be as efficient as Pareto selection, in terms of searching for diverse solutions with the fewest number of function evaluations, because at each step only one ‘direction’ is favoured, and this direction is not dependent on any of the solutions previously found. On the other hand, MOEAs that use Pareto selection, may not be the most efficient methods, either, since local search may be more suited to many problems, particularly when good local-search heuristics are known, or can be found.

We also noted that while local-search methods and hybrid approaches like MAs have received significant attention in much evolutionary computation literature, there has been little interest in these techniques for multiobjective optimization, until very recently. Although simulated annealing and tabu search approaches for multiobjective optimization have been previously proposed, they have not been tested against any of the popular multiobjective EAs. There has been no real interaction (competitive or cooperative) between users of MOEAs and users of other multiobjective metaheuristics. This means that there is a lack of any understanding, at present, of just how good the current breed of MOEAs is, on any given problem. As yet, there is no real culture of baselining multiobjective EAs against likely competitors. This is a problem because there is little impetus or scope for building hybrid approaches, even though hybrid approaches are very successful in single-objective search. There is also a lack of understanding of how the different landscape of a multiobjective problem affects search, in general. Without a culture of competition and diversity of approaches, it is difficult to see how we will come to understand how multiobjective landscapes and different search methods interact. From these considerations, we proposed that a basic local-search Pareto optimization algorithm, based on Pareto selection, should be devised.

In Chapter 4, the local search Pareto optimization algorithm, (1+1)-PAES was developed according to the requirements specified in Chapter 3. Several difficulties with satisfying these requirements were identified, particularly with the design of the acceptance function,

and with methods for maintaining a diverse, but bounded set of solutions. However, each of these was overcome in the final (1+1)-PAES algorithm we presented. Its acceptance function uses Pareto comparisons, making use of an archive of previously found solutions to aid in comparing the current and mutant solutions. The archive also allows a diverse, but bounded set of solutions to be returned at the end of a run of the algorithm. Various strategies for updating the archive were considered and we proposed a technique — adaptive grid archiving — which has low computational overhead, and requires no niche parameter or other measurements of the objective space to be made. As predicted, the development of PAES allowed us to devise other local-search methods based on Pareto comparisons. Three other variants of PAES were described: multi-start, simulated annealing, and tabu-search algorithms.

Chapter 5 presented two empirical studies of the performance of PAES versus other MOEAs. The first one showed that PAES is very competitive with the niched Pareto GA (NPGA) and the nondominated sorting GA (NSGA), and has lower computational overhead than these algorithms, as well as fewer parameters to set. The second study used a suite of test functions, each with a specific feature that might cause difficulty for a MOEA. PAES performs surprisingly well on this test set, too. It is only outperformed by SPEA on two of the six problems: a deceptive trap function problem, and a very highly multimodal problem. On the latter, the performance of PAES was found to improve with higher mutation rates, and allowing this ‘tuning’ to be done, PAES gives better results than the published SPEA results for that problem. On the deceptive problem, one-point crossover is a big advantage, and it is clear that PAES cannot compete with SPEA on this problem. Overall, however, the results showed that PAES is a capable baseline approach, able to compete with many previous population-based MOEAs. It achieves this with little need for tuning, and low computational overhead.

Memetic algorithms are generally used with problem-specific local improvement operators that take a solution to a local optimum. However, in Chapter 6, we decided to design an MA that we could first test using only standard operators, to allow comparison with other MOEAs, initially. The MA proposed uses the basic (1+1)-PAES algorithm as a subroutine for performing local improvement of solutions. Because PAES does not stop at a local optimum, however, we had to experiment with stopping criteria for the PAES procedure. With the recombination phase of the MA, we experimented with using restricted mating and elitist selection. We found the latter useful but the former not, in some preliminary experiments. The M-PAES algorithm presented combines local search and recombinative phases, uses the diversity maintenance and archiving techniques developed for PAES, and Pareto selection throughout.

M-PAES was tested using a variety of knapsack problems. Results showed that it was not very robust to parameter settings on these problems but that it outperformed SPEA, using the same operators and number of evaluations. It also outperformed (1+1)-PAES. Further testing of M-PAES on a telecommunications application, the ADDMP, revealed that M-PAES and SPEA were both outperformed by PAES, emphasizing the fact that recombination may not be useful on all multiobjective problems, and illustrating once more the usefulness of the simple PAES algorithm.

In Chapter 7, constrained and multiobjective versions of the classic minimum spanning tree problem were considered. A combination of direct and decoder representations were developed to tackle these problems, together with specialized mutation and recombination operators. A set of benchmark problem generators for producing problems with different features, and numbers of objectives, was also described. A set of difficult instances was generated, and low-weight spanning tree solutions to these were evolved using PAES, M-PAES and a steady state EA based on PAES, called AESSEA. These trees were compared to those generated by a constructive approximation approach, and for some problems, a complete enumeration of the search space. The results for M-PAES were generally the best, particularly on the problems with the most difficult constraints. The success of M-PAES was attributed to its repeated use of the mutation operator to improve solutions effectively.

## 8.2 Contributions

Local search is a fundamental technique in optimization upon which hillclimbing techniques and more sophisticated approaches are based. The development of a simple and effective local search algorithm for Pareto optimization — PAES — is thus a significant contribution to research in metaheuristic search and optimization. The contribution can be judged in several ways. First, we should consider the usage of PAES in this thesis and by others, as a baseline approach. Does PAES fulfil the requirements of a good baseline method, and what has been learnt from comparisons between PAES and other methods? Second, we may ask whether PAES performs well itself, as a multiobjective technique in its own right. To do this, we can once again review our results and those of others who have used PAES. Third we can ask whether PAES has shed any new light on the nature of multiobjective search spaces. Finally, we can judge PAES as a means of developing new techniques, thereby increasing the diversity of current approaches, and allowing other heuristics to be incorporated into multiobjective evolution.

In addition to contributions relating directly to the PAES algorithm and its variants, contri-

butions have also been made in in several other areas. These include: the M-PAES algorithm, the use of new metrics for assessing and comparing the performance of MOEAs, a critical analysis of previous techniques for this, new test problem generators and benchmark instances, and new benchmark results. In the following sections these contributions are evaluated.

### 8.2.1 The contribution of PAES

When verifying the quality of a new algorithm or approach, a simpler algorithm giving a baseline set of results can be used for comparison. Baselinining results in this way shows that the new approach is, at least, sensible, and better than the most straightforward of plausible methods. We argued that a ‘baseline’ algorithm that provides these baseline results is useful only if it has the following attributes:

- Easy to use with little or no setting or tuning of parameters necessary.
- Flexible enough to be applied to a wide range of applications.
- Provides a ‘good’ level of search performance i.e. not worse than random search.
- Simplicity in concept and design.

Thus we can ask: “Is (1+1)-PAES a good baseline multiobjective approach?” In answer, we can argue that is easy to use: it has only 3 parameters, and these just relate to the stopping criterion, the sizing of the archive, and the adaptive grid. None of them requires tuning. The mutation rate is not really a parameter of the algorithm as such, but of the neighbourhood operator. For best performance, this does need tuning but this is the same in a single-objective hillclimber. We can also argue that PAES is easy to use because a number of researchers have taken the C code and used the algorithm without need for further instruction. It has also been used as a baseline algorithm for testing in two independent published papers, so far, [DAPM00b] and [CT01].

PAES is also clearly flexible for use on a wide range of applications since it uses just two operators that are easily designed for most problems: random mutation and initialization. It can be applied to problems of different numbers of objectives, and the number of solutions returned can be controlled.

PAES also provides a good level of search performance. This point is easy to argue since in tests in this dissertation it has demonstrated better performance than some popular MOEAs on some problems. It is therefore capable of providing the required ‘baseline’ at a high enough

level for it to be of merit for a new approach to outperform it. It was also demonstrated that it can generate good results on a broad range of problems. In both of the independent papers where PAES has been used, it performed well also. In the paper by Deb [DAPM00b], PAES was not beaten by the new NSGA-II algorithm on all problems, nor by SPEA on all problems, confirming the general quality of results reported in this thesis. In the paper by Coello, numerical results were not given except for run-times, but visual results indicate that PAES is at least close to competitive with NSGA-II and the new micro-GA in terms of final solutions obtained.

Finally, PAES is simple. It is conceptually simple because it is based on single-point search and mutation only. It is also quite simple in its overall design, although the archiving mechanism does depend on a large cascade of rules. In any case, it is simple compared to most specialized algorithms. This is enough to justify that it would be no more difficult to implement or embed in an application than a specialized algorithm which was outperformed by it. This is important because a baseline algorithm should be a viable alternative to more sophisticated approaches.

These points confirm that PAES is a contribution as a baseline approach. One further positive point is that PAES has lower overall computational cost<sup>1</sup>, per function evaluation, compared to other (MOEA) approaches. Although this is not essential in a baseline algorithm, it is desirable because then it cannot be argued that the baseline algorithm is better only if computational cost is neglected.

We should also consider what has been learnt as a result of using PAES as a baseline algorithm. First, from the test problems F1–F6, we have learnt that the use of elitism and the accurate judgment of solution quality, as used in PAES, appear as important for successful search, as evolving solutions from a population of points, perhaps with recombination, as used, for example, in NPGA. The test problems to which this observation applies are sufficiently diverse, and the results achieved by PAES so consistent, that it seems that this observation may apply to a wide range of problems. Similarly, in the second study in which the performance of SPEA was compared with PAES, the results indicated that local search was a viable alternative to population-based, recombinative approaches on all but the deceptive trap function problem. In the combinatorial problem, knapsack, similar performance between SPEA and PAES was also observed, and on a variety of instances of the ADDMP, PAES actually performed better, according to our metrics. Although we are not yet able to predict when local search may be as good or better than population-based recombinative search, the results in this dissertation do suggest that (single-point) local search cannot be

---

<sup>1</sup>This refers to average-case time complexity; see Chapter 4. The runtimes of PAES given in Chapter 5 also confirm its relatively low computational cost.

dismissed as unsuitable for Pareto optimization, in general.

Let us now consider whether PAES is a viable Pareto optimization technique in its own right. Clearly, the results achieved, discussed above, provide evidence that PAES may be one of several techniques that should be investigated given a new multiobjective optimization problem, as it has outperformed other modern MOEAs on some problems. In the independent study by Deb, as well, PAES outperformed both NSGA-II and SPEA on the multimodal problem, *TC4*, in terms of distance from the true Pareto front. In [CK00], PAES was competitive with PESA and SPEA on some of the test problems, over the larger evaluation limit runs. PAES has also been used or evaluated by several other researchers in real-world applications. These applications include groundwater extraction and echo-location calibration.

We argued in Chapter 4 that a further reason for developing a Pareto hillclimber was to shed some light on the nature of multiobjective landscapes, and whether they differed greatly from single-objective landscapes. In Chapter 5 we conjectured that PAES may be able to move around a multiobjective landscape more easily, in general, than a hillclimber can move around a single-objective landscape. The reason stated was that more points are effectively of the same evaluation because, given a general point in the search space, there are usually a large number of tradeoff solutions that are mutually nondominated, and which are all valid places to ‘move’ to. This is a case of having fewer effective local optima. If the number of effective local optima is lower in multiobjective optimization problem we would expect local search hillclimbers like PAES to do comparatively well. This conjecture has not been investigated in the work presented in the body of the thesis but we discuss some published work in this area, based on PAES, in Section 8.4.

Our last question regarding the contribution of PAES concerns whether it provides a basis or an impetus for the development of new approaches to Pareto optimization. Once again, this question can be answered in the affirmative. In Chapter 4, several variants of PAES were proposed. The multi-start version is particularly promising as a simple but worthy competitor to MOEAs, much as multi-start hillclimbers in single objective optimization. The tabu search and simulated annealing algorithms are also based heavily on (1+1)-PAES, maintaining the use of the dominance relation in the acceptance function, the archive as a comparison set, and the adaptive grid crowding for diversity maintenance. These algorithms are quite different from both current MOEAs and local search multiobjective optimizers that use scalarizing selection. There is much work to be done in evaluating these variants of PAES but they demonstrate that PAES can be used to devise new methods based on local search and the use of Pareto selection.

PAES is also the basis of the M-PAES algorithm developed in Chapter 6 which we evaluate below. It may also have provided some of the impetus for the development of the NSGA-II and the micro-GA. Both of these works cite PAES as an example of a low-computational overhead technique with elitism and efficient diversity maintenance. Both algorithms also use these elements themselves.

### 8.2.2 The adaptive grid archiving strategy

This contribution concerns the maintenance of the nondominated archive in PAES. The problems associated with using the dominance relation in the acceptance function of a single-point searcher led us to adopt the use of a comparison set, in a similar way to Horn and Nafpliotis' NPGA. This, in turn led to the question of how we should update the comparison set, solution by solution, so that it remained bounded, diverse, and mutually nondominated. *This question arises whenever it is desirable to store a limited set of solutions to a multiobjective optimization problem.* Realistically, all multiobjective search algorithms need to store a representative set of the solutions they have found. However, few others have undertaken a solution to this problem. Notably, Zitzler used clustering to reduce the external population in SPEA, and this method works well when a number of solutions (a generation) are being added at once to the set. But to use it for the addition of one solution at a time — as we require for use in a single point searcher — would be computationally too expensive. We investigated several approaches to the archiving procedure. Of these, a method based on using an adaptive grid in the objective space to employ an approximate crowding strategy, was proved to have desirable convergence properties under certain limiting conditions, whilst being computationally very efficient, when used with a quadtree encoding of the grid regions. The method also avoids the need for setting a niche parameter, or having to introduce any scaling of objectives. No prior information about the search space is needed at all, in fact. One parameter, controlling the number of grid regions is required but this only affects the size of the archive needed to guarantee certain convergence properties. Recent studies by Laumanns *et al.* [LZT00a] have independently confirmed the efficacy of the grid crowding approach at maintaining diversity. The use of the adaptive grid was also later used by Corne and Knowles in the population based MOEA, PESA [CK00], to carry out diversity maintenance in the reproductive population.

Some strategies for updating an archive with several solutions at once were also proposed. In the population-based variants of PAES, two problems arose. How to choose the best solution(s) from among the  $\lambda$  new ones, with respect to the archive, and how to update that archive with the new solutions. The proposed approach did not seem to perform well, how-

ever, as  $(1+1)$ -PAES was generally a better performer than the population-based variants. For the tabu search version of PAES, we thus proposed a different method of updating the nondominated solutions archive, and of choosing which of the solutions to ‘accept’. Although the proposed method was not implemented or tested, we argued that its design should overcome the problems identified in the original approach put forward for use in  $(1 + \lambda)$  and  $(\mu + \lambda)$  PAES.

### 8.2.3 Metrics for performance assessment

In Chapter 3, we considered the issue of assessing and comparing stochastic multiobjective optimizers. Using the outperformance relations proposed by Hansen and Jaszkiewicz, we were able to classify and critically evaluate many of the most visible methods of performance assessment. We consider this an important contribution as many of the metrics in use do not provide useful evaluations of the output of different optimizers. We hope that this review will discourage the use of some metrics which were identified as being of very limited use, and encourage those metrics which accurately measure more salient properties of nondominated sets.

We also proposed some new assessment methods based on the  $\mathcal{S}$  metric [Zit99] and sampling of attainment surfaces, following Fonseca and Fleming. These methods combine, in different ways, evaluations of the nondominated sets from a number of independent runs. A number of these methods were used in this thesis, enabling sound evaluation of the different algorithms tested.

We particularly encourage other researchers to report on ‘hard’ benchmark problems by using some meaningful, absolute measure of the overall quality of the Pareto fronts achieved. This would encourage competition and accurate evaluation of new methods. We contend that the size of the combined attainment surface, the median attainment surface, and interquartile attainment surfaces achieved, which we used to report the benchmark  $d$ -MST results will allow others to compare their results meaningfully against ours, in the future. We also endorse the utility function based metrics of Hansen and Jaszkiewicz [HJ98] for reporting on benchmark results.

### 8.2.4 M-PAES

The memetic PAES algorithm developed and tested in Chapter 6 is the first multiobjective memetic algorithm based entirely on Pareto selection. How much of a contribution this al-

gorithm is will depend upon the outcome of testing M-PAES more thoroughly in future. In particular, studies comparing M-PAES with RD-MOGLS, the cellular MOGLS algorithm, and newer hybrid MOEAs such as Deb and Goel's Hybrid MOEA [DG01] should shed further light on whether Pareto selection-based local improvement offers any advantage over scalarizing selection methods. However, whether or not M-PAES is more effective than other techniques, it is an original contribution to the available methods, and it has already been independently compared with RD-MOGLS [Jas00]. The M-PAES results presented in this thesis indicate that it is a promising approach, able to outperform MOEAs using the same operators and heuristics. The true worth of M-PAES may be seen when good local improvement heuristics are available for a particular problem. With the *mcd*-MST problem we glimpsed the potential of M-PAES because on this problem local search seems to be particularly useful. However, a more specialized local search procedure than successive random mutations may offer a greater advantage to the M-PAES approach.

### 8.2.5 MST benchmarks

Further contributions have been made in the development of heuristics for the *d*MST and multi-criterion MST problems. These heuristics have been tested on a range of benchmark problem instances. Because existing available instances for these problems were not always sufficiently difficult, new benchmark problem generators were also devised. The generator for the *d*-MST was used by Raidl in [Rai00] to provide difficult problem instances for testing his new GA approach. The generators developed for the multiobjective MST can produce combinatorial optimization problem instances with different shaped Pareto fronts including concave ones, and with varying difficulty in terms of constraints. These should be a valuable addition to MOEA test suites. Benchmark results for some of these instances have also been reported.

## 8.3 Limitations

### 8.3.1 PAES

There are a number of ways in which the central contributions of this thesis, namely the PAES framework and algorithms, are limited. First, the empirical testing of (1+1)-PAES has only just begun and there are a number of key issues relating to its performance that have not yet been answered. In particular, PAES has only been tested on problems possessing integer valued parameters. Even where the continuous test functions  $F1$ – $F5$  and  $\mathcal{T}1$ – $\mathcal{T}5$  were used,

these functions were tackled using a binary encoding of the (heavily discretized) parameters. The question of how well PAES performs should a more natural encoding be used, together with an appropriate Gaussian mutation operator (or similar) like those used in real ESs, has not been considered in this thesis. The use of real values, and continuous variation operators also opens up the possibility of using and evolving strategy parameters, another avenue that has not been considered here.

The experimental evaluation of PAES on test functions has not encompassed functions with constraints, either. We have not proposed or investigated constraint handling techniques that would work effectively within the PAES framework. Other researchers have recently applied (1+1)-PAES to constrained problems, however [CT01], and further study on this important aspect of the algorithm will surely follow.

More importantly perhaps, it would have been in keeping with the themes and philosophy of this thesis to compare the performance of (1+1)-PAES with other simple approaches which may be candidates for strong baseline approaches to Pareto optimization. For example, it would be interesting to compare the performance of (1+1)-PAES with a random search algorithm using the same archiving strategy. It remains an open question whether such an approach would perform better than PAES on test functions such as the deceptive and highly multimodal test functions  $\mathcal{T}4$  and  $\mathcal{T}5$ . It is also regrettable that (1+1)-PAES has not been compared directly with a simple local searcher that uses scalarizing selection. We have made the hypothesis that Pareto selection using an archive is more efficient (in terms of function evaluations) at obtaining a diverse and close approximation to the Pareto front, than if scalarizing selection (with randomly selected scalarizing vectors at each step) is used, but we have not empirically tested this hypothesis so far. It also remains an avenue of future research for methods based on the PAES framework, including SA and TS versions, to be compared with similar methods deriving from the MCDM literature which are based on scalarizing, rather than Pareto, methods.

The testing of population-based variants of PAES has been limited. The early experiments reported in [KC00b] showed that on some problems the use of a population was beneficial but because of the way selection and archiving were combined in these PAES variants, poor performance was generally seen. The testing of multi-start PAES and the other variants carried out to date has been unstructured and is not reported in this thesis.

### 8.3.2 M-PAES

The key limitation with the M-PAES algorithm is the difficulty we have at times experienced in setting its various parameters. Much further investigation is required to express the useful ranges of the parameters, and how they interact with one another, and to devise a reliable method of setting them. On the mc-MST problems we found that M-PAES was robust to the settings we used, across the set of instances tackled. This is encouraging because on some of the earlier test function problems, it was quite difficult to set parameters effectively.

### 8.3.3 Performance metrics

In this thesis, some of the performance metrics used to assess algorithm performance, statistically, are subject to several objections. First, the method **AS3** for testing multiple runs from multiple algorithms is not statistically significant in the same way as the metric **AS2** (for testing runs from only two algorithms) is. Fonseca has noted<sup>2</sup> that there are multiple testing issues with **AS3**. For example, if three samples  $a, b, c$  from three different algorithms  $A, B, C$  are to be compared then three tests must be carried out:  $a$  vs  $b$ ,  $a$  vs  $c$ , and  $b$  vs  $c$ . If  $a$  beats both  $b$  and  $c$  at the 95% confidence level in separate tests using the same sample from  $A$  in both cases then it is not statistically correct to conclude that  $A$  beats both  $B$  and  $C$  with 95% confidence. In fact, it is not even possible to say that  $A$  beats both  $B$  and  $C$  with a confidence of 0.95<sup>2</sup>. This is because only one sample ( $a$ ) is being used to make inferences in two separate tests. Using separate samples does not solve this problem either. For now, we accept that the interpretation of the output of **AS3** should not be that the ‘unbeaten’ and ‘beats all’ statistics are significant at the confidence levels used in each individual test. Rather, they are significant at some lower level which may be difficult to calculate. Further research is needed to be able to estimate the  $\alpha$  level that should be used in the pair-wise tests to achieve a particular confidence level in the outcome of the multiple test.

Second, it is possible to calculate the median attainment surface exactly (that is, without sampling) in any number of dimensions, provided that the total number of attainment surfaces  $\alpha$  is odd<sup>3</sup>, although the computational cost of doing this is high if the number of objectives is greater than two. Thus we could obtain more accurate values for the metric **S2** which measures the hypervolume of the median attainment surface. In two dimensions, the median attainment surface can be found in  $O(n \log n)$  time by first sorting the points in the first objective and then finding all the points that define where the ‘middle’ surface in the other

<sup>2</sup>Personal communication.

<sup>3</sup>Personal communication with Carlos Fonseca.

dimension lies. Using such an approach, we could have calculated exact **S2** values for the mc-MST problems with low computational overhead.

## 8.4 Further work

### 8.4.1 Reducing the effect of local optima in search

To date, little research in Pareto optimization has focused on characterizing the nature of search landscapes under different multiobjective selection strategies, and how these might differ from landscapes in single-objective optimization. One interesting question related to this is whether local optima (defined on some neighbourhood operator) in a single-objective problem can be “removed” if the problem is re-formulated as a Pareto optimization problem. The reason for thinking that this might be possible is that in a Pareto optimization landscape, points may be equal to, better (dominating), worse (dominated), or incomparable (nondominated) with respect to other points in the search space. In contrast, in single objective optimization, they may be only equal, better or worse. The fact that points in a Pareto optimization problem can be incomparable opens the door to the possibility of the landscape looking quite different. Consider using a (local search) Pareto selection algorithm which allows moving from one solution to any other solution that is incomparable. Allowing this kind of move means that the number of solutions that it is possible to ‘walk’ to from any given solution may be increased relative to the situation in single-objective local-search. Therefore, the number of effective local optima in the search landscape may be reduced compared to what is usually experienced in single-objective optimization.

In work recently published by Knowles *et al.* [KWC01], the effect on local search of potentially reduced optima was investigated. In the paper, two different single-objective problems were “multiobjectivized”, that is, reformulated as Pareto optimization problems: H-IFF [WP00] and TSP. On both problems the performance of (1+1)-PAES on the transformed (multi-objective) landscape was compared with that of a simple hillclimber (SHC) on the original landscape. On both problems, PAES was able to search more effectively than SHC using the same neighbourhood operators. On H-IFF, PAES was competitive with using simulated annealing. In the case of the TSP problem, the multiobjectivization of the problem was achieved by arbitrarily defining two sub-tours, and seeking to minimize both. Over a set of different types of instance, PAES was always significantly better than SHC in terms of the average of final tour lengths found, given equal numbers of evaluations. Other experiments showed that the effect was still significant when population-based (non-recombinative) local searchers were used, as well.

This study leaves many questions unanswered, and motivates several new lines of research. In further work it might be fruitful to investigate if there exist any real-world problems where a multiobjectivization can be found in which PAES performs competitively with simulated annealing or tabu search, or other local search methods that deal effectively with local optima. Other lines of enquiry include investigating the optimal number of objectives to balance exploration and exploitation in the search space. It may be best to begin search with a large number of objectives, which would reduce local optima and encourage exploration, and gradually conglomerate these back to a single objective function to encourage exploitation.

#### 8.4.2 Highly multiobjective problems

It is an unfortunate fact that the vast majority of the research in evolutionary multiobjective optimization actually concerns problems of only two objectives. In this thesis, some problems with three and four objectives have been attempted but the test functions and problems that are currently available mean that (1+1)-PAES and the other algorithms have not been tested on highly multiobjective problems. Fonseca and Fleming [FF95] conjectured some time ago that pure Pareto EAs would not perform well on problems with large numbers of competing objectives, partly because the tradeoff front would become too large and unmanageable for good compromise solutions to be found. This led them to develop methods for incorporating preferences into their Pareto ranking methods. Similarly, preference approaches have been developed by Cvetkovic and Parmee [CP99], who have tackled conceptual design problems with up to 13 objectives. Despite the undoubtedly need for these kind of mixed search/decision making approaches for engineering applications, it seems that the question of how to deal with large numbers of objectives *in the absence of preferences* is still an interesting theoretical question.

This question may not be entirely academic, either. In some game-playing applications, in which many different strategies exist, a player can only be accurately evaluated by considering its success against other players. In many interesting games the payoff relationships between the different strategies are intransitive. Thus, to evaluate a new strategy it may be necessary to test it against a large number of other game-playing opponents. In this scenario, the score against each opponent can be considered as a different objective. In a recent paper [NW01], Noble and Watson showed that better strategies were co-evolved in a game of Texas Hold'em poker when Pareto selection was used to select nondominated playing strategies than if selection was based on the aggregated scores against all opponents. In this kind of co-evolutionary application, one is not trying to find a single good compromise solution; rather, one hopes to leverage the performance of the whole co-evolving population by using a good selection

strategy. Clearly, in this kind of application the number of objectives may be quite high.

In applications like the game-playing one just described, and also on truly multiobjective test problems (i.e. with  $K >> 2$ ) like those proposed in [DTLZ01], the archiving strategy of PAES might perform very poorly. This is because the number of solutions desired (and hence the archive size) will be small compared to the number of objectives in the problem. This will mean that many of the requirements necessary to achieve a good distribution of solutions in the archive will be violated. Other Pareto methods will face similar problems because in high dimensional objective spaces too many nondominated solutions exist. This might be overcome in a number of different ways. One strong possibility is to use other scaling and range independent relations in place of the dominance relation, in order to rank solutions. For example, *favour* [DDB01] may help rank solutions that are normally incomparable using Pareto dominance alone. The epsilon-dominance relation, recently used in archiving strategies proposed by Laumanns *et al.* [LTDZ01], is another technique capable of restricting the number of incomparable solutions discovered. Improvements to the ability of the PAES archiving strategy to handle numerous objectives will be a key area of further work.

## Appendix A

# Proof that Upper Grid Boundaries Converge

Convergence of the upper grid boundaries can only be proved under certain conditions. Consider a three objective case in which the vector in the efficient set with largest value in objective 1 is  $\mathbf{z}^1 = (6, 1, 3)$ . Another vector in the Pareto set is  $\mathbf{z}^2 = (6, 3, 1)$ . However, the non-efficient vector  $\mathbf{z}^3 = (7, 3, 2)$  also exists. If  $\mathbf{z}^1$  (but not  $\mathbf{z}^2$ ) is currently in the archive and  $\mathbf{z}^3$  is generated at the next step, then it will be accepted and the grid boundaries will be updated. Thus, the upper grid boundary on objective 1 will become larger than the any vector in the efficient set's value in this objective. However, this value of the grid boundary may not be maintained because if  $\mathbf{z}^2$  is generated at some future time,  $\mathbf{z}^3$  will be dominated, and removed from the archive, and the grid boundaries will be appropriately updated, accordingly. Because it is possible for  $\mathbf{z}^2$  to be 'lost' from the archive, in future iterations, the same cycle of events can occur again. Therefore, in this kind of situation the upper grid boundaries may never converge.

Condition A.1, below, formalizes the properties which must be true of  $Z^*$  if we are to avoid situations such as the above, hence enabling us to prove that the upper grid boundaries will converge. In the following, we prove the convergence of the upper boundaries in the restricted case when Condition A.1 is true.

**Condition A.1**  $\forall k, \exists \mathbf{z} \in C(Z^*), z_k \geq \max_{Z^*} z_k, \exists \mathbf{z}^* \in Z^*, z_k^* = \max_{Z^*} z_k, \mathbf{z} \sim \mathbf{z}^*$ , where  $C(Z^*)$  denotes the complement of the efficient set. In other words, on any objective, there are no non-efficient vectors that have a component value that is larger than or equal to the largest component value of any efficient vector, and that are also nondominated with respect

---

to any of the efficient vectors with the maximum component value.

**Lemma A.1** *If Condition A.1 is true and a vector  $\mathbf{z}^*$  with component  $z_k = \max z_{k,Z^*}$  for some  $k$  is generated at time  $t$ , then  $\max z_{k,M_t} = \max z_{k,Z^*}$ .*

**Proof A.1** *Assume that a vector  $\mathbf{z}^*$  with component  $z_k = \max z_{k,Z^*}$  for some  $k$  is generated at time  $t$ . Then we have  $\max z_{k,N_t} = \max z_{k,Z^*}$ .*

*The vector  $\mathbf{z}^*$  will enter the archive if  $\text{Domination}(t)$  or  $\text{Diverge}(t)$  or  $\text{Low\_pop\_region}(t)$  or  $\text{Fill}(t)$  executes, and hence it is clear that  $\max z_{k,M_t} \geq \max z_{k,Z^*}$ . But because Condition A.1 is true, we also know that  $M_t$ , which is always a nondominated set, cannot contain any non-efficient vector with  $z_k \geq \max z_{k,Z^*}$ . Thus,  $\max z_{k,M_t} = \max z_{k,Z^*}$ .*

*if  $\mathbf{z}^*$  does not enter the archive, it is because  $\text{Steady\_state}(t)$  executes. However, one of the conditions for  $\text{Steady\_state}(t)$  to execute is that  $\forall k, \max z_{k,N_t} = \max z_{k,M_{t-1}}$ . So we have  $z_k = \max z_{k,Z^*}$  and  $\forall k, \max z_{k,N_t} = \max z_{k,M_{t-1}}$ , and  $\text{Steady\_state}(t)$  executes. Therefore, since  $\mathbf{z}^* \in N_t$ , then  $\max z_{k,M_{t-1}} = \max z_{k,N_t} = \max z_{k,Z^*}$ , and because  $\text{Steady\_state}(t)$  executes  $\max z_{k,M_t} = \max z_{k,M_{t-1}}$ , so  $\max z_{k,M_t} = \max z_{k,Z^*}$ , as required.*

**Lemma A.2** *If Condition A.1 is true, and for some  $k \in 1..K, \exists \mathbf{z}^* \in M_{t_m}$  with  $z_k = \max z_{k,Z^*}$  then  $\forall t > t_m, \exists \mathbf{z} \in M_t$  with  $z_k = \max z_{k,Z^*}$ .*

**Proof A.2** *Assume that for some  $k \in 1..K, \exists \mathbf{z}^* \in M_{t_m}$  with  $z_k = \max z_{k,Z^*}$ . We show that no archiving rule is capable of removing all the vectors with component  $z_k = \max z_{k,Z^*}$ .*

*Let us consider each of the rules that can remove vectors. These are the rules  $\text{Diverge}(t)$ ,  $\text{Low\_pop\_region}(t)$ , and  $\text{Domination}(t)$ .*

*$\text{Diverge}(t)$  can remove only one vector from the archive. If there is only one vector  $\mathbf{z} \in M_t$  with  $z_k = \max z_{k,Z^*}$  then, since Condition A.1 is true, there can be no vectors in  $M_t$  with  $z_k > \max z_{k,Z^*}$  (because they would be dominated). Thus  $\mathbf{z}$  is a unique extremum in the archive and  $\text{Diverge}(t)$  cannot remove it by the definition of  $\text{Diverge}(t)$ .*

*On the other hand, if there are  $n > 1$  vectors in the archive with  $z_k = \max z_{k,Z^*}$  then  $\text{Diverge}(t)$  may remove one of them. However, one or more will remain in the archive.*

*$\text{Low\_pop\_region}(t)$  can also remove only one vector from the archive and only if it is not a unique extremum. Thus, the same argument as for  $\text{Diverge}(t)$  applies, so  $\text{Low\_pop\_region}(t)$  cannot remove all vectors with  $z_k = \max z_{k,Z^*}$ .*

---

Domination( $t$ ) can remove multiple vectors at once. However, it cannot remove any efficient vector. Since Condition A.1 is true any vector in  $M_t$  with component  $z_k = \max z_{k,Z^*}$  must be efficient. Therefore Domination( $t$ ) cannot remove these vectors.

**Theorem A.1** *If Condition A.1 is true then the upper boundaries of the grid  $ub_{k,t}$  converge for all  $k$ .*

**Proof A.3** *To show that the upper boundaries converge it is sufficient to show that there is a time  $t_m$  such that  $\forall t \geq t_m, \forall k, \max z_{k,M_t} = \max z_{k,Z^*}$ . For this we just require:*

1. *If a vector  $\mathbf{z}^*$  with component  $z_k = \max z_{k,Z^*}$  for some  $k$  is generated at time  $t$ , then  $\max z_{k,M_t} = \max z_{k,Z^*}$ . This is Lemma A.1.*
2. *If for some  $k \in 1..K, \exists \mathbf{z}^* \in M_{t_m}$  with  $z_k = \max z_{k,Z^*}$  then  $\forall t > t_m, \exists \mathbf{z} \in M_t$  with  $z_k = \max z_{k,Z^*}$ . This is Lemma A.2.*

Due to Condition A.1, Item 2. ensures that  $\forall t \geq t_m, \forall k, \max z_{k,M_t} = \max z_{k,Z^*}$  because vectors with  $z_k > \max z_{k,Z^*}$  are always dominated.

We now provide two cases when Condition A.1 is always true, so that the upper boundaries of the adaptive grid converge.

**Theorem A.2** *if  $K = 2$  then Condition A.1 is true.*

**Proof A.4** *Assume  $K = 2$  and Condition A.1 is not true. Then there is a vector  $\mathbf{z} \in C(Z^*)$  and an objective  $k \in 1..2$  such that  $z_k \geq \max z_{k,Z^*}$ , and  $\mathbf{z}$  is nondominated with respect to a vector  $\mathbf{z}^* \in Z^*$  with component  $z_k = \max z_{k,Z^*}$ .*

*Without loss of generality, we choose  $k = 1$ . Since  $z_1 \geq \max z_{1,Z^*}$  and  $\mathbf{z} \sim \mathbf{z}^*$ , then we have  $z_2 < z_2^*$ . But this must mean that  $\mathbf{z}$  is efficient (a contradiction) because if it were not there would have to be another vector  $\mathbf{z}^{**} \in Z^* < \mathbf{z}$ . But then  $\mathbf{z}^{**} < \mathbf{z}^*$  because  $z_1^{**} \leq \max z_{1,Z^*}$  and  $z_2^{**} \leq z_2$ . This is also a contradiction since  $\mathbf{z}^*$  is efficient.*

**Theorem A.3** *if  $\forall k, \exists \mathbf{z} \in C(Z^*), z_k \geq \max z_{k,Z^*}$  then Condition A.1 is true. In other words, if the efficient set spans the feasible objective space in all objectives then Condition A.1 is true.*

**Proof A.5**  $\forall k, \exists \mathbf{z} \in C(Z^*), z_k \geq \max z_{k,Z^*} \Rightarrow \forall k, \exists \mathbf{z} \in C(Z^*), z_k \geq \max z_{k,Z^*}, \exists \mathbf{z}^* \in Z^*, z_k^* = \max z_{k,Z^*}, \mathbf{z} \sim \mathbf{z}^*$ .

## Appendix B

### Proof of Theorem 4.4

**Conjecture B.1** *The maximum number of mutually non-inferior regions in a  $K$  dimensional vector space divided up into  $\text{div}$  equal divisions in each dimension, is  $\text{div}^K - (\text{div} - 1)^K$ .*

**Lemma B.1** *There are  $\text{div}^K - (\text{div} - 1)^K$  regions with a co-ordinate vector  $\mathbf{c} = (c_1, c_2, \dots, c_K)$  such that  $\forall i \in 1..K, c_i = 1..\text{div}, \exists j \in 1..K, c_j = 1$ . In other words, there are  $\text{div}^K - (\text{div} - 1)^K$  regions with a co-ordinate vector in which at least one of component has the value 1.*

**Proof B.1** *Let  $\mathbf{c} = (c_1, c_2, \dots, c_K)$  be the co-ordinate of a region, with  $\forall i \in 1..K, c_i \in 1..\text{div}$ . Clearly there are  $\text{div}^K$  such regions.*

*Let  $\mathbf{d} = (d_1, d_2, \dots, d_K)$  be the co-ordinates of a region with no components having the value 1. So,  $\forall i \in 1..K, d_i \in 2..\text{div}$ . Clearly there are  $(\text{div} - 1)^K$  of these regions. Therefore, by subtracting the second set of regions from the first, we have that there are  $\text{div}^K - (\text{div} - 1)^K$  regions with a co-ordinate vector in which at least one of its components has the value 1.*

**Definition B.1** *Every region with co-ordinates  $\mathbf{c} = (c_1, c_2, c_3, \dots, c_K)$  with  $\forall i \in 1..K, c_i \in 1..\text{div}$ , can be mapped to a unique root region of  $\mathbf{c}$  with co-ordinates  $\text{rt}_{\mathbf{c}} = (c_1 - a_c, c_2 - a_c, c_3 - a_c, \dots, c_K - a_c)$ , where  $a_c = \min(\{c_i \mid i \in 1..K\}) - 1$ . Notice that a root region has the property that at least one of its co-ordinates has the value 1.*

**Lemma B.2** *Two regions with co-ordinates  $\mathbf{c} = (c_1, c_2, c_3, \dots, c_K)$  and  $\mathbf{d} = (d_1, d_2, d_3, \dots, d_K)$ , with  $\forall i \in 1..K, c_i < d_i$ , share the same root region if and only if  $\forall b \in \mathbb{N}^+, \forall i \in 1..K, d_i = c_i + b$ , and  $\forall i \in 1..K (c_i, d_i \in 1..\text{div})$ .*

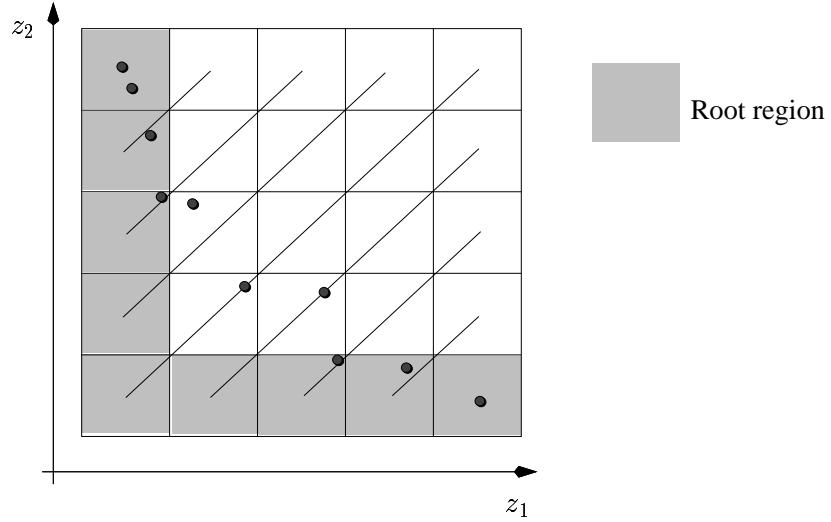


Figure B.1: An illustration of root regions. Diagonal lines have been drawn to show all of the regions that map to the same root region. Clearly, if two vectors are in different regions on the same diagonal then one must be superior to the other. Therefore, any nondominated set (like the one shown) cannot occupy more regions than there are root regions.

**Proof B.2** We have:

$\mathbf{c} = (c_1, c_2, c_3, \dots, c_K)$  and  $\mathbf{d} = (c_1 + b, c_2 + b, c_3 + b, \dots, c_K + b)$  and  $a_c = \min(\{c_i \mid i \in 1..K\}) - 1$ ,  $a_d = \min(\{c_i + b \mid i \in 1..K\}) - 1$ .

Clearly,  $a_c = a_d - b$ .

Now,  $\mathbf{rt}_c = (c_1 - a_c, c_2 - a_c, c_3 - a_c, \dots, c_K - a_c)$  and  $\mathbf{rt}_d = (c_1 + b - a_d, c_2 + b - a_d, c_3 + b - a_d, \dots, c_K + b - a_d)$

By substituting  $a_d - b$  for  $a_c$  in the expression for  $\mathbf{rt}_c$  we have:

$\mathbf{rt}_c = (c_1 + b - a_d, c_2 + b - a_d, c_3 + b - a_d, \dots, c_K + b - a_d) = \mathbf{rt}_d$ , proving the sufficient condition.

To prove the necessary condition:

Assume  $\mathbf{rt}_c = \mathbf{rt}_d$  and  $\mathbf{c} = (c_1, c_2, c_3, \dots, c_K)$  and  $\mathbf{d} = (c_1 + b_1, c_2 + b_2, c_3 + b_3, \dots, c_K + b_K)$  and  $\neg(b_1 = b_2 = b_3 = \dots = b_K)$ . Now  $\mathbf{rt}_c = (c_1 - a_c, c_2 - a_c, c_3 - a_c, \dots, c_K - a_c)$  and  $\mathbf{rt}_d = (c_1 + b_1 - a_d, c_2 + b_2 - a_d, c_3 + b_3 - a_d, \dots, c_K + b_K - a_d)$

Since  $\mathbf{rt}_c = \mathbf{rt}_d$ ,  $(b_1 - a_d, b_2 - a_d, b_3 - a_d, \dots, b_K - a_d) = (-a_c, -a_c, -a_c, \dots, -a_c)$  which implies that  $b_1 = b_2 = b_3 = \dots = b_K$ , a contradiction.

**Lemma B.3** If two regions have different co-ordinates but map to the same root region then one is superior to the other.

---

**Proof B.3** From lemma B.2 the two regions must have co-ordinates  $\mathbf{c} = (c_1, c_2, c_3, \dots, c_K)$  and  $\mathbf{d} = (c_1 + b, c_2 + b, c_3 + b, \dots, c_K + b)$  with  $b \in \mathbb{N}^+$ . Therefore  $c$  is superior to  $d$  because its co-ordinates are lower in every dimension.

**Corollary B.1** In any set of mutually non-inferior regions, each region must map to a different root region. This follows from lemma B.3.

**Theorem B.1** The maximum number of mutually non-inferior regions in a  $K$  dimensional vector space divided up into  $\text{div}$  equal divisions in each dimension, is  $\text{div}^K - (\text{div} - 1)^K$ .

**Proof B.4** We can see that there are exactly  $\text{div}^K - (\text{div} - 1)^K$  root regions in the vector space from lemma B.1 and the definition of a root region. Since every non-inferior region maps to a different root region B.2 then the maximum number of non-inferior regions is also  $\text{div}^K - (\text{div} - 1)^K$ .

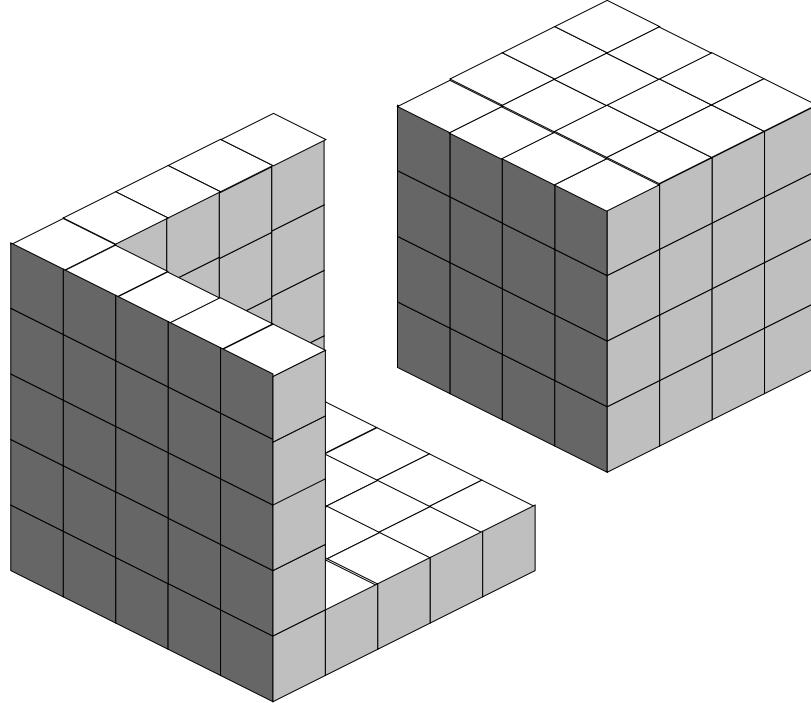


Figure B.2: An illustration of the equation  $\text{div}^K - (\text{div} - 1)^K$ .

## Appendix C

### Derivation of Equation 4.3

#### Reproduced from [Rob01]

Q. What is the number of ways of getting the sum  $s$  on  $n$  dice with  $x$  faces each?

A. The generating function for one die with  $x$  sides is

$$f(z) = z + z^2 + z^3 + \dots + z^x. \quad (\text{C.1})$$

The coefficient of a term tells you how many ways you can roll the exponent of  $z$  in the term. In this case, for numbers from 1 to  $x$ , you can roll each in one way. For other numbers, like 0 and numbers greater than  $x$ , you can roll them in zero ways. Thus the above generating function is right for one die. For two dice, square the above function. For  $n$  dice, raise it to the  $n$ th power. Then you want the coefficient of  $z^s$  in that expression. When calculating this coefficient for some specific value of  $s$ , you can do the arithmetic and ignore all powers of  $z$  with exponents bigger than  $s$ . If you want all the probabilities, you can ignore all exponents bigger than  $(n \cdot x + 1)/2$ , because the number of ways of rolling  $s$  is the same as the number of ways of rolling  $n \cdot x + 1 - s$ , and if one is larger than  $(n \cdot x + 1)/2$ , then the other is smaller.

---

To express this in binomial coefficients, you can write:

$$f(z) = (z - z^{x+1})/(1 - z) \quad (C.2)$$

$$f(z)^n = [(z - z^{x+1})/(1 - z)]^n \quad (C.3)$$

$$= z^n \cdot (1 - z^x)^n \cdot (1 - z)^{-n} \quad (C.4)$$

$$= z^n \cdot \sum_{k=0}^n (-1)^k \cdot \binom{n}{k} \cdot z^{(x \cdot k)} \cdot \sum_{i=0}^n \binom{n+i-1}{i} \cdot z^i \quad (C.5)$$

$$= \sum_{k=0}^n \sum_{i=0}^n (-1)^k \cdot \binom{n}{k} \cdot \binom{n+i-1}{n-1} \cdot z^{(n+x \cdot k+i)}. \quad (C.6)$$

Now the coefficient of  $z^s$  in this will be given by:

$$\sum_{k=0}^{(s-n)/x} (-1)^k \cdot \binom{n}{k} \cdot \binom{s-x \cdot k-1}{n-1}. \quad (C.7)$$

## Appendix D

# Proof of Incorrectness of Zhou and Gen's Proposed Enumeration Algorithm for Pareto Optimal Multi-criterion Spanning Trees

### D.1 Statement of the problem

The multi-criterion minimum spanning tree (mc-MST) problem can be simply stated. Given a weighted graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E$  and edge weight vectors  $w_i(e) \in \mathbb{R}^+, e \in E, i \in 1..K$  where  $K$  is the number of criteria, find a spanning tree  $T$  in  $G$  such that there does not exist another spanning tree whose total weights Pareto dominate  $T$ .

### D.2 Zhou and Gen's proposed enumeration algorithm

In an article by Zhou and Gen [ZG99], an algorithm for enumerating all Pareto optimal spanning trees was proposed. The operation of the algorithm can be summarised as follows:

Step 1: Pick an arbitrary start-vertex  $v_1$ . In turn, consider each edge adjacent to  $v_1$ . Put all edges that are nondominated into a set of subtrees  $S$ .

Step 2: For each subtree  $s \in S$  consider, in turn, each adjacent edge that does not cause a cycle to be created when added to  $s$ . For each such edge that can be added to  $s$  to form a new subtree  $t$  which is nondominated by any other subtree sprouting from  $s$ , put  $t$  into a new set  $T$ .

Step 3:  $S \leftarrow T$ ,  $T \leftarrow \emptyset$ . Compact down the set of subtrees  $S$  such that all dominated subtrees, and all repeated subtrees, are removed.

Step 4: If the subtrees in  $S$  have  $V - 1$  edges then  $S$  is the required set of unique, Pareto optimal spanning trees. Else return to Step 2.

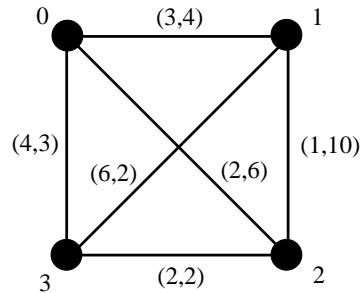
In [ZG99] the above algorithm was used to enumerate the set of Pareto optimal spanning trees on some small mc-MST instances. The resulting solution sets were then used to measure the proportion of solutions that were Pareto optimal from those generated by the authors' proposed genetic algorithm. In the next section we prove that the enumerative algorithm is incorrect: it neither guarantees returning all Pareto optimal solutions, nor that those returned are Pareto optimal.

### D.3 Incorrectness of the proposed approach

**Theorem D.1** *The proposed algorithm is not guaranteed to generate all Pareto optimal spanning trees.*

**Proof D.1** *The proof is by example. Consider a 4-vertex weighted complete graph  $G_1$  in which each edge has two weights associated with it as follows:*

| edge | weights |
|------|---------|
| 0-1  | (3,4)   |
| 0-2  | (2,6)   |
| 0-3  | (4,3)   |
| 1-2  | (1,10)  |
| 1-3  | (6,2)   |
| 2-3  | (2,2)   |



*In total there are 16 spanning trees of the graph  $G_1$ , of which 6 are Pareto optimal. The complete list of spanning trees of  $G_1$  is given in Table D.1 with the 6 Pareto optimal spanning*

*trees shown diagrammatically.*

*Given  $G_1$  and a start-vertex of 0, the algorithm of Zhou and Gen generates the trace given in Table D.2. The trace is interpreted as follows: The left hand column lists all the edges that can be added to  $v_1 = 0$ , and next to each listed edge, its corresponding weight vector is given. Since all of these edges are nondominated each becomes a subtree in  $S$ , to which other edges can be joined. The second column shows each of the edges that can be added to each of the subtrees in column 1, each forming a new subtree of two edges. Similarly, for each subsequent column, the edges that can be added to the subtree of the previous column are listed. Next to each subtree in the trace the total vector subtree weight is given. A double-asterisked subtree denotes one which is dominated by other subtrees sprouting from the same subtree at the previous level, and a single-asterisked solution denotes one which is dominated by a subtree sprouting from a different sub-tree at the previous level. Both single and double-asterisked solutions are discarded in Zhou and Gen's algorithm, and no further edges are added to them. The final solutions returned by the algorithm are thus those in the right column with no asterisks. A repeated tree, i.e. one with identical edges to another one, is denoted by an "R". The trace shows that the algorithm generates only 4 of the 6 Pareto optimal spanning trees of the graph  $G_1$ .*

| Edges                     | Total tree weights | Pareto optimal | Figure |
|---------------------------|--------------------|----------------|--------|
| 0-1,0-2,0-3               | (9,13)             | no             |        |
| 0-1,0-2,1-3               | (11,12)            | no             |        |
| <b>0-1,0-2,2-3 (7,12)</b> |                    | yes            |        |
| 0-1,0-3,1-2               | (8,17)             | no             |        |
| <b>0-1,0-3,2-3 (9,9)</b>  |                    | yes            |        |
| 0-1,1-2,1-3               | (10,16)            | no             |        |
| <b>0-1,1-2,2-3 (6,16)</b> |                    | yes            |        |
| <b>0-1,1-3,2-3 (11,8)</b> |                    | yes            |        |
| 0-2,0-3,1-2               | (7,19)             | no             |        |
| 0-2,0-3,1-3               | (12,11)            | no             |        |
| 0-2,1-2,1-3               | (9,18)             | no             |        |
| <b>0-2,1-2,2-3 (5,18)</b> |                    | yes            |        |
| 0-2,1-3,2-3               | (10,10)            | no             |        |
| 0-3,1-2,1-3               | (11,15)            | no             |        |
| 0-3,1-2,2-3               | (7,15)             | no             |        |
| <b>0-3,1-3,2-3 (12,7)</b> |                    | yes            |        |

Table D.1: The spanning trees of  $G_1$ .

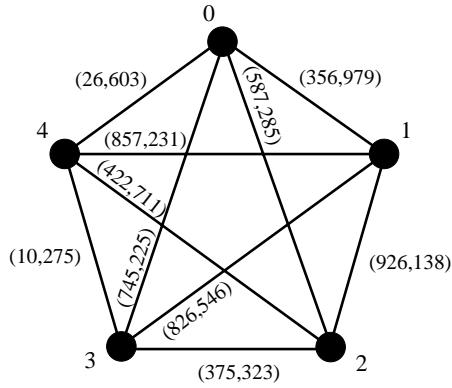
|                  |                     |
|------------------|---------------------|
| <b>0-1</b> (3,4) |                     |
|                  | <b>0-2</b> (5,10)*  |
|                  | <b>0-3</b> (7,7)*   |
|                  | <b>1-2</b> (4,14)*  |
|                  | <b>1-3</b> (9,6)*   |
| <b>0-2</b> (2,6) |                     |
|                  | <b>0-1</b> (5,10)** |
|                  | <b>0-3</b> (6,9)**  |
|                  | <b>1-2</b> (3,16)   |
|                  | <b>0-3</b> (7,19)** |
|                  | <b>1-3</b> (9,18)** |
|                  | <b>2-3</b> (5,18)   |
| <b>2-3</b> (4,8) |                     |
|                  | <b>0-1</b> (7,12)   |
|                  | <b>1-2</b> (5,18)R  |
|                  | <b>1-3</b> (10,10)* |
| <b>0-3</b> (4,3) |                     |
|                  | <b>0-1</b> (7,7)**  |
|                  | <b>0-2</b> (6,9)**  |
|                  | <b>1-3</b> (10,5)** |
|                  | <b>2-3</b> (6,5)    |
|                  | <b>0-1</b> (9,9)    |
|                  | <b>1-2</b> (7,15)*  |
|                  | <b>1-3</b> (12,7)   |

Table D.2: The trace of the algorithm of Zhou and Gen as it generates spanning trees of the graph  $G_1$ , using start-vertex  $v_1 = 0$ .

**Theorem D.2** *The algorithm of Zhou and Gen can generate spanning trees that are not Pareto optimal.*

**Proof D.2** *The proof is by example. Consider a 5-vertex weighted complete graph  $G_2$  in which each edge has two weights associated with it as follows:*

| edge | weights   |
|------|-----------|
| 0-1  | (356,979) |
| 0-2  | (587,285) |
| 0-3  | (745,225) |
| 0-4  | (26,603)  |
| 1-2  | (926,138) |
| 1-3  | (826,546) |
| 1-4  | (857,231) |
| 2-3  | (375,323) |
| 2-4  | (422,711) |
| 3-4  | (10,275)  |



Then given  $G_2$  and a start-vertex of 0, the algorithm of Zhou and Gen generates the trace given in Table D.3. The generated spanning tree 0-3,1-4,2-3,3-4 with total weight vector = (1987,1054) is not dominated by any other generated in the trace, and is thus returned as an optimal solution. However, this spanning tree is dominated by the feasible spanning tree 0-2,1-2,2-3,3-4 (shown in Figure D.1) which has a weight vector = (1898,1021).

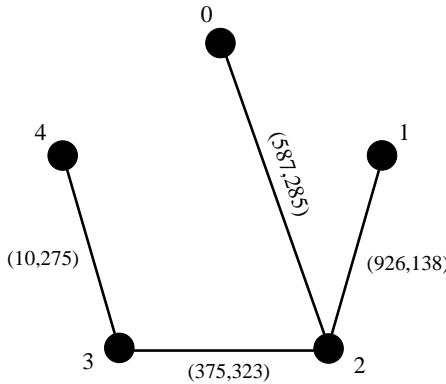


Figure D.1: The spanning tree 0-2,1-2,2-3,3-4 in  $G_2$ .

|                           |  |
|---------------------------|--|
| <b>0-1</b> (356,979) **   |  |
| <b>0-2</b> (587,285)      |  |
| <b>0-1</b> (943,1264) **  |  |
| <b>0-3</b> (1332,510) *   |  |
| <b>0-4</b> (613,888) *    |  |
| <b>1-2</b> (1513,423)     |  |
| <b>0-3</b> (2258,648)     |  |
| <b>0-4</b> (2284,1251) ** |  |
| <b>1-4</b> (3115,879) *   |  |
| <b>2-4</b> (2680,1359) ** |  |
| <b>3-4</b> (2268,923)     |  |
| <b>0-4</b> (1539,1026) *  |  |
| <b>1-3</b> (2339,969) **  |  |
| <b>1-4</b> (2370,654) **  |  |
| <b>2-3</b> (1888,746) *   |  |
| <b>2-4</b> (1935,1134) ** |  |
| <b>2-3</b> (962,608) *    |  |
| <b>2-4</b> (1009,996) **  |  |
| <b>0-3</b> (745,225)      |  |
| <b>0-1</b> (1101,1204) ** |  |
| <b>0-2</b> (1332,510) **  |  |
| <b>0-4</b> (771,828) **   |  |
| <b>1-3</b> (1571,771) **  |  |
| <b>2-3</b> (1120,548) **  |  |
| <b>3-4</b> (755,500)      |  |
| <b>0-1</b> (1111,1479) *  |  |
| <b>0-2</b> (1342,785)     |  |
| <b>0-1</b> (1698,1764) *  |  |
| <b>1-2</b> (2268,923) *   |  |
| <b>1-3</b> (2168,1331) *  |  |
| <b>1-4</b> (2199,1016) *  |  |
| <b>1-3</b> (1581,1046) ** |  |
| <b>1-4</b> (1612,731)     |  |
| <b>0-2</b> (2199,1016) *  |  |
| <b>1-2</b> (2538,869)     |  |
| <b>2-3</b> (1987,1054) R  |  |
| <b>2-4</b> (2034,1442) ** |  |
| <b>2-3</b> (1130,823)     |  |
| <b>0-1</b> (1486,1802) *  |  |
| <b>1-2</b> (2056,961)     |  |
| <b>1-3</b> (1956,1369) *  |  |
| <b>1-4</b> (1987,1054)    |  |
| <b>2-4</b> (1177,1211) ** |  |
| <b>0-4</b> (26,603)       |  |
| <b>0-1</b> (382,1582) **  |  |
| <b>0-2</b> (613,888) **   |  |
| <b>0-3</b> (771,828) *    |  |
| <b>1-4</b> (883,834) **   |  |
| <b>2-4</b> (448,1314) **  |  |
| <b>3-4</b> (36,878)       |  |
| <b>0-1</b> (392,1857)     |  |
| <b>0-2</b> (979,2142)     |  |
| <b>1-2</b> (1318,1995) *  |  |
| <b>2-3</b> (767,2180)     |  |
| <b>2-4</b> (814,2568) **  |  |
| <b>0-2</b> (623,1163)     |  |
| <b>0-1</b> (979,2142) R   |  |
| <b>1-2</b> (1549,1301)    |  |
| <b>1-3</b> (1449,1709) *  |  |
| <b>1-4</b> (1480,1394) *  |  |
| <b>1-3</b> (862,1424) **  |  |
| <b>1-4</b> (893,1109)     |  |
| <b>0-2</b> (1480,1394) *  |  |
| <b>1-2</b> (1819,1247)    |  |
| <b>2-3</b> (1268,1432)    |  |
| <b>2-4</b> (1315,1820) ** |  |
| <b>2-3</b> (411,1201)     |  |
| <b>0-1</b> (767,2180) R   |  |
| <b>1-2</b> (1337,1339)    |  |
| <b>1-3</b> (1237,1747)    |  |
| <b>1-4</b> (1268,1432) R  |  |
| <b>2-4</b> (458,1589) **  |  |

Table D.3: The trace of the algorithm of Zhou and Gen as it generates spanning trees of the graph  $G_2$ , using start-vertex  $v_1 = 0$ .

## References

- [ADG00] Ayman Abdalla, Narsingh Deo, and Pankaj Gupta. Random-tree diameter and the diameter-constrained MST. *Congressus Numerantium*, 144:161–182, 2000.
- [AKvL97] Emile H. L. Aarts, Jan H. M. Korst, and Peter J. M. van Laarhoven. Simulated annealing. In Aarts and Lenstra [AL97], pages 91–120.
- [AL96] M. B. Anderson and W. R. Lawrence. Launch conditions and aerodynamic data extraction by an elitist Pareto genetic algorithm. In *AIAA Atmospheric Flight Mechanics Conference*, San Diego, California, July 1996. AIAA Paper 96-3361.
- [AL97] Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997.
- [ALG96] M. B. Anderson, W. R. Lawrence, and G. A. Gebert. Using an elitist Pareto genetic algorithm for aerodynamic data extraction. In *4th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 1996. AIAA Paper 96-0514.
- [AM94] H. Asoh and H. Muehlenbein. On the mean convergence time of evolutionary algorithms without selection and mutation. In Y. Davidor, H.-P. Schwefel, and R. Maenner, editors, *Parallel Problem Solving from Nature (PPSN III)*, volume 866 of *Lecture Notes in Computer Science*, pages 88–97. Springer-Verlag, 1994.
- [BB91] Mark F. Bramlette and Eugene E. Bouchard. Genetic algorithms in parametric design of aircraft. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 109–123. Van Nostrand Reinhold, 1991.
- [BBC] BBC Television Centre, 80 Wood Lane, London W12 0TT.
- [BBS85] J. D. Barrow, S. P. Bhavsar, and D. H. Sonoda. Minimal spanning trees, filaments and galaxy clustering. *Mon. Not. R. astr. Soc.*, 216:17–35, 1985.
- [BC81] L. Bostock and S. Chandler. *Mathematics — the core course for A-level*. Stanley Thornes, 1981.
- [BDK95] Bruce Boldon, Narsingh Deo, and Nishit Kumar. Minimum-weight degree-constrained spanning tree problem: Heuristics and implementation on an SIMD parallel machine. Technical Report CS-TR-95-02, Department of Computer Science, University of Central Florida, Orlando, FL 32816, 1995.
- [Bel55] R. Bellman. Dynamic programming and multi-stage decision processes of stochastic type. In *Proceedings of the second symposium in linear programming*, volume 2, pages 229–250, Washington D.C., 1955. NBS and USAF.
- [Bet80] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, University of Michigan, Ann Arbor, MI, 1980. Computer and Communication Sciences, Dissertation Abstracts International 41(9), 3503B, University Microfilms Number 8106101.

- [BFM97] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.
- [BGK<sup>+</sup>95] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and William R. Stewart. Designing and reporting on computational experiments with heuristic methods. In *Proceedings of the International Conference on Metaheuristics for Optimization*, pages 1–17. Kluwer Publishing, 1995.
- [BM97] Thomas Bäck and Zbigniew Michalewicz. Test landscapes. In Bäck et al. [BFM97], page B2.7.4.
- [BMSC94] A. D. Belegundu, D. V. Murthy, R. R. Salagame, and E. W. Constants. Multiobjective optimization of laminated ceramic composites using genetic algorithms. In *Fifth AIAA/USA/NASA Symposium on Multidisciplinary Analysis and Optimization*, pages 1015–1022, Panama City, Florida, 1994. AIAA. Paper 84-4363-CP.
- [Bor26] O. Boruvka. O jistém problému minimálním. *Parca Moravské Přírodovědecké Společnosti*, 3:37–58, 1926. (In Czech.).
- [Bor00] Pedro Castro Borges. CHESS-changing horizon efficient set search: A simple principle for multiobjective optimization. *Journal of Heuristics*, 6(3):405–418, August 2000.
- [BPH98] Pedro Castro Borges and Michael Pilegaard Hansen. A basis for future successes in multiobjective combinatorial optimization. Technical Report IMM-REP-1998-8, Institute of Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, Denmark, March 1998.
- [BW97] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, Part 5, pages 231–240, London, June 1997. Springer Verlag London Limited. (Presented at the 2nd On-line World Conference on Soft Computing in Design and Manufacturing (WSC2)).
- [Cav70] D. J. Cavicchio. *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, 1970.
- [Cay89] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376–378, 1889.
- [CCH98] Carlos A. Coello Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. Using a new GA-based multiobjective optimization technique for the design of robot arms. *Robotica*, 16(4):401–414, July–August 1998.
- [CCR96] Emma Collingwood, David W. Corne, and Peter Ross. Useful diversity via multiploidy. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 810–813. IEEE Neural Networks Council, 1996.

- [CDG99] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization*. McGraw-Hill, London, UK, 1999.
- [CER95] Scott E. Cieniawski, J. W. Eheart, and S. Ranjithan. Using Genetic Algorithms to Solve a Multiobjective Groundwater Monitoring Problem. *Water Resources Research*, 31(2):399–409, February 1995.
- [CJ98] P. Czyzak and Andrzej Jaszkiewicz. Pareto simulated annealing - a meta-heuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, January 1998.
- [CJKO01] D. Corne, N. Jerram, J. Knowles, and M. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of GECCO-2001: Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kaufmann, 2001. (To appear).
- [CK00] David W. Corne and Joshua D. Knowles. The Pareto-envelope based selection algorithm for multiobjective optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 839–848, Berlin, 2000. Springer-Verlag.
- [CP99] Dragan Cvetković and Ian C. Parmee. Genetic algorithm-based multi-objective optimisation and conceptual engineering design. In *Congress on Evolutionary Computation – CEC99*, volume 1, pages 29–36, Washington D.C., USA, 1999. IEEE.
- [CT01] Carlos A. Coello Coello and Gregorio Toscano Pulido. A micro-genetic algorithm for multiobjective optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.
- [DA95] Kalyanmoy Deb and R Agrawal. Simulated binary crossover for continuous search spaces. *Complex Systems*, 9:115–148, 1995.
- [DAPM00a] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.
- [DAPM00b] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. Fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Schoenauer et al. [SDR<sup>+</sup>00], pages 849–858.
- [Dav91a] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.
- [Dav91b] Lawrence Davis. Order-based genetic algorithms and the graph coloring problem. In *Handbook of Genetic Algorithms*, chapter 6, pages 72–90. Van Nostrand Reinhold, 1991.

- [DBT00] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16:851–871, 2000.
- [DDB01] Nicole Drechsler, Rolf Drechsler, and Bernd Becker. Multi-objective optimisation based on relation *favour*. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 154–166. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.
- [Deb97] Kalyanmoy Deb. Deceptive landscapes. In Bäck et al. [BFM97], page B2.7.1.
- [Deb98] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.
- [Deb99a] Kalyanmoy Deb. Evolutionary algorithms for multi-criterion optimization in engineering design. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135–161. John Wiley & Sons, Ltd, Chichester, UK, 1999.
- [Deb99b] Kalyanmoy Deb. Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions. KanGAL report 99002, Indian Institute of Technology, Kanpur, India, 1999.
- [DeJ75] K. A. DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975. Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381.
- [DG92] Kalyanmoy Deb and David E. Goldberg. Analyzing deception in trap functions. In *FOGA-92, Foundations of Genetic Algorithms*, Vail, Colorado, 24–29 July 1992. Email: deb@iitk.ernet.in, goldberg@vmd.cso.uiuc.edu.
- [DG00] Kalyanmoy Deb and Tushar Goel. Controlled elitist non-dominated sorting genetic algorithm for better convergence. In Schoenauer et al. [SDR<sup>+</sup>00], pages 67–81.
- [DG01] Kalyanmoy Deb and Tushar Goel. A hybrid multi-objective evolutionary approach to engineering shape design. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 385–399, Berlin, 2001. Springer-Verlag.
- [DPT96] D. Duvivier, Philippe Preux, and E.-G. Talbi. Climbing up NP-hard hills. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 574–583, Berlin, 1996. Springer.
- [DTLZ01] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 2001001, KanGAL, Kanpur, 2001.

- [DW91] Rajarshi Das and Darrell Whitley. The only challenging problems are deceptive: Global search by solving order-1 hyperplanes. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 166–173, San Mateo, California, 1991. Morgan Kaufmann Publishers.
- [EG00] Matthias Ehrgott and Xavier Gandibleux. An annotated bibliography of multi-objective combinatorial optimization. Technical Report 62/2000, Fachbereich Mathematik, Universitat Kaiserslautern, Kaiserslautern, Germany, 2000.
- [Ehr00] Matthias Ehrgott. *Multicriteria Optimization*. Number 491 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 2000.
- [EK97] Matthias Ehrgott and Kathrin Klamroth. Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97:159–166, 1997.
- [ES96] Reuven Elbaum and Moshe Sidi. Topological design of local-area networks using genetic algorithms. *IEEE/ACM Transactions on Networking*, 4:766–778, 1996.
- [Esh97] Larry J. Eshelman. Genetic algorithms. In Bäck et al. [BFM97], page B1.2.
- [FB69] L. J. Fogel and G. H. Burgin. Competitive goal-seeking through evolutionary programming. Technical Report Final Report Contract AF19(628)-5927, Air Force Cambridge Research Laboratories, 1969.
- [FF93] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.
- [FF94] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.
- [FF95] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.
- [FF96] Carlos M. Fonseca and Peter J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 584–593, Berlin, Germany, September 1996. Springer-Verlag.
- [FKK<sup>+</sup>97] S.P. Fekete, S. Khuller, M. Klemmstein, B. Raghavachari, and N. Young. A network-flow technique for finding low-weight bounded-degree spanning trees. *Journal of Algorithms*, 24:310–324, 1997.
- [Fog64] L. J. Fogel. *On the organization of intellect*. PhD thesis, University of California, 1964.

- [Fog94] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1994.
- [Fou85] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum, 1985.
- [FR95] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
- [GDH92] David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
- [Gen00] Mitsuo Gen. Network design by genetic algorithms. In *2000 Genetic and Evolutionary Computation Conference Tutorial Program*, pages 410–470, 2000.
- [GETA99] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.
- [GF98] Xavier Gandibleux and Arnaud Fréville. Tabu search based procedure for solving the 0/1 multiobjective knapsack problem: The two objective case. *Journal of Heuristics*, Accepted 1998. (to appear).
- [GGST86] H.N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected graphs. *Combinatorica*, 6:109–122, 1986.
- [GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP Completeness*. W. H. Freeman and Company, San Francisco, 1979.
- [GJRR01] Jens Gottlieb, Bryant A. Julstrom, Günther R. Raidl, and Franz Rothlauf. Prüfer numbers: A poor representation of spanning trees for evolutionary search. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 343–350, 2001.
- [GL97] Fred Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.
- [GMF96] Xavier Gandibleux, Nazik Mezdaoui, and Arnaud Fréville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In R. Caballero and R. Steuer, editors, *Proceedings volume of MOPGP'96*, pages 291–300, Berlin, 1996. Springer-Verlag.
- [GMF97] Xavier Gandibleux, Nazik Mezdaoui, and Arnaud Fréville. A tabu search procedure to solve combinatorial optimisation problems. In Rafael Caballero, Francisco Ruiz, and Ralph E. Steuer, editors, *Advances in Multiple Objective and Goal Programming*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 291–300. Springer-Verlag, 1997.

- [Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
- [GR87] David E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.
- [GR01] Thomas Gaube and Franz Rothlauf. The link and node biased encoding revisited: Bias and adjustment of parameters. In Egbert J. W. Boers, Jens Gottlieb, Pier Luca Lanzi, Robert E. Smith, Stefano Cagnoni, Emma Hart, Günther R. Raidl, and H. Tijink, editors, *EvoWorkshops*, volume 2037 of *Lecture Notes in Computer Science*. Springer, 2001.
- [Gre93] John J. Grefenstette. Deception considered harmful. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 75–91, San Mateo, 1993. Morgan Kaufmann.
- [GW00] Justin Gan and Kevin Warwick. A variable radius niching technique for speciation in genetic algorithms. In Darrell Whitley *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 96–103, San Francisco, CA, 2000. Morgan Kaufmann.
- [Han97a] Michael Pilegaard Hansen. Generating a diversity of good solutions to a practical combinatorial problem using vectorized simulated annealing. Technical report, Institute of Mathematical Modelling, Technical University of Denmark, August 1997. Working Paper.
- [Han97b] Michael Pilegaard Hansen. Tabu search in multiobjective optimisation : MOTS. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97)*, Cape Town, South Africa, January 1997.
- [Han99] Thomas Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, 117(3):553–564, 1999.
- [HG95] Jeffrey Horn and David E. Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 243–269. Morgan Kaufmann, San Francisco, CA, 1995.
- [HGD92] Jeffrey Horn, David Goldberg, and Kalyan Deb. Long path problems for mutation-based algorithms. Technical Report 92011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana IL, 1992.
- [HJ98] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, March 1998.
- [HL92] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

- [HN93] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective optimization using the niched Pareto genetic algorithm. Technical Report IlliGAI Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.
- [HNG94] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.
- [Hol75] John H. Holland. *Adpatation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
- [Hor97a] Jeffrey Horn. Multicriterion decision making. In Thomas Bäck, David Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, pages F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press, 1997.
- [Hor97b] Jeffrey Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.
- [HR94] Horst W. Hamacher and Günter Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.
- [HTdW97] Alain Hertz, Eric Taillard, and Dominique de Werra. Tabu search. In Aarts and Lenstra [AL97], pages 121–136.
- [IM96] Hisao Ishibuchi and Tadahiko Murata. Multi-objective genetic local search algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.
- [IMT97] Hisao Ishibuchi, Tadahiko Murata, and Shigemitsu Tomoioka. Effectiveness of genetic local search algorithms. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 505–512. Morgan Kaufmann, 1997.
- [Jar30] V. Jarník. O jistém problému minimálním. *Acta Societatis Scientiarum Natur. Moravicae*, 6:57–63, 1930.
- [Jas98] Andrzej Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznán University of Technology, 1998.
- [Jas99] Andrzej Jaszkiewicz. Improving performance of genetic local search by changing local search space topology. *Foundations of Computing and Decision Sciences*, 24:77–84, 1999.
- [Jas00] Andrzej Jaszkiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem. a comparative experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, July 2000.

- [JFS97] Kenneth De Jong, David B. Fogel, and Hans-Paul Schwefel. A history of evolutionary computation. In Bäck et al. [BFM97], page A2.3.
- [JLM96] Cameron L. Jones, Greg T. Lonergan, and David E. Mainwaring. Minimal spanning tree analysis of fungal spore spatial patterns. *Bioimages (Japan)*, 4:91–98, October 1996.
- [JW94] Ari Juels and Martin Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Technical Report CSD-94-834, Department of Computer Science, University of California at Berkeley, USA, 18 1994.
- [Kau89] Stuart A. Kauffman. Adaptation on rugged fitness landscapes. In D. Stein, editor, *SFI Studies in the Sciences of Complexity, Lecture Volume 1*, pages 527–618. Addison Wesley, 1989.
- [KC99a] J. D. Knowles and D. W. Corne. Local search, multiobjective optimization and the Pareto archived evolution strategy. In Bob McKay, Xin Yao, Ruhul Sarker, Yasuhiro Tsujimura, Akira Namateme, and Mitsuo Gen, editors, *Proceedings of Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 209–216, Ashikaga, Japan, November 1999. University of New South Wales and Ashikaga Institute of Technology.
- [KC99b] Joshua D. Knowles and David W. Corne. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.
- [KC00a] Joshua Knowles and David Corne. A new evolutionary approach to the degree-constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4(2):125–134, July 2000.
- [KC00b] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
- [KC00c] Joshua D. Knowles and David W. Corne. M-PAES: A memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*, pages 325–332, Piscataway, NJ, 2000. IEEE Press.
- [KC01a] Joshua D. Knowles and David W. Corne. A comparison of encodings and algorithms for multiobjective minimum spanning tree problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC'01)*, pages 544–551. IEEE Press, 2001.
- [KC01b] Joshua D. Knowles and David W. Corne. Enumeration of Pareto optimal multi-criteria spanning trees — a proof of the incorrectness of Zhou and Gen’s proposed algorithm. *European Journal of Operational Research*, 2001. (In press).
- [Kea96] A. J. Keane. A brief comparison of some evolutionary optimization methods. In V. J. Rayward-Smith and I. H. Osman, editors, *Modern Heuristic Search Methods*. John Wiley and Sons Ltd., 1996.

- [KES99] M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. Technical report, CSIRO Mathematical and Information Sciences, Clayton, Australia, 1999. (Revised for re-submission to Journal of Heuristics, 25 Oct 1999).
- [Kin95] V. King. A simpler minimum spanning tree verification algorithm. In *Proc. 4th Worksh. Algorithms and Data Structures*, pages 440–448, 1995.
- [KKT95] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm for finding a minimum spanning tree. *Journal of the ACM*, 42:321–328, 1995.
- [KmV83] S. Kirkpatrick, C. D. Gelatt, Jnr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
- [Kru56] J. B. Kruskal. On the shortest spanning tree of graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.
- [KRY96] Samir Khuller, Balaji Raghavachari, and Neal Young. Low-degree spanning trees of small weight. *SIAM Journal of Computing*, 25(2):355–368, 1996.
- [Kur91] Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.
- [KWC01] Joshua D. Knowles, Richard A. Watson, and David W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *Evolutionary Multi-Criterion Optimization : first international conference; proceedings / EMO 2001*, volume 1993 of *LNCS*, pages 269–283. Springer, 2001.
- [LB99] Yu Li and Youcef Bouchebaba. A new genetic algorithm for the optimal communication spanning tree problem. Technical Report 99-09, LaRIA, Université de Picardie Jules Verne, France, 1999.
- [Li00] Yu Li. An effective implementation of a GA using a direct tree representation for constrained minimum spanning tree problems. Technical Report 2000-15, LaRIA, Université de Picardie Jules Verne, France, 2000.
- [LMS<sup>+</sup>98] Jussi Lahtinen, Petri Myllymki, Tomi Silander, Henry Tirri, and Hannes Wettig. An empirical evaluation of stochastic search methods in real-world telecommunication domains. In P Koikkalainen and S. Puuronen, editors, *Human and Artificial Information Processing: Proceedings of the 8th Finnish Artificial Intelligence Conference*. Finnish Artificial Intelligence Society, 1998.
- [LMST96] Jussi Lahtinen, Petri Myllymki, Tomi Silander, and Henry Tirri. Empirical comparison of stochastic algorithms in a graph optimization problem. In *Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications*, pages 45–59, 1996.

- [LP98] W. B. Langdon and R. Poli. Why ants are hard. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 193–201, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.
- [LR93] Sushil J. Louis and Gregory J. E. Rawlins. Pareto optimality, GA-easiness and deception. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-5)*, pages 118–123, San Mateo, CA, 1993. Morgan Kaufmann.
- [LTDZ01] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms. Technical Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.
- [LZT00a] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In Schoenauer et al. [SDR<sup>+</sup>00], pages 181–195.
- [LZT00b] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *2000 Congress on Evolutionary Computation*, volume 1, pages 46–53, Piscataway, New Jersey, July 2000. IEEE Service Center.
- [Mah95] Samir W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, May 1995. IlliGAL Report 95001.
- [MDS00] Filippo Menczer, Melania Degeratu, and W. Nick Street. Efficient and scalable Pareto optimization by evolutionary local selection algorithms. *Evolutionary Computation*, 8(2):223–247, Summer 2000.
- [MF98] Peter Merz and Bernd Freisleben. On the effectiveness of evolutionary search in high-dimensional NK-landscapes. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC98)*, pages 741–745. IEEE Press, 1998.
- [MF99] P. Merz and B. Freisleben. Fitness landscapes and memetic algorithm design. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 245–260. McGraw-Hill, 1999.
- [MFH91] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems*, pages 245–254, Paris, France, 11-13 December 1991. MIT Press, Cambridge, MA.

- [MG92] Samir Mahfoud and David E. Goldberg. Parallel recombinative simulated annealing: A genetic algorithm. IlliGAL Report 92002, University of Illinois at Urbana-Champaign, April 1992.
- [MHF94] Melanie Mitchell, John H. Holland, and Stephanie Forrest. When will a genetic algorithm outperform hill climbing? In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*. Morgan Kauffman, San Francisco, CA, 1994.
- [Mic96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.
- [Mie01] Kaisa Miettinen. Some methods for nonlinear multi-objective optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 1–20, Berlin, 2001. Springer-Verlag.
- [Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.
- [MIT96] Tadahiko Murata, Hisao Ishibuchi, and H. Tanaka. Multi-objective genetic algorithm and its application to flowshop scheduling. *Computers and Industrial Engineering Journal*, 30(4):957–968, September 1996.
- [MJ96] Brad L. Miller and Shaw M. J. Genetic algorithms with dynamic niche sharing for multimodal function optimization. Technical Report 95010, IlliGAL, University of Illinois, 1996.
- [Mos99] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, 1999.
- [MRR<sup>+</sup>53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
- [MS96] Jason W. Mann and George D. Smith. A comparison of heuristics for telecommunications traffic routing. In V. J. Rayward-Smith and I. H. Osman, editors, *Modern Heuristic Search Methods*. John Wiley and Sons Ltd., 1996.
- [MT90] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley and Sons Ltd., Chichester, UK, 1990.
- [Nes97] Jaroslav Nesetril. A few remarks on the history of the MST-problem. *Archivum Mathematicum*, 33:15–22, 1997.
- [NH80] S.C. Narula and C.A. Ho. Degree-constrained minimum spanning tree. *Computers and Operations Research*, 7(4):39–49, 1980.

- [NW01] Jason Noble and Richard A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 493–500, San Francisco, California, 2001. Morgan Kaufmann Publishers.
- [Oat00] Martin Oates. Scalability of EC based web server load balancing. The SAB/PPSN Workshops Booklet, September 2000.
- [OC98] Martin J. Oates and David W. Corne. Investigating evolutionary approaches to adaptive database management against various quality of service metrics. In T. Bäck, M. Schoenauer, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature V*, pages 775–784. Springer, 1998.
- [OC00] Martin J. Oates and David Corne. Exploring evolutionary approaches to distributed database management. In David Corne, Martin J. Oates, and George D. Smith, editors, *Telecommunications Optimization: Heuristic and Adaptive Techniques*, pages 235–264. John Wiley and Sons, Ltd, 2000.
- [PCWB00] Ian C. Parmee, Dragan Cvetković, Andrew H. Watson, and Christopher R. Bonham. Multiobjective satisfaction within an interactive evolutionary design environment. *Evolutionary Computation*, 8(2):197–222, Summer 2000.
- [Pea84] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley, 1984.
- [PK97] Charles C. Palmer and Aaron Kershenbaum. Representing trees in genetic algorithms. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, pages G1.3:1–8. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.
- [PM98] Geoffrey T. Parks and I. Miller. Selective breeding in a multiobjective genetic algorithm. In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature — PPSN V*, pages 250–259, Amsterdam, Holland, 1998. Springer-Verlag.
- [Por97] V William Porto. Evolutionary programming. In Bäck et al. [BFM97], page B1.4.
- [Pri57] R.C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, 36:1389–1401, 1957.
- [Prü18] H. Prüfer. Neuer beweis eines satzes über permutationen. *Arch. Math. Phys.*, 27:742–744, 1918.
- [RA00] Günter Rudolph and Alexandru Agapie. Convergence properties of some multi-objective evolutionary algorithms. In *Proceedings of the 2000 Conference on Evolutionary Computation*, volume 2, pages 1010–1016, Piscataway, New Jersey, July 2000. IEEE Press.

- [Rad97] Nicholas J. Radcliffe. Schema processing. In Bäck et al. [BFM97], page B2.5.
- [Rai00] Günther R. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*, pages 104–111, Piscataway, NJ, 2000. IEEE Press.
- [RASG98] R. M. Ramos, S. Alonso, J. Sicilia, and C. González. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111:617–628, 1998.
- [Rec65] Ingo Rechenberg. Cybernetic solution path of an experimental problem, 1965. Library Translation 1122, Royal Aircraft Establishment, Farnborough, UK.
- [Ree96] Colin R. Reeves. Modern heuristic techniques. In V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, editors, *Modern Heuristic Search Methods*, chapter 1, pages 1–26. John Wiley and Sons Ltd., 1996.
- [RJ00] Günther R. Raidl and B. A. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In Janice Carroll, Ernesto Damiani, Hisham Haddad, and Dave Oppenheim, editors, *Proceedings of the 15th ACM Symposium on Applied Computing*, volume 1, pages 440–445. ACM Press, 2000.
- [RMR<sup>+</sup>93] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt. Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings of 25th Annual ACM STOC*, pages 438–447, 1993.
- [Rob01] “Dr Rob”. Probability of a sum on multiple dice. URL: [www.drmath.com/dr.math/problems/regan.3.26.01.html](http://www.drmath.com/dr.math/problems/regan.3.26.01.html), March 2001.
- [RS01] Celso C. Ribeiro and Mauricio C. Souza. Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 2001. (accepted for publication).
- [Rud97] Günter Rudolph. Evolution strategies. In Bäck et al. [BFM97], page B1.3.
- [Rud98a] Günter Rudolph. Evolutionary search for minimal elements in partially ordered finite sets. In *Evolutionary Programming VII - Proceedings of the Seventh Annual Conference on Evolutionary Programming (EP-98)*, Cambridge MA, 1998. MIT Press.
- [Rud98b] Günter Rudolph. On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, Piscataway, New Jersey, 1998. IEEE Press.
- [SB00] Ricardo Sznit and Amnon Barak. Evolution strategies for a parallel multi-objective genetic algorithm. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 227–234, San Francisco, California, 2000. Morgan Kaufmann.

- [Sch84] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.
- [Sch85] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.
- [Sch95] Jason R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master’s thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.
- [Sch97] Hans-Paul Schwefel. Advantages (and disadvantages) of evolutionary computation over other approaches. In Bäck et al. [BFM97], page A1.3.
- [SD93] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.
- [SD94] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.
- [SD95] N. Srinivas and Kalyanmoy Deb. Comparative study of vector evaluated GA and NSGA applied to multiobjective optimization. In P. K. Roy and S. D. Mehta, editors, *Proceedings of the Symposium on Genetic Algorithms*, pages 83–90, 1995.
- [SDR<sup>+</sup>00] Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel, editors. *Parallel Problem Solving from Nature - PPSN VI : 6th International Conference Proceedings*. Number 1917 in LNCS. Springer, 2000.
- [SDRW00] George D. Smith, Jason C. W. Debuse, Michael D. Ryan, and Iain M. Whittley. An effective genetic algorithm for the fixed channel assignment problem. In David Corne, Martin J. Oates, and George D. Smith, editors, *Telecommunications Optimization: Heuristic and Adaptive Techniques*, pages 357–371. John Wiley and Sons, Ltd, 2000.
- [Ser94] Paolo Serafini. Simulated annealing for multiple objective optimization problems. In G.H. Tzeng, H.F. Wang, U.P. Wen, and P.L. Yu, editors, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*, volume 1, pages 283–294, Berlin, 1994. Springer-Verlag.
- [Sha00] Oliver John Sharpe. *Towards a Rational Methodology for Using Evolutionary Search Algorithms*. PhD thesis, COGS, University of Sussex, 2000.
- [Sin99] Mark C. Sinclair. Evolutionary telecommuniocations: A summary. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Program*, pages 209–212, 1999.

- [SS96a] M. Sun and R. E. Steuer. Interquad: An interactive quad tree based procedure for solving the discrete alternative multiple criteria problem. *European Journal of Operational Research*, 89:462–472, 1996.
- [SS96b] M. Sun and R. E. Steuer. Quad trees and linear list for identifying nondominated criterion vectors. *INFORM Journal on Computing*, 8:367–375, 1996.
- [SV85] M. Savelsbergh and T. Volgenant. Edge exchanges in the degree-constrained minimum spanning tree problem. *Computers and Operations Research*, 12(4):341–348, 1985.
- [Sys89] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, 1989.
- [Tan89] R. Tanese. *Distributed Genetic Algorithms for Function Optimization*. PhD thesis, University of Michigan, 1989. Computer Science and Engineering.
- [TMA95] Hisashi Tamaki, M. Mori, and M. Araki. Generation of a set of Pareto-optimal solutions by genetic algorithms. *Transactions of the Society of Instrument and Control Engineers*, 31(8):1185–1192, 1995.
- [TS97] David S. Todd and Pratyush Sen. A multiple criteria genetic algorithm for containership loading. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 674–681, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.
- [TYK96] Y. Takada, M. Yamamura, and S. Kobayashi. An approach to portfolio selection problems using multi-objective genetic algorithms. In *Proceedings of the 23rd Symposium on Intelligent Systems*, pages 103–108, 1996.
- [Tzb99] Jürgen Teich, Eckart Zitzler, and Shuvra S. Bhattacharyya. 3D exploration of software schedules for DSP algorithms. In *7th International Workshop on Hardware/Software Codesign (CODES'99)*, pages 168–172, May 1999.
- [UTFT99] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.
- [Vel99] 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.
- [VL99] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm test suites. In Janice Carroll, Hisham Haddad, Dave Oppenheim, Barrett Bryant, and Gary B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, San Antonio, Texas, 1999. ACM.
- [VL00] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

- [Wal23] J. L. Walsh. A closed set of orthogonal functions. *American Journal of Mathematics*, 55:5–24, 1923.
- [Whi89] Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufman.
- [Whi91] Darrell Whitley. Fundamental principles of deception in genetic search. Technical Report CS-91-101, Department of Computer Science, Colorado State University, Fort Collins, February 1991.
- [Whi94] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.
- [Whi99] D. Whitley. A free lunch proof for gray versus binary encodings. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 726–733, Orlando, Florida, USA, 13–17 1999. Morgan Kaufmann.
- [WHP98] Richard A. Watson, Gregory S. Hornby, and Jordan B. Pollack. Modeling building-block interdependency. In *Parallel Problem Solving from Nature - PPSN V*, pages 97–106. Springer-Verlag, 1998.
- [Wil91] S. W. Wilson. GA-easy does not imply steepest-ascent optimizable. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 85–89. Morgan Kaufmann, 1991.
- [WM97] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.
- [WP00] Richard A. Watson and Jordan B. Pollack. Hierarchically-consistent test problems for genetic algorithms. In Peter Angeline, Zbigniew Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of 1999 Congress on Evolutionary Computation (CEC 99)*, pages 1406–1413. IEEE Press, 2000.
- [WW90] R.J. Wilson and J.J. Watkins. *Graphs: an introductory approach: a first course in discrete mathematics*. John Wiley and Sons. Inc., 1990.
- [YA94] Y. Yoshida and N. Adachi. A diploid genetic algorithm for preserving population diversity — pseudo-meiosis GA. In Davidor, Schwefel, and Manner, editors, *Parallel Problem Solving from Nature III*, pages 36–45. Springer, 1994.
- [YI96] M. Yagiura and T. Ibaraki. Metaheuristics as robust and simple optimization tools. In *Proceedings 1996 IEEE Conference on Evolutionary Computation*, pages 541–546. IEEE Press, 1996.
- [ZDT00] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

- [ZG97] Gengui Zhou and Mitsuo Gen. Evolutionary computation on multicriteria production process planning problem. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 419–424, Piscataway, New Jersey, April 1997. IEEE Press.
- [ZG99] Gengui Zhou and Mitsuo Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114(1), April 1999.
- [Zit99] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
- [ZT98a] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.
- [ZT98b] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.
- [ZT99] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.