Babes-Bolyai University of Cluj-Napoca, Romania
Faculty of Mathematics and Computer Science
Department of Computer Science

Catalin Stoean

New Evolutionary Techniques in Multi-modal Optimization

PhD Dissertation

Supervisor: D. Dumitrescu

Thesis commitee:

Prof. Dr. Thomas Bartz-Beielstein, University of Cologne, Germany
Prof. Dr. Henri Luchian, Al. I. Cuza University, Iasi, Romania
Prof. Dr. Doina Tatar, Babes-Bolyai University, Cluj-Napoca, Romania

February 2008

Dedication

To my beloved wife, Ruxandra.

Acknowledgements

I would like to express my deep gratitude to the many people I have collaborated with for almost any scientific progress that has been made. The most consistent part of this work has been submitted to computer science journals or presented at related conferences. The other people that brought important contributions to these articles are: D. Dumitrescu, Mike Preuss, Elia El-Darzi and Ruxandra Stoean.

The first person I would like to express my deepest gratitude is my PhD supervisor, Professor D. Dumitrescu. Ever since he accepted me as his PhD student, after being severely tested in May 2004 with the occasion of my presentation at the International Conference on Computers and Communications in Oradea, he has been providing me with constant support, wise advice and warm friendship. It is not only the magic and exciting world of evolutionary computation he introduced me to, but he also taught me how to be a true scientist.

I am very thankful to Mike Preuss, for the inspiring collaboration, in spite of the large distance separating us most of the time. I wish to thank DAAD for the provided research grant at the University of Dortmund: The time spent there had been very valuable to me, especially for completing the work included in chapter 7 and for the contouring of the entire dissertation. I am also grateful to Mike Preuss for taking good care of me during my entire stay in Dortmund.

I would like to express my sincere gratitude to Professor Günther Rudolph and Professor Hans-Paul Schwefel for they provided me with the opportunity to research beside an important evolutionary computation group within the Chair of Algorithm Engineering at the University of Dortmund. While there, I also had the chance to meet Professor Thomas Bartz-Beielstein who helped me in improving the experimentation through the use of his automatic tuning mechanism. I use this opportunity to thank all the people from the chair for the friendly atmosphere they offered and without which the completion of this work would not have been possible.

I wish to thank Dr. Elia El-Darzi for the valuable discussions and for supplying his useful comments that contributed to the improvement of the papers.

I would like to show my appreciation towards the support received from Professor Nicolae Tandareanu, from my home University of Craiova. In all these years I have been studying for PhD, he helped me with useful advice and fatherly took good care that I had all the conditions for research.

Many thanks go to the staff from the Computer Science Department of the Babes-Bolyai University of Cluj-Napoca who, during my exams and reports, have constantly helped me with suggestions and encouragements.

Nothing would have been achieved without the love and cooperation of my wife and colleague, Ruxandra Stoean. Last but not least, I want to say that no accomplishment would have been possible without the hearty support of my parents, Magdalena and Ion Stoean.

Published work associated with thesis

Articles in international journals

Journals indexed by ISI

1. **Catalin Stoean**, Mike Preuss, Ruxandra Stoean, D. Dumitrescu, Multimodal Optimization by means of a Topological Species Conservation Algorithm, IEEE Transactions on Evolutionary Computation (2006 Impact Factor: 3.77), IEEE Intelligence Computational Society, submitted, 2008.

2. D. Dumitrescu, **Catalin Stoean**, Genetic Chromodynamics for Multimodal Optimization. Application to Function Optimization, Clustering and Classification, Information Sciences Journal (2006 Impact Factor: 1.003), Elsevier, submitted, 2007.

3. Ruxandra Stoean, Mike Preuss, **Catalin Stoean**, Elia El-Darzi, D. Dumitrescu, An Evolutionary Resemblant to Support Vector Machines for Classification and Regression, Journal of the Operational Research Society (2006 Impact Factor: 0.597), Palgrave Macmillan, submitted, 2007.

4. **Catalin Stoean**, D. Dumitrescu, Mike Preuss, Ruxandra Stoean, Cooperative Coevolution as a Paradigm for Classification, Journal of Universal Computer Science (2006 Impact Factor: 0.34), Springer-Verlag, in press, 2008.

5. Ruxandra Stoean, D. Dumitrescu, Mike Preuss, **Catalin Stoean**, Evolutionary Support Vector Machines for Classification with Multiple Outcomes, Journal of Universal Computer Science (2006 Impact Factor: 0.34), Springer-Verlag, in press, 2008.

6. Ruxandra Stoean, **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Forecasting Soybean Diseases from Symptoms by Means of Evolutionary Support Vector Machines, Phytologia Balcanica, Vol. 12, No. 3, pp. 345 - 350, Sofia, Bulgaria, 2006.

Journals indexed by INSPEC(IEE)

7. D. Dumitrescu, **Catalin Stoean**, Genetic Chromodynamics Metaheuristic for Multimodal Optimization, WSEAS Transactions on Information Science and Application, WSEAS Press,

Zoran Bojkovic (Ed.), Issue 8, Volume 3, pp. 1444-1452, ISSN 1790-0832, 2006.

Articles in national journals

Journals indexed by CNCSIS

8. Ruxandra Stoean, **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Evolutionary Detection of Separating Hyperplanes in E-mail Classification, Acta Cibiniensis, Vol. LV Technical series, University "Lucian Blaga" Sibiu Press, pp. 41-46, 2007.

9. **Catalin Stoean**, D. Dumitrescu, Elitist Generational Genetic Chromodynamics as a Learning Classifier System, Annals of University of Craiova, Mathematics and Computer Science Series, Vol. 33, pp. 132-140, ISSN 1223-6934, 2006.

10. Ruxandra Stoean, D. Dumitrescu, **Catalin Stoean**, Nonlinear Evolutionary Support Vector Machines. Application to Classification, Studia Babes-Bolyai, Seria Informatica, Vol. LI, No. 1, pp. 3-12, 2006.

11. **Catalin Stoean**, D. Dumitrescu, On Improving Classification Accuracy for Diabetes Diagnosis by Evolving Weights, Scientific Bulletin, University of Pitesti, Mathematics and Computer Science Series, Issue 11, pp. 67-74, 2005.

12. **Catalin Stoean**, Ruxandra Gorunescu, Mike Preuss, D. Dumitrescu, An Evolutionary Learning Classifier System Applied to Text Categorization, Annals of West University of Timisoara, Mathematics and Computer Science Series, vol. XLII, special issue 1, pp. 265-278, 2004.

13. **Catalin Stoean**, On Building Hierarchical Document Categories Using Equivalence Classes in an Information System, Annals of University of Craiova, Mathematics and Computer Science Series, pp. 177 - 183, 2004.

Papers at international conferences

Conferences indexed by IEEE

14. Ruxandra Stoean, Mike Preuss, **Catalin Stoean**, D. Dumitrescu, Concerning the Potential of Evolutionary Support Vector Machines, The IEEE Congress on Evolutionary Computation - CEC 2007, Singapore, pp. 1436 - 1443, 2007.

15. **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Ruxandra Stoean, Cooperative Evolution of Rules for Classification, IEEE Postproceedings SYNASC 2006, IEEE Press, Lisa O'Conner (Ed.), Los Alamitos, CA, USA, ISBN 0-7695-2740-X, pp. 317-322, 2006.

16. Ruxandra Stoean, Mike Preuss, D. Dumitrescu, **Catalin Stoean**, Evolutionary Support Vector Regression Machines, IEEE Postproceedings SYNASC 2006, IEEE Press, Lisa O'Conner (Ed.), Los Alamitos, CA, USA, ISBN 0-7695-2740-X, pp. 330-335, 2006.

17. Ruxandra Stoean, **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Evolutionary Support Vector Machines for Spam Filtering, RoEduNet IEEE International Conference, Sibiu, Romania, pp. 261-266, 2006.

18. Ruxandra Stoean, **Catalin Stoean**, Mike Preuss, Elia El-Darzi, D. Dumitrescu, Evolutionary Support Vector Machines for Diabetes Mellitus Diagnosis, Proceedings 3rd International IEEE Conference on Intelligent Systems - IS 2006, University of Westminster, London, pp. 182-187, ISBN 1-4244-0196-8, 2006.

19. **Catalin Stoean**, Mike Preuss, Ruxandra Gorunescu, D. Dumitrescu, Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization, The 2005 IEEE Congress on Evolutionary Computation - CEC 2005, Edinburgh, UK, Vol. 2, pp. 1839 - 1846, ISBN 0-7803-9363-5, 2005.

Conferences indexed by ACM

20. **Catalin Stoean**, Mike Preuss, Ruxandra Stoean, D. Dumitrescu, Disburdening the Species Conservation Evolutionary Algorithm of Arguing with Radii, The ACM Genetic and Evolutionary Computation Conference - GECCO 2007, London, UK, pp. 1420 - 1427, 2007.

21. **Catalin Stoean**, Ruxandra Stoean, Elia El-Darzi, Breast Cancer Diagnosis by Means of Cooperative Coevolution, Proceedings of the Third ACM International Conference on Intelligent Computing and Information Systems - ICICIS 2007, Police Press, Cairo, pp. 493-497, ISBN 977-237-172-3, 2007.

ISI proceedings of the conferences above

(a) **Catalin Stoean**, Mike Preuss, Ruxandra Stoean, D. Dumitrescu, Disburdening the Species Conservation Evolutionary Algorithm of Arguing with Radii, The ACM Genetic and Evolutionary Computation Conference - GECCO 2007, London, UK, pp. 1420 - 1427, 2007.

(b) Ruxandra Stoean, Mike Preuss, **Catalin Stoean**, D. Dumitrescu, Concerning the Potential of Evolutionary Support Vector Machines, The IEEE Congress on Evolutionary Computation - CEC 2007, Singapore, pp. 1436 - 1443, 2007.

(c) **Catalin Stoean**, Mike Preuss, Ruxandra Gorunescu, D. Dumitrescu, Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization, The 2005 IEEE Congress on Evolutionary Computation - CEC 2005, Edinburgh, UK, Vol. 2, pp. 1839 - 1846, ISBN 0-7803-9363-5, 2005.

Other International Conferences

22. **Catalin Stoean**, Ruxandra Stoean, Mike Preuss, D. Dumitrescu, Competitive Coevolution for Classification, Proceedings of the 7th International Conference on Artificial Intelligence and Digital Communications - AIDC 2007, pp. 28-39, 2007.

23. D. Dumitrescu, **Catalin Stoean**, Ruxandra Stoean, Genetic Chromodynamics for the Job Shop Scheduling Problem, International Conference Knowledge Engineering Principles and Techniques (KEPT 2007), Studia Univ. Babes - Bolyai, Informatica, Special Issue, pp. 153-160, 2007.

24. **Catalin Stoean**, Ruxandra Stoean, Elia El-Darzi, Breast Cancer Diagnosis by Means of Cooperative Coevolution (2), Proceedings First International Conference on Medical Informatics - ICMI 2007, Misr University for Science and Technology, March 19th, Cairo, Egypt, pp. 19-24, 2007.

25. **Catalin Stoean**, Ruxandra Stoean, Elia El-Darzi, Breast Cancer Diagnosis by Means of Cooperative Coevolution (3), Proceedings Workshop "Medical Informatics (in frame of Third ACM International Conference on Intelligent Computing and Information Systems ICICIS 2007), pp. 33-38, 2007.

26. D. Dumitrescu, **Catalin Stoean**, Genetic chromodynamics - a novel evolutionary heuristic for multimodal optimization, First International Conference on Multidisciplinary Information Sciences and Technologies, InSciT2006, Merida, Spain, Current Research in Information Sciences and Technologies. Multidisciplinary approaches to Global Information Systems, Volume 2, V.P. Guerrero-Bote (Ed.), Open Institute of Knowledge, 238-243, 2006.

27. **Catalin Stoean**, D. Dumitrescu, Mike Preuss, Ruxandra Stoean, Cooperative Coevolution for Classification, Bio-Inspired Computing: Theory and Applications, BIC-TA 2006, China, D. Dumitrescu, Linqing Pan (Eds.), pp. 289 - 298, 2006.

28. Ruxandra Stoean, D. Dumitrescu, Mike Preuss, **Catalin Stoean**, Different Techniques of Multi-class Evolutionary Support Vector Machines, Bio-Inspired Computing: Theory and Applications, BIC-TA 2006, China, D. Dumitrescu, Linqing Pan (Eds.), pp. 299-306, 2006.

29. **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Ruxandra Stoean, A Cooperative Coevolutionary Algorithm for Multi-class Classification, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC 2006, pp. 7-14, 2006.

30. Ruxandra Stoean, Mike Preuss, D. Dumitrescu, **Catalin Stoean**, Epsilon - Evolutionary Support Vector Regression, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC 2006, pp. 21-27, 2006.

31. **Catalin Stoean**, Ruxandra Stoean, Mike Preuss, D. Dumitrescu, Spam Filtering by Means of Cooperative Coevolution, 4th European Conference on Intelligent Systems and Technologies, ECIT 2006, Iasi, Romania, Advances in Intelligent Systems and Technologies - Selected Papers, H. N. Teodorescu (Ed.), Performantica Press, pp. 157-159, ISBN 973-730-246-X , 2006.

32. **Catalin Stoean**, Various Collaborator Selection Pressures for Cooperative Coevolution for Classification, 6th International Conference on Artificial Intelligence and Digital Communications - AIDC 2006, Thessaloniki, Greece, Research Notes in Artificial Intelligence and Data Communications, N. Tandareanu (Ed.), Reprograph Press, pp. 45-53, ISBN 973-742-413-1, 2006.

33. **Catalin Stoean**, Ruxandra Stoean, Mike Preuss, D. Dumitrescu, A Cooperative Evolutionary Algorithm for Classification, International Conference on Computers and Communications - ICCC 2006, Baile Felix Spa - Oradea, Romania, pp. 417-422, ISSN 1841-9836, 2006.

34. Ruxandra Stoean, **Catalin Stoean**, Mike Preuss, D. Dumitrescu, Evolutionary Multi-class Support Vector Machines for Classification, International Conference on Computers and Communications - ICCC 2006, Baile Felix Spa - Oradea, Romania, pp. 423-428, ISSN 1841-9836, 2006.

35. **Catalin Stoean**, Ruxandra Stoean, Mike Preuss, D. Dumitrescu, Diabetes Diagnosis through the Means of a Multimodal Evolutionary Algorithm, Proceedings of the First East European Conference on Health Care Modelling and Computation - HCMC 2005, Craiova, Romania, pp. 277-289, ISBN 973-7757-67-X, 2005.

36. **Catalin Stoean**, D. Dumitrescu, On Solving the 3-SAT Problem Using an Evolutionary Multimodal Optimization Technique, 5th International Conference on Artificial Intelligence and Digital Communications - AIDC 2005, Craiova, Romania, pp. 136-142, ISBN 973-671-014-9, 2005.

37. **Catalin Stoean**, Ruxandra Gorunescu, D. Dumitrescu, A New Evolutionary Model for the Optimization of Multimodal Functions, The Anniversary Symposium Celebrating 25 Years of the Seminar Grigore Moisil and 15 Years of the Romanian Society for Fuzzy Systems and A.I., International participation and committee, 2005, Iasi, Romania, Intelligent Systems, Selected Papers, H. N. Teodorescu, J. Watada, J. Gil Aluja, M. Mihaila (Eds.), Performantica Press, pp. 65 - 72, ISBN 973-730-070-X, 2005.

38. **Catalin Stoean**, Ruxandra Gorunescu, Mike Preuss, D. Dumitrescu, An Evolutionary Learning Spam Filter System, SYNASC 2004, Timisoara, Romania, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, D. Petcu, V. Negru, D. Zaharie, T. Jebelean (Eds.), Mirton Press, pp. 512-522, ISBN 973-661-441-7, 2004.

39. **Catalin Stoean**, Ruxandra Gorunescu, Mike Preuss, D. Dumitrescu, Evolutionary Discovery of Adaptive Rules for Spam Detection, Fourth International Conference on Applied Mathematics - ICAM4, North University of Baia Mare, Romania, Abstract Proceedings, pp. 34, 2004.

40. **Catalin Stoean**, Ruxandra Gorunescu, Mike Preuss, D. Dumitrescu, Evolutionary Detection of Rules for Text Categorization. Application to Spam Filtering, Third European Conference on Intelligent Systems and Technologies-ECIT'2004, July 21-23, 2004, Iasi, Romania, Inteligent Systems, Selected papers, H. N. Teodorescu (Ed.), Performantica Press, pp. 87-95, ISBN 973-7994-85-X, 2004.

41. **Catalin Stoean**, On Classifying Good and Bad E-mails, 4th International Conference on Artificial Intelligence and Digital Communications, Craiova, Romania, June 2004, Research Notes in Artificial Intelligence and Digital Comunications, 4, N. Tandareanu (Ed.), Reprograph Press, 2004, pp. 79-85, ISBN 973-671-014-9, 2004.

42. **Catalin Stoean**, Hierarchical Learning Text Categorization, International Conference on Computers and Communications - ICCC 2004, Bile Felix Spa - Oradea, Romania, pp. 383-387, ISBN 973-613-542-X, 2004.

Papers at national conferences

43. **Catalin Stoean**, D. Dumitrescu, Cloning within Genetic Chromodynamics, Proceedings of the "Colocviul Academic Clujean de Informatica", Babes-Bolyai University of Cluj-Napoca, Faculty of Mathematics and Computer Science, pp. 9-14, 2005.

44. **Catalin Stoean**, A New Technique for Text Categorization. Application to Spam Filtering, Proceedings of the "Zilele Academice Clujene" Symposium, May 25, 2004, Babes-Bolyai University of Cluj-Napoca, pp. 95 - 100, 2004.

Technical reports

International reports

45. **Catalin Stoean**, Ruxandra Stoean, Mike Preuss, D. Dumitrescu, Coevolution for Classification, Technical Report Nr. CI-239/08, Collaborative Research Center on Computational Intelligence, University of Dortmund, 2008.

46. Ruxandra Stoean, Mike Preuss, **Catalin Stoean** and D. Dumitrescu, Evolutionary Support Vector Machines and their Application for Classification, Technical Report Nr. CI-212/06, Collaborative Research Center on Computational Intelligence, University of Dortmund, 2006.

National reports

47. **Catalin Stoean**, D. Dumitrescu, A New Algorithm within Genetic Chromodynamics. Applications to Function Optimization and Classification, Technical Report, Department of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University, 2005, http://www.cir.cs.ubbcluj.ro.

Contents

List of Figures

List of Tables

Abstract

The problems that have a solutions search space with several points that are fitter than all the solutions in their neighbourhood represent *multi-modal problems*. Each of these points is called *local optimum* and, among them, the one with the highest value for the fitness function corresponds to the *global optimum*.

Most of the real-world problems have multiple local optima in the solutions search space. When dealing with such tasks, the goal is to escape the local optima and reach the global one. However, when there exist several global optima or even local optima that are very close to the global one(s), the aim of an evolutionary algorithm that is applied for such a problem is to provide a set of all almost global optima so that the user selects the most convenient one.

The thesis is concentrated on the development of efficient evolutionary techniques for multi-modal optimization problems. There are two techniques proposed within the existing genetic chromodynamics framework that proved, in the experiments conducted on a large suite of benchmark test functions, to be more efficient than the original method with respect to both accuracy and the necessary fitness evaluations budget, especially when the dimension of the problem is increased. A hybridized evolutionary technique, topological species conservation, is proposed for the same type of problems: It does not make use of any radius for separating the population into subpopulations and proved however to be very competitive even when the problem dimension is enlarged.

As there is a great necessity of applying research to real-world problems, the secondary goal of the current work is to find means of applying multi-modal evolutionary techniques to classification tasks. One of the techniques achieved within the genetic chromodynamics framework is used as an engine for a classifier. Then, both the cooperative and competitive coevolution algorithms are considered, in turn, as tools for proposed classifiers. All classifiers are tested against several real-world benchmark classification problems.

Chapter 1

Introduction

1.1 Evolutionary Computation. Inspiration and Rationale

The power of evolution in nature is obvious since diverse species that make up our world manage to survive and adapt in their own niches, sometimes in very cruel environments. Thus, it is not surprising that some computer scientists chose natural evolution as a source of inspiration for problem solving.

In order to solve a problem, an environment is created and is filled with a population of individuals. Learning is viewed as a process of continuous adaptation of an individual to the initially unknown environment. The adaptation of the individuals in the population is achieved through reproduction and mutation. The fitness of the individuals is closely related to how well they adapted to the environment and represents their chance of survival and multiplication. The individuals that form the population represent a collection of candidate solutions for the considered problem. As an outline of the correspondences between *natural evolution* and *problem solving*, the environment represents the problem, individuals are candidate solutions and the fitness corresponds to the quality of the solution. The quality of a candidate determines the chance that the considered individual has to be used as a seed for building new candidate solutions that will form the next generation.

Given an environment that can host only a limited number of individuals and the capacity to reproduce added to them, selection is inevitable if one does not want the population size to grow exponentially. Obviously, natural selection favours those individuals that adapt to the environment condition best – survival of the fittest – so the best ones survive and reproduce and the evolution progresses step by step. Occasional mutations take place in order to introduce new individuals to be tested. Thus, the constitution of the population changes as time passes and it evolves, offering in the end the solution(s) for the considered problem.

The evolutionary computation notion was introduced as a term only at the beginning of the 90s as an effort to bring together researchers working with different approaches, all simulating evolution: genetic algorithms, evolution strategies, evolutionary programming. All of them imply the use of selection of individuals in a population, reproduction, random variation and competition, which are the essence of evolution, both in nature or inside a computer [Fogel, 1997].

Evolution represents an optimization process that does not reach perfection, but still can obtain highly precise solutions to a large scale of optimization problems.

Most of the real-world problems have an immense solutions search space and a high number of local optima that the global solution cannot be found in reasonable time. Moreover, the problems can have dynamic components that can change the location of the optima in time and, therefore, the technique that is considered for solving them must adapt to the changes. Additionally, the final solution may have nonlinear constraints that have to be fulfilled (constrained problems) or may have objectives that are in conflict (multiobjective problems). Evolutionary algorithms represent an appropriate alternative for solving such problems: They are population-based approaches, thus multiple regions of the search space can be simultaneously explored. This represents a good advantage, especially when dealing with a multi-modal search space where an evolutionary algorithm keeps track of several optima in parallel and maintains diversity in the population of solutions, with the aim of performing a better exploration of the search space for finding the global optimum. When the considered problem is dynamic, the solutions in the population continuously adapt to the changing landscape and move towards the new optima. For multiobjective problems, evolutionary algorithms provide, in the end of the evolution process, a set of trade-off solutions for the conflicting objectives, while traditional techniques only produce one solution at the end of a run. For constrained problems, evolutionary algorithms offer a set of feasible and unfeasible solutions. Probably the main advantage of the evolutionary algorithms and the reason why they are so used nowadays is that they can be applied to any type of optimization problem, be that it is continuous or discrete. Another important advantage is that they can be easily hybridized with existing techniques.

Although evolutionary algorithms may appear as the right solver for any optimization problem, this is not the case: "If there is already a traditional method that solves a given problem, evolutionary algorithms should not be used" [Schwefel, 1997], as they cannot be better or with less computational effort. The computational effort indeed represents an important drawback of evolutionary algorithms as many candidate solutions have to be evaluated in the evolutionary process. Even if evolutionary algorithms do not necessarily provide the best possible solution for the problem at hand, they can be used for adding further improvements to solutions obtained by other means, only by incorporating these solutions in the starting population.

The domain whatsoever presents a continuous interest for researchers in this field as well as from completely different areas. Every time one deals with an optimization problem, evolutionary algorithms almost became the most attractive alternative, mainly because of their simplicity and flexibility.

1.2 Thesis Motivation and Objectives

In the evolutionary computation context, when dealing with multi-modal optimization there exist two goals. On the one hand, the global optimum has to be detected, meaning that search does

not have to get blocked into local optima. On the other hand, it is desired that a set of the most promising optima is kept and provided in the end of the evolutionary process, so that the user could select the ones that are most convenient.

Finding just the global optimum of an optimization problem represents a challenge for researchers that construct techniques for this matter. Designing methods that have to find several optima is therefore even more motivating. Why design other evolutionary techniques for multi-modal optimization? According to the no-free-lunch theorem [Wolpert and Macready, 1997], there cannot exist any algorithm for solving all problems that is generally superior/inferior to any alternative approach. Thus, there always exists the need for creating new evolutionary techniques but, especially, it is necessary to point out on what type of problem they perform better than other existing approaches and, it is of great interest to understand why they do so.

The main goal of this thesis regards the development of new, efficient evolutionary techniques for multi-modal optimization problems. The potential of the techniques is verified through their application on suites of benchmark problems. The second aim of the thesis is to try and find new means of addressing real-world tasks through evolutionary algorithms.

1.3 Achievements and Restraints

Regarding the development of new evolutionary methods for multi-modal optimization, three techniques are proposed:

- The first one is built on the bases offered by the genetic chromodynamics framework. It is meant to speed up the original algorithm and, additionally, to improve the solutions accuracy through the employment of a crowding procedure within [Stoean et al., 2005c], [Stoean and Dumitrescu, 2005a].

- The second proposed method [Stoean et al., 2005a] starts from the genetic chromodynamics with crowding technique. It introduces *clones* of the individuals that remain alone in their subpopulations, connected to different optima, and then mutates these clones. This way, fine tuning is performed for the final solutions. Both techniques are described in chapter 4.

- Another evolutionary algorithm for multi-modal optimization is proposed through the topological species conservation technique [Stoean et al., 2007a] (chapter 7): It does not make use of any radius for separating subpopulations, but only employs the fitness landscape topology instead. The technique arises from the hybridization of two recent multi-modal evolutionary methods, but avoids their drawbacks and inherits their strengths; the obtained results proved that, indeed, the new method in more accurate and, at the same time, disburdens itself of a crucial parameter, the species radius, which is very hard to parameterize.

As the second goal is concerned, the real-world task that is dealt with in the thesis is classification. Five benchmark data sets are considered, three coming from the University of California

at Irvine (UCI) Repository of Machine Learning Databases[1] and two with raw information. The ones coming from UCI represent widely used data sets like:

- Fisher's Iris data set that refers to the classification of iris plants, based on some of their attributes [Stoean et al., 2005c], [Stoean and Dumitrescu, 2005b], [Stoean et al., 2006b], [Stoean, 2007].
- Pima-Indian Diabetes data set in which positive or negative diabetes diagnosis, based on a list of patient features, has to be reached [Stoean et al., 2005c], [Stoean et al., 2005d].
- Breast Cancer data set that targets the breast cancer diagnosis of a set of patients, based on the values they have for some medical indicators [Stoean et al., 2007b], [Stoean, 2007].

The raw collections are briefly described below:

- The first one, from the University Hospital in Craiova, Romania, deals with hepatic cancer early diagnosis [Stoean et al., 2008c].
- The second one contains real, not preprocessed e-mails that are categorized into good or undesired messages, based only on their text [Stoean, 2004b], [Stoean, 2004a], [Stoean, 2004c].

All data sets considered in this thesis are described in Appendix B.

There are two chapters that target the practical assignment of classification in which three possibilities of evolutionary approaches for classification are proposed:

- One of them is based on the newly developed genetic chromodynamics with crowding technique [Stoean et al., 2005d], [Stoean and Dumitrescu, 2006], [Stoean and Dumitrescu, 2005c] (chapter 5).
- The cooperative coevolutionary approach is then used as a classification engine [Stoean et al., 2006d], [Stoean et al., 2006e], [Stoean et al., 2006b], [Stoean et al., 2006a].
- Competitive coevolution is finally employed as a classification mechanism [Stoean, 2007]. Both coevolutionary approaches are presented in chapter 6.

Regarding the restraints of the thesis, there are some open problems that remain for future work. As one of the targets of the thesis is to develop evolutionary tools designed for real-world application, the aim is to create the techniques as complete and easy to adjust as possible, i.e. with a minimum number of evolutionary parameters to be set. The mutation strength parameter is the one that is very much dependent of the considered problem and adding an adaptation mechanism for it would not only open a path leading to substantially increased performance, but also ease the work of the user. Another task that remains for future work is the development of a tool for detecting the topology of the fitness landscape, i.e. whether there exists a very high

[1] Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html

number of local and/or global optima. Knowing this information in advance can be very valuable for a technique dealing with a certain problem, be that it is evolutionary or not: It can help in setting the proper values for parameters or even in deciding the method that should be employed for solving the problem.

1.4 Thesis Outline

The thesis contains two survey chapters and then, starting with chapter 4 to chapter 7 the presentation is focused on the methods in which the author was directly implicated. Note however that the surveys, especially the one about evolutionary algorithms, do not contain too many details regarding representation, fitness function, operators etc, but only a brief introduction is provided. This happens due to fact that it is not the aim of this work to present generalities about evolutionary algorithms and, at the same time, the methods in which there exist personal contributions are largely described and space limitation has to be obeyed. The thesis is organized as follows.

Chapter 2 briefly sketches out the ideas behind any evolutionary algorithm, provides the steps that are followed in every such technique and outlines the main components.

Chapter 3 presents an overview of the most commonly used and recently developed evolutionary techniques designed for multi-modal optimization problems.

Chapter 4 outlines alternatives within the genetic chromodynamics framework. The methods are applied on a set of benchmark functions and comparison of the results with the original genetic chromodynamics algorithm, as well with other evolutionary techniques, is undertaken.

In chapter 5, an approach for classification based on one of the techniques presented in the previous chapter is presented. It is tested on several data sets and results proved to be competitive to those obtained by other state-of-the-art techniques in the literature.

Two different approaches for classification are proposed in chapter 6: They are based on cooperative and competitive coevolution. Again, application on real data sets is included.

Chapter 7 contains a new hybrid evolutionary method for multi-modal optimization, namely topological species conservation. The method is applied for function optimization and direct comparisons are conducted with the most powerful of the parent techniques. Results indicate that the newly developed method performs better while also removes a parameter whose value is hard to set.

The thesis encloses with the conclusions and ideas for future work in chapter 8.

Chapter 2

Evolutionary Algorithms. Basic Concepts

2.1 Objectives of this Chapter

The goal of this chapter is to summarize what is an evolutionary algorithm and which are its components. A general scheme that forms the common basis of most evolutionary algorithm variants is presented. The main components of EAs are then intuitively described. Note, however, that it is not the aim of current work to elaborate on general details about *evolutionary algorithms* (EAs) like certain selection or variation operators or to describe different types of representations specific to genetic algorithms, evolutionary strategies or evolutionary programming: For such information, please consult introductory books into evolutionary computation like [Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003], [Michalewicz, 1996].

2.2 What is an Evolutionary Algorithm?

There are many types of classes of evolutionary computation models and they are usually referred to as evolutionary algorithms [Bäck, 1996], [Bäck et al., 1997], [Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003]. A common underlying idea lies behind all these models: Given a population of individuals, the environment pressure causes natural selection, *survival of the fittest*, and consequently the average fitness of the generations rises step by step. Having a fitness function to be maximized, a set of randomly generated candidate solutions (or individuals) are created in the domain of the function and are evaluated – the better individuals (or solutions) are considered those with higher values for the fitness function. The evolutionary process starts when, based on the fitness values, some of the individuals are *selected* to be the parents of the population in the next generation; descendants are obtained by applying *recombination* and/or *mutation* to the previously chosen individuals. Recombination takes place between two or more individuals and one or more *descendants* (or *offspring*) are obtained; descendants borrow particularities from each of the parents. When mutation is applied to a candidate solution, the result is one new candidate that is usually only slightly different from its parent.

After applying these *variation operators*, mutation and recombination, a set of new individuals is obtained that will fight for survival with the old ones for a place in the next generation; the candidate solutions that are fitter are advantaged in this competition. The evolutionary process resumes and usually stops after a predefined computational limit is reached. A general scheme of an EA is presented in pseudocode and as flow-chart, in Algorithm 1 and Figure 2.1 respectively.

Algorithm 1 Pseudocode of an evolutionary algorithm

```
Require: A search/optimization problem
Ensure: The best obtained individual(s)
  begin
  Initialize population with random candidate solutions;
  Evaluate each candidate;
  while termination condition is not satisfied do
    Select parents;
    Recombine pairs of parents;
    Apply mutation to offspring;
    Evaluate resulted offspring;
    Select individuals that will form the next generation;
  end while
  return best obtained solution
  end
```

Variation operators have the role of introducing new candidates into the population and, in this way, explore the space of solutions. But by applying variation operators, worse solutions might be encountered; the task of avoiding the decrease in quality of the populations with respect to the fitness evaluations is achieved by selection. Selection chooses, exploits the fitter candidate solutions, so the average value of the population fitness consequently increases. In conclusion,

1. Variation operators create diversity in the population, having an explorative role, while
2. Selection favours the fitter individuals, having therefore an exploitative task.

In order to reach the optimum solution of the problem to be solved, a good equilibrium between exploration and exploitation has to be established ([Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003]). If too much exploitation is done and only little exploration, some promising regions from the solutions search space might remain unexplored, so the best solutions might not be found at all; in this case, the search process remains blocked into a local optimum. Otherwise, if too much exploration is done and only little exploitation, the search process is very much slowed down and the convergence of the algorithm to the optimum solution might become too much time-consuming.

Many components of an EA are stochastic. When selection is applied, fitter chromosomes have higher chances to be chosen than the less fit ones, but even the weak individuals, with respect to the fitness evaluation, have a (smaller) chance to be selected. In the same way, when

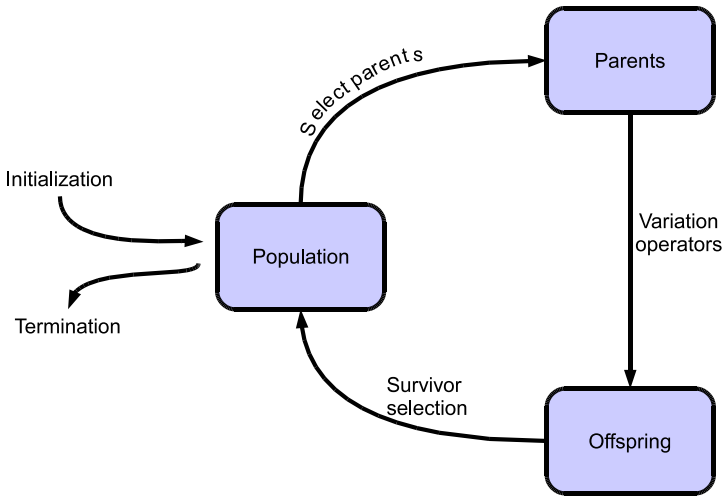

Figure 2.1: General scheme of an evolutionary algorithm

recombination is applied to two or more candidates, the offspring pieces are randomly chosen from each of the parents. In the case of mutation, the parts of the individual that are changed or *mutated* are also randomly chosen.

There is a variety of classes of EA models and all follow the above general outline, differing only in technical details. Next section presents the common components of a typical EA.

2.3 Components of an Evolutionary Algorithm

The most important components of an EA may be identified as follows:

- Representation of candidate solutions
- Population model
- Fitness function
- Selection
  - Parent selection strategy
  - Survival selection strategy
- Variation operators

Each of these components has to be specified in order to define a particular EA. For completing an EA, there appears the need for an initialization of the candidate solutions that will form the initial population and for a termination condition, as well.

2.3.1 Representation

When solving a problem using EAs, the first thing to do is to set up a bridge between the problem space and the EA space. Objects forming possible solutions of the original problem (*phenotypes*) have to be encoded into individuals within the EA (*genotypes*).

Sometimes, the phenotypes space and the genotypes space may coincide, while other times they can be completely different. For instance, if one has an optimization problem on integers, real-valued representation for genotypes can be chosen, so the two spaces would be the same, or we could decide on a binary representation, i.e. a solution of the form "21" from the phenotypes space would be represented as "10110" in the genotypes space. The way representation is chosen depends very much on the problem to be solved. In conclusion, a mapping is therefore necessary to transfer the solution of the problem into the EA space (*encoding*) and an inverse mapping will be needed also, in order to transform the EA solution back into the problem solution (*decoding*).

Genotypes or *individuals* are also called *chromosomes*. The chromosome is composed of *genes*. A gene is located at a particular position in the chromosome. A gene may contain several values or may have several forms. Each value of a gene is referred to as an *allele* of that gene.

2.3.2 Population

The population has the role of preserving all candidate solutions at one time; it consists of a set of genotypes that are not necessarily all different from each other. The population as a unit is the one that evolves, not the individuals; when selection is applied for choosing the parents of the population that will form the next generation, selection relates to the current population, by picking better candidates.

The population size represents the number of individuals that the population contains. Usually, the population size, which is a parameter of the EA, is constant from the start to the end of the algorithm.

The initialization of the population concerns the population size together with representation; each gene of every individual generally takes a random value from its domain of representation. Sometimes, the EA can start using as an initial population a set of fixed candidate solutions that might be obtained using other methods.

2.3.3 Fitness function

The role of the fitness function (or *evaluation function*) is to measure how much the individuals adapted to the environment. The evaluation function is mostly used by selection and thereby it makes improvements possible.

The fitness function is defined on the space of the chromosomes and its values are real numbers, in order to be able to make comparisons between the qualities of different individuals. Returning to the example from the subsection dealing with representation, presuming a real-valued function is to be optimized, e.g. $f(x) = x^2$, the fitness of the genotype 10110 is $21^2 = 441$.

In many cases, the objective function, which is the name used in the original context of the problem, coincides with the fitness function or the fitness function is a transformation of the given objective function.

2.3.4 Selection

Selection appears twice during a *while* loop of Algorithm 1: First, there is selection for reproduction, when parents of the next generation are chosen (*parent* or *mating selection*) and then, selection for replacement, when individuals that will form the population in the next generation are chosen from the offspring and chromosomes in the current population (*survivor selection*). Both selection types are responsible for quality enhancing.

Parent selection

Parent selection has the role of choosing, based on their quality, which of the individuals in the current population are considered to undergo variation in order to create offspring. Parent selection is typically probabilistic; high-quality individuals have a good chance to be selected for reproduction, while low-quality ones have small chances of becoming parents.

The operator does not create any new candidate solutions, but only selects relatively good solutions from a population and deletes the remaining solutions. Multiple copies of certain individuals are placed in a new population by removing inferior solutions.

As stated before, the main idea is that individuals with a better fitness must have a higher probability of being selected. Nevertheless, selection operators differ in the way the chances are assigned to better solutions: Some operators sort the individuals in the population according to their fitness and then deterministically choose some few best individuals. Meanwhile, other operators assign a selection probability to each individual which is dependent on its fitness, and subsequently achieve selection through the use of a probability distribution. In the latter situation, there exists the possibility of selecting a bad solution and, at the same time, rejecting a good solution. However, this can be an advantage: the fittest individuals in a population can be connected to a suboptimal region in the fitness landscape and, by using a deterministic selection, the EA would evidently converge to the wrong, suboptimal solution. If, conversely, a probabilistic selection is employed in such case, diversity is maintained for a higher number of generations by selecting some less fit individuals and, therefore, more exploration is performed, the EA would therefore be prevented from converging to a wrong solution.

Some of the most popular selection schemes are briefly mentioned in the following paragraphs.

When *proportional selection* is used, the number of copies an individual will have is directly proportional to its fitness: A solution having double the fitness of another solution will also have twice as many copies in the selected population. The most commonly used form of implementing selection probabilities within the proportional type is the roulette-wheel (or Monte-Carlo) mechanism, where each individual in the population occupies a section on a roulette that has a size directly proportional to its fitness. Then, the wheel is spun as many times as the population size

and each time the solution indicated by the wheel is selected. Evidently, solutions with better fitness have higher chances to be selected (and therefore to have several copies in the population) as their sections on the wheel are proportional to their fitness. There are however limitations of the proportional selection scheme:

- It cannot handle minimization problems directly, but they have to be transformed into maximization problems.
- It cannot handle negative values for fitness, as they should correspond to the sections on the roulette-wheel
- If there exists a very fit individual in comparison to all the others in the population, proportional selection selects a very high number of copies of that individual and this causes to loss of diversity in the population which conducts to premature convergence.
- On the other hand, if all individuals have very similar evaluations, fact that usually happens later on in the evolution process, the roulette-wheel will be marked approximately equally for all individuals in the population and all of them will have almost the same chances of being selected (the effect of random selection).

All the drawbacks mentioned above can be avoided by using a scaling scheme, where the fitness of every solution is mapped into another interval before marking the roulette wheel [Goldberg, 1989].

Within *tournament selection*, all these limitations are eliminated: In a tournament of n individuals, the best solution is chosen either in a deterministic or probabilistic fashion. In the simplest form, n is 2, the scheme is called binary tournament selection and the best one of the two solutions is chosen. This selection operator does not depend whether the fitness values are positive or negative and, when one deals with a minimization problem (and not a maximization one), the only difference is that the individuals with a smaller fitness value are selected (instead of the one with the higher fitness value).

The last selection operator mentioned here is the *ranking selection*. It is similar to the proportional selection, but instead of using the direct fitness value, the individuals are ordered according to their performance values and attributed a corresponding rank. It can also treat negative evaluations and, when minimization is required, the only difference is that ranking has to be inversely performed.

Survivor selection

The survivor selection decides which individuals from the current population and their offspring are retained to form the population of the next generation; as the number of individuals in the population is usually constant, survivor selection intervenes in order to preserve population size. This selection decision is usually based on the fitness of the individuals, favouring fitter candidate solutions.

2.3.5 Variation operators

While selection operator has the task of focusing the search on the most promising regions of the search space, the role of variation operators is to create new candidate solutions from the old ones and to increase population diversity. Variation operators are representation dependent, as for different representations, different operators have to be defined [Bäck, 1996], [Bäck et al., 1997], [Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003]. The most often used variation operators are recombination and mutation.

Recombination

Recombination or *crossover* is a binary variation operator that usually forms one or two offspring by generally combining the genes of two individuals that are considered as parents. Recombination represents a stochastic operator as choices like which parts are to be inherited from one parent and which from the other or the way the two parts are combined depend on a pseudo-random number generator.

In conclusion, by mating two individuals with different features, an offspring (or two) that combine those features is (are) obtained. The aim is to explore the space between the two individuals, in search of a chromosome that has a better quality.

Crossover can even take place between p individuals ($p > 2$) and p offspring may be obtained [Bäck et al., 1997], [Dumitrescu, 2000a], [Dumitrescu et al., 2000]; one offspring can also be obtained by combining features from the p parents.

Mutation

Mutation is a unary variation operator. When applied to an individual, the result, also called *offspring*, contains minor modifications of the initial individual. Through mutation, individuals that cannot be generated using crossover may be introduced into the population. This operator makes all the values of a gene available for the search process. The mutation operator is a stochastic one also, so the genes whose values are considered to be changed are chosen in a probabilistic manner.

The termination condition of a typical EA refers to reaching a previously set number of generations, or not exceeding a predefined number of generations that passed without achieving any improvement, or it may concern finding a solution with a given accuracy.

As a result of the algorithm, one may consider the best individual with respect to the fitness function, from the last generation, or the best from the entire process, or, sometimes, the entire (or a subset of the) population from the last generation [Bäck, 1996], [Dumitrescu et al., 2000].

2.4 Summary

The basic ideas of this introductory chapter into evolutionary computing can be summarized as follows:

- A general idea of what is an evolutionary algorithm is firstly outlined.
- The components of a typical evolutionary algorithm are subsequently enumerated and are followed by brief descriptions.

Chapter 3

Evolutionary Techniques for Multi-modal Optimization

3.1 Objectives of this Chapter

The aim of this chapter is to outline what is a multi-modal problem and which are the most common and powerful evolutionary state-of-the-art techniques that are able to approximate solutions for these problems. The target of all the evolutionary techniques for the multi-modal optimization purpose represents the maintenance of diversity for a longer period; in this way, the search space would be more extensively covered for a longer period and local optima could be detected, as a global optimum would not attract the entire population of candidate solutions. There are several possibilities for maintaining diversity like the modification of the fitness function (section 3.3), the arrangement of the candidate solutions into groups of individuals that can only interact between themselves (section 3.4), schemes of inserting the descendants into the population (section 3.5) or the division of individuals depending on the search space (section 3.6). The last can be divided into two parts: Radii-based evolutionary techniques, in which the mating partners of the individuals are chosen from their close range, and the division of the individuals according to the fitness landscape topology.

3.2 Multi-modal Problems and the Necessity for Diversity Preservation

Most of the real-world problems have a fitness landscape topology with numerous unequal hills which can be misleading for an evolutionary technique. Thus, EAs must avoid the blocking of the search into local optima and direct it towards the global optimum.

The solution space may be intuitively viewed as a 2D (or 3D) representation, where candidate solutions can be plotted as a set of points, each individual being a dot in this landscape. In such a representation, high altitude stands for high fitness (Figure 3.1). Therefore, each peak corresponds to highly fit solutions that are reached through successful combinations, while lower areas correspond to less fit solutions. Hence, evolution is achieved when the population gradually

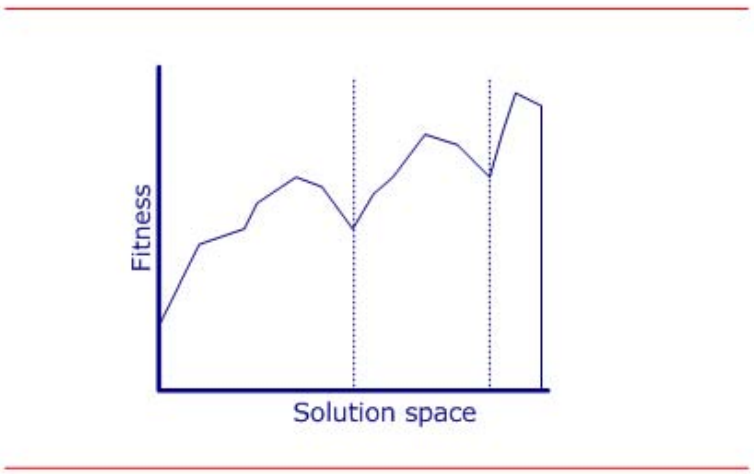

Figure 3.1: Three optima with different shapes and different sizes for their basins of attraction.

advances to high altitude regions by means of variation operators and natural selection.

The problems in which there is a number of points that are fitter than all the solutions in their neighbourhood represent *multi-modal problems*. Each of these points is called *local optimum* and, among them, the one with the highest value for the fitness function corresponds to the *global optimum*. The disjoint regions of high fitness are also called *niches* [Bäck et al., 1997], [Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003].

There are two distinct targets that are followed when an evolutionary technique deals with such a multi-modal optimization task:

- Probably the most commonly met goal is to find the (almost) optimal solution out of a high number of suboptimal solutions. In order to achieve that, one has to keep track of the already found peaks and direct the search towards the unexplored parts of the solutions space.

- Multiple (almost) optimal or even suboptimal solutions have to be located. For some problems, a set of different solutions might be necessary in order for the most convenient one to be selected by a human expert as the final decision.

It is important to notice that the space of solutions is partitioned into *basins of attraction* around optima, meaning those areas that delimit each of the local optima. In Figure 3.1 there are three optima, each one with its own basin of attraction (basins are delimited by doted lines); the size of the basin is very important, as if it is bigger, more candidate solutions are present there, at least in the stage of generating the population.

As previously stated, multi-modal EAs are mostly used for identifying more highly fit solutions, corresponding to several local optima, but they can be also used in the case when only a global optimum has to be located, especially when the searched optimum does not have the largest

basin of attraction. Suppose there are two nearly equal optima in the search space of the problem and the basin of the worse local optimum is a little larger. In such a case, if one used a typical EA, the population would be randomly scattered following a uniform distribution at first, but a bit more individuals (maybe 52% of them) in the basin of the worse local optimum. As there are more candidate solutions in this basin, selection is most likely to take more of the seeds for next generation from it. As generations pass, the two subpopulations become more and more unbalanced, until, in the end, the basin of the global optimum of the two remains with no individuals; metaphorically speaking, the solutions stepped down from the hill that represented the global optimum and then climbed on the other hill. That is the reason why in such cases EAs that preserve subpopulations connected to different local optima have to be exploited. In what follows, some largely used evolutionary models for solving multi-modal optimization problems are presented.

To conclude, the main target of an EA that deals with a multi-modal optimization problem is to avoid the premature convergence, i.e. stagnation into a local optimum. This goal can be achieved by maintaining diversity within the population for a longer period of time. Genetic diversity is important for pertaining a better exploration of the search space: This is especially achieved through recombination of very different individuals, while applying the same recombination operator to very similar chromosomes does not yield new, different candidate solutions. However, although recombining very different individuals leads to a good search space exploration, two parents that follow different peaks are very likely to produce an offspring that has a very low fitness. Hence, premature convergence is to be avoided, diversity is important in early stages, but its high maintenance can be of no use, especially when approaching the final steps of the evolutionary process; now, fine tuning should be performed and crossover between very different individuals would only damage the solutions.

From the early years of evolutionary computation, numerous techniques, or extensions to existing ones, have been proposed for improving performance when dealing with multi-modal problems. In the following, the most popular together with some of the recently developed evolutionary techniques, all gathered in groups depending on their common characteristics, are to be presented.

3.3 Fitness Modification for Diversity Maintenance

Two approaches in which the evaluation of the individuals is differently performed are outlined in this section: One of the most used evolutionary techniques for multi-modal optimization is represented by *fitness sharing*. The fitness of each individual is computed by directly depending on the number and fitness of its closely ranged individuals. Thus, each region of the search space contains (in the best case scenario) a number of individuals that is directly dependent to its fitness.

The other method that is included in the same group of techniques with fitness modifications

for diversity maintenance is the cooperative coevolution approach. A complete solution of the problem at hand is decomposed into several pieces and each such piece is searched by means of an EA. When evaluating an individual from one subpopulation (that correspond to a piece of solution), collaborators from the other subpopulations are selected in order to assemble a complete solution that is evaluated. The two approaches are further presented.

3.3.1 Fitness Sharing

Each subpopulation corresponds to a niche and each niche represents an optimal region in the search space. The goal of the *niching method* is to force the population to preserve different subpopulations connected to the niches.

In an EA based on fitness sharing, individuals of a certain niche have to share the niche resources and that is achieved by modifying the fitness function. Therefore, very similar individuals have to share the resources at a higher level, as they belong to the same niche and, at the same time, very different individuals have a low level of sharing.

For each individual x_i in the population, a value m_i (the *sum of its niche*) is computed as (3.1).

$$m_i = \sum_{i=1}^{n} sh(d(x_i, x_j)) \tag{3.1}$$

where

- n is the population size;
- $d(x_i,\ x_j)$ is the distance between individual x_i and x_j; there can be various measures for the distance, depending on representation (e.g. Euclidean, Manhattan distance for real encoding, Hamming distance for discrete encoding etc.);
- $sh(d)$ is a sharing function given by (3.2).

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha}, \text{if } d < \sigma_{share} \\ 0, \text{ otherwise} \end{cases} \tag{3.2}$$

There are two parameters to be set here: α is one of them, but it is usually taken equal to 1, and the other is the share radius σ_{share}, which decides how many niches are maintained and its value directly depends on the problem to be solved. If two individuals are identical, the value of the sharing function applied to the distance between them is maximal. The value for the sharing function tends to zero as the distance between individuals grows towards infinity.

Deb and Goldberg [Deb and Goldberg, 1989] proposed a manner of computing the value for the σ_{share} radius that leads to the formation of subpopulations; that was afterwards embraced by most of the researchers dealing with such parameters. It uses the radius of the smallest hypersphere containing feasible space, which is given by (3.3).

$$r = \frac{1}{2}\sqrt{\sum_{i=1}^{n}(x_i^u - x_i^l)^2} \tag{3.3}$$

where n represents the number of dimensions of the problem at hand and x_i^u and x_i^l are the upper and the lower bounds of the i-th dimension. Knowing the number of global optima N_G and being aware that each niche is enclosed by an n-dimensional hypersphere of radius r, the niche radius σ_{share} can be estimated as:

$$\sigma_{share} = \frac{r}{\sqrt[n]{N_G}} \tag{3.4}$$

Although the approximation given by equation (3.4) is very precise, in many cases, especially for real-world applications, one does not know the number of global optima in advance and, therefore, in such situations, it cannot be used.

The fitness sharing function of an individual x_i will be obtained by dividing its fitness $f(x_i)$ to the sum of its niche (3.5).

$$f'(x_i) = \frac{f(x_i)}{m_i} \tag{3.5}$$

As similar individuals share their fitness on a high level, the number of individuals in any region of the search space is limited by the region fitness. Thus, the number of individuals in the attraction basin of an optimum depends on the height of that peak.

3.3.2 Cooperative Coevolution

The first step that has to be undertaken when a problem is intended to be solved by cooperative coevolution is to find a proper decomposition of the problem into components. Then, each component is assigned to a population (or species); each population evolves independently (but concurrently with the others), except for the moment when the evaluation process takes place. Each individual in a population represents a component of the solution to the problem and thus a potential solution for every component in turn cannot be assessed apart from those of the complementary parts. Therefore, every individual of each population is evaluated by selecting collaborators from every other species; a complete solution to the problem at hand is thus reached and its performance is computed and returned as the fitness value of the current individual.

Algorithm 2 describes the usual steps that are followed by a cooperative coevolutionary algorithm. It starts with the initialization of each population. In order to measure the quality of a certain individual, for the first evaluation, a random selection of individuals (collaborators) from each of the other populations is performed and obtained solutions are evaluated. After this starting phase, each population is evolved using a canonical EA.

As mentioned before, the choice of collaborators represents an important issue in this process. Consequently, when building a cooperative coevolutionary algorithm, there are three attributes regarding collaborator selection that have to be chosen [R. P. Wiegand and Jong, 2001]. Further

Algorithm 2 Cooperative coevolutionary algorithm

Require: A search/optimization problem
Ensure: A complete solution to the problem
begin
$t = 0$;
for each species s **do**
randomly initialize population $\text{Pop}_s(t)$;
end for
for each species s **do**
evaluate $\text{Pop}_s(t)$;
end for
while termination condition = false **do**
$t = t + 1$;
for each species s **do**
select population $\text{Pop}_s(t)$ from $\text{Pop}_s(t - 1)$;
apply genetic operators to $\text{Pop}_s(t)$;
evaluate $\text{Pop}_s(t)$;
end for
return a complete solution
end while
end

detail regarding these attributes and applications of the cooperative coevolution approach can be found in chapter 6.

Through this manner of evaluating the candidate solutions, their fitness depends entirely on the collaborators that are chosen. Therefore, there are no individuals that are clearly advantaged by the selection operator, therefore the diversity is maintained for a longer period of time. Cooperative coevolutionary results for the optimization of multi-modal functions proved to be comparable to the ones obtained by other EAs designed for multi-modal optimization. However, note that they were applied for multi-modal optimization problems aiming to find only the global optimum out of several possible points, and not for the detection of several optima.

3.4 Spatial Population Topologies

A good way to avoid premature convergence is to separate the entire population into groups of individuals that, can basically only interact between themselves. In this way, different groups can converge to different regions of the search space and, genetic diversity is especially maintained more extensively preserved. Three methods are presented in this group of techniques, two of which are classical and one is recently developed.

3.4.1 Island Model

One of the initial ways of tackling multi-modal optimization problems is to rerun the same EA several times and to collect all the found solutions in each of the undertaken runs. However, it has the disadvantage that it is not always the case that different optima are detected. This simulation could be nevertheless achieved in only one run, by using an *island approach.*

The island model assumes that multiple subpopulations are built and maintained in tandem. After a (usually fixed) number of generations called epochs, *migrations* between the islands take place, meaning that a number of individuals are taken from each subpopulation in order to make exchanges with the other neighbouring subpopulations [Bäck et al., 1997], [Dumitrescu, 2000a], [Dumitrescu et al., 2000], [Eiben and Smith, 2003].

A good equilibrium between exploitation and exploration is achieved since, in between epochs, when subpopulations evolve independently, they explore and exploit the search space around the fitter solutions; when migration takes place, individuals with potentially high fitness or completely different genotypes are exchanged between neighbouring islands; moreover, recombination is to take place between very dissimilar individuals. This interchange of individuals represents a plus as compared to rerunning the same EA several times and each time collecting the results.

Although it looks very nice in theory, it is not always the case that different islands are connected to different optima; each subpopulation, or most of them, might very well converge to the same optimum that may be the one that has the largest basin of attraction.

There are also some parameters that have to be set:

1. How to divide the population into subpopulations – a minimum subpopulation size has to be considered. On the other hand, the more subpopulations, the higher the chances to find the optima (or global optimum), as more subpopulations have chances to explore more peaks.

2. How often do migrations take place – if they are too frequent, all subpopulations converge to the same optimum; if migrations happen rarely, slow convergence may take place in some of the islands. Usually migrations take place after 25-150 generations or if after a previously set number of generations, say 25, no improvement is achieved.

3. Which individuals and how many of them are exchanged – to avoid premature convergence, a small number of individuals is exchanged (2-5). Chromosomes that are exchanged may be taken based on fitness selection or randomly. Another decision has to be taken, that is whether the migrating individuals are moved or just copied; in the latter option, a replacement mechanism has to be introduced for preserving subpopulations' sizes.

3.4.2 Diffusion Model

A single population is considered and it is split into a large number of smaller subpopulations (or neighbourhoods, or demes) that are distributed within algorithmic space: this is achieved by

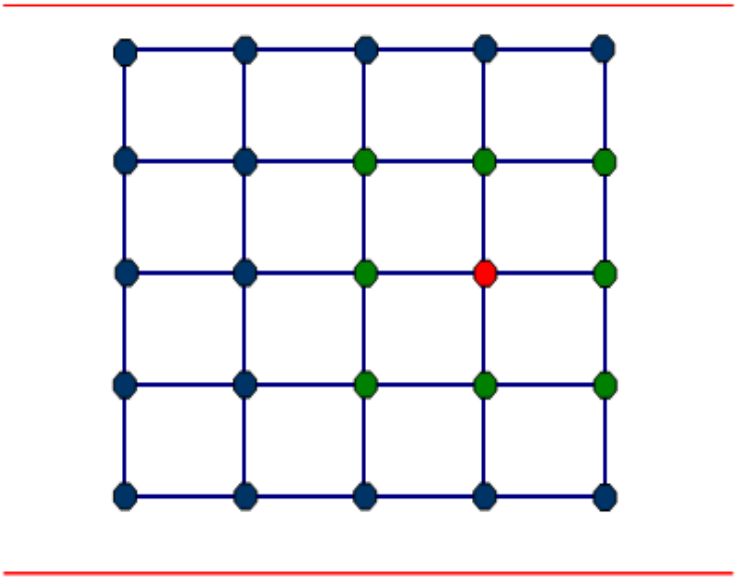

Figure 3.2: Population distribution on a grid. The current individual is the red one and its neighbours are represented in green.

considering that each individual exists on a different point on a *grid* and selection and recombination take place only between neighbours (see Figure 3.2): Note that individuals that are situated on the margins of the grid have as neighbours the individuals positioned on the opposite side of the grid, which means that the entire grid represents a torus. In this way, even if a chromosome has a very good fitness, it cannot change the entire population immediately; first it would change only its neighbouring individuals and, then, these might spread the good genotype structure forward and so on, thus only slowly *diffusing* throughout the population. Diffusion models are also referred to as distributed EAs or cellular EAs, since the intuitive idea behind these models is that the individuals spread throughout the global population like molecules in a diffusion process [Bäck et al., 1997], [Dumitrescu, 2000a], [Eiben and Smith, 2003].

There are many implementations for the diffusion EA and one of them (a more common one) is outlined in Algorithm 3.

3.4.3 Religion-Based Evolutionary Algorithms

The religion-based EAs (RBEAs) [Thomsen et al., 2000] is inspired by the religion concepts in the real world and how religions attract their believers. The technique starts from the idea of the diffusion model and adds the concept of religions (subpopulations) and a scheme for drawing new believers to a religion.

The RBEA uses a grid world like in the diffusion model case, where each cell may be empty or may contain an individual. Each individual has three fixed actions that are performed in turn. First, each agent tries to do a *random walk* to a vacant cell from its neighbourhood. After this

Algorithm 3 Pseudocode of a diffusion evolutionary algorithm

Require: A search/optimization problem
Ensure: The best obtained individuals
begin
Initialize population and distribute it randomly on a grid, with one individual per node;
Define neighbourhood (deme) for each individual the (usually the node and all its immediate neighbours are considered);
while termination condition is not satisfied **do**
for each deme **do**
Select two individuals from the nodes in the deme;
Recombine them and obtain an offspring;
Mutate the offspring;
Replace a randomly taken solution in a node of the current deme with resulted offspring;
end for
end while
return a set of the best solutions
end

step is performed, the individual tries to convert individuals belonging to other religions that are located in its neighbourhood on the grid. An individual belief can be changed to another religion if its own religion has a number of believers higher than a given threshold value. Moreover, conversion is stochastic and only possible if the individual that is trying to convert others is fitter than them. Finally, the individual attempts to reproduce with a nearby mate, but only from its own religion. Therefore, mating is restricted to individuals belonging to the same religion (subpopulation), which corresponds somehow to a mixture between the island and the diffusion models: There exist *migrations* between subpopulations (which correspond here to the conversion from one religion to another) but, on the other hand, all candidate solutions are arranged on a grid like in the diffusion model case. For more details regarding the insights of the RBEA, see [Thomsen et al., 2000].

In the undertaken experiments, RBEA proved to be significantly better than a standard EA and better on five out of six test cases than a cellular EA.

3.4.4 Multipopulation Differential Evolution

Differential Evolution (DE) has been proposed in [Storn and Price, 1995] as a tool for global optimization. The idea behind the technique is to select from the population of candidate solutions, in each generation and for each genome, three random individuals x_{r_1}, x_{r_2} and x_{r_3} and, starting from these, to form a *trial* chromosome 3.6.

$$y_i^j = \begin{cases} x_{i,r_3}^j + F(x_{i,r_1}^j - x_{i,r_2}^j) \text{ with probability } p_c \\ x_i^j \text{ with probability } 1 - p_c \end{cases} \tag{3.6}$$

where x_i^j represents the j-th gene of the current individual x_i, F is a differentiation constant and $p_c \in [0, 1]$ is a probability value which controls the ratio of new components in the descendant. The trial individual subsequently fights for survival with the current one and with the best individual in the population.

One enlargement of DE to multi-modal optimization regards the integration of crowding within the algorithm [Thomsen, 2004]: the obtained offspring replaces its most similar individual in the population if it is fitter.

Another modification of the DE for multi-modal optimization that actually assumes the existence of several subpopulations is proposed in [Zaharie, 2004a], [Zaharie, 2004b]. The population is divided into equally sized subpopulations and a DE is applied for each one of them. In the initialization process, each subpopulation is randomly generated in a certain subdomain. During evolution, the subpopulations are not restricted to the initial subdomains. After a given number of generations, a migration process take place: Each element of a subpopulation has a probability to be picked for migration to another randomly chosen subpopulation where it can replace a random individual. The best elements from each subpopulation are stored into an archive; to avoid redundancy in the archive, the hill-valley mechanism in [Ursem, 1999] (that is also described in subsection 3.6.2, Algorithm 9) is utilized and then doubled by the verification of whether the selected individuals are not too close to each other. For the latter condition, one employs a threshold that depends on a resolution factor, which is also used for cutting the domain into subdomains during initialization. The proposed technique proved to be very powerful for the optimization of several multi-modal functions.

3.5 Replacement Schemes

Cavicchio's dissertation [Cavicchio, 1970] was one of the first attempts to use a replacement scheme for preserving diversity for a longer time. Cavicchio introduced a procedure called *preselection*: Each obtained offspring has to fight for survival with the weakest parent. Only five years later, at the same University of Michigan, De Jong generalizes Cavicchio's work by introducing the *crowding* method [Jong, 1975] that proved to be of great importance for future research in the evolutionary computing field.

3.5.1 Crowding

The crowding technique was introduced as a method of maintaining diversity: new obtained individuals replace only similar individuals in the population. In this method, a percentage G (*generation gap*) of the individuals is chosen via fitness proportional selection in order to create an equal number of offspring. For each of these offspring, CF (a parameter called *crowding factor*) individuals from the current population are randomly selected; the offspring then replaces the most similar individual of the CF that are selected. Usually, the CF parameter is taken equal to 2 or 3, while generation gap parameter G is 10%. However, there were some issues that arise

around this technique, as it suites only a limited set of multi-modal problems.

Mahfoud proposed an improvement to the original crowding algorithm that was called *deterministic crowding* ([Mahfoud, 1995]). In this technique, the offspring fight for survival with their parents only. A small description of it is outlined below:

1. Random pairs are formed from the parent population.
2. Each pair generates two offspring via recombination.
3. Mutation is then applied to these offspring.
4. Each offspring competes for survival with the most similar parent.

This way, subpopulations are preserved on each niche but, in this case, the size of each subpopulation does not depend on the fitness of each optimum region; moreover, the population is equally distributed amongst the available peaks.

3.6 Search Space Division

In all the techniques discussed in this section, the entire population is separated into subpopulations that contain only neighbouring individuals. The presented techniques are grouped into two types: The ones that make use of one (or more) radius (or radii) for creating the subpopulations and one that employs the fitness landscape topology for the same purpose. While the radii-based schemes have the disadvantage of using an additional parameter (i.e. the radius) that is very dependent on the problem to be solved, the converse method of using the fitness landscape is very expensive with respect to the number of fitness evaluations.

3.6.1 Radii-Based Schemes

There is a high amount of methods that make use of a radius parameter for separating the entire population into subpopulations. For computing the value for the radius, one can use formula (3.4). The fitness sharing method of section 3.3, could be also included here, because it uses a radius parameter as well. Other important radii-based evolutionary techniques for multi-modal optimization tasks are subsequently outlined.

Species Conservation Genetic Algorithm

The species conservation genetic algorithm (SCGA) concentrates on two aspects [Li et al., 2002]: The determination of species based on similarity criteria, where each subpopulation is dominated by a locally best individual, and the preservation of these individuals to the following generation of the evolutionary cycle.

A species is defined as a subset of individuals where the distance between every two members is less than the diameter given by a user-defined species radius. Each subpopulation is built

around the best local individual, called *species seed*. A species is centred on its seed if, for its every member, the distance to it is less than the species radius. At the beginning of every generation, the mechanism that differentiates the species collects the seeds by taking into account each individual of the population, in decreasing order of fitness, and testing whether it belongs to any species centred on the already found seeds (i.e. its distance to a certain known seed is less than the species radius) or is a new seed. The way of finding the seeds of each subpopulation is described by Algorithm 4. Population size is denoted by n.

Algorithm 4 Seeds selection within the SCGA

Require: The population P of the EA
Ensure: The seeds within P
begin
$Seeds = \Phi$;
Sort population P decreasingly according to fitness;
for $i = 1$ to n **do**
$found$ = FALSE;
for every s in $Seeds$ **do**
if $\mathrm{d}(P_i, s) \leq radius$ **then**
$found$ = TRUE;
end if
end for
if not $found$ **then**
$Seeds = Seeds \cup P_i$;
end if
end for
return the seeds set
end

The method that is responsible for conserving the dominating individuals acts on the population after the application of the variation operators. For every seed s in turn, the worst fit individual w that belongs to the species centred on s (i.e. its distance to s is less than the species radius) is considered. If this individual exists and is less fit than s, then the former is replaced by the latter. Otherwise, if the species of s does not contain any individual at all (as they may have disappeared because of selection or variation operators), then the worst individual from the entire population w is replaced by s. The individual s that enters the population is prevented from further substitution during the current generation. Apart from conserving the fittest individuals from the entire population, this mechanism allows the preservation of even less fit individuals, different enough from the global best ones, that could be at that moment positioned at the basis of an empty peak and thus become useful in future iterations. Subpopulation conservation is thus carried out through the conservation of the seeds of each species; the process is illustrated by Algorithm 5.

An important remark that concerns the EA as a whole is that selection and recombination of

Algorithm 5 Seeds conservation within the SCGA

Require: The current population P
Ensure: The population that contains the seeds
begin
Mark all individuals in P as unprocessed;
for every s in *Seeds* **do**
Take worst unprocessed w, such that $\mathrm{d}(w, s) \leq radius$;
if w exists **then**
if $f(w) < f(s)$ **then**
$w = s$;
end if
else
Take worst unprocessed w in P;
$w = s$;
end if
Mark w as processed;
end for
return the population with the integrated seeds
end

individuals are both performed globally, irrespective of the different species they belong to. This may be counterproductive and resembles one of the critical parts of the technique.

At the end of the algorithm, the optima are selected from the seeds that are lastly chosen. This *Seeds* set will not contain only the desired optima, but also low fitness individuals that are stored because they are sufficiently different from all the other individuals. Therefore, another parameter, *solution acceptance threshold* r_f $(0 < r_f \leq 1)$, is introduced so that all solutions with high enough fitness are selected: $x \in Seeds$ is selected if the inequality

$$f(x) \geq (f_{max} - f_{min}) \cdot r_f$$

is satisfied, where $f(x)$ denotes the fitness of x, while f_{max} and f_{min} represent the maximum and minimum fitness in the final population.

Particle Swarm Model

Some generalities about the standard particle swarm model are primarily introduced in order to prepare the presentation of the specific technique designed for multi-modal optimization.

The particle swarm based optimization has been introduced in [Kennedy and Eberhart, 1995] as an alternative for genetic algorithms for function optimization. The optimization is obtained by particles flying through the problem solution space. The particles are evaluated and the position of the best found so far solution is retained (personal best or *pbest*). The particles communicate with the neighbouring ones in order to store the best position found by the others (best global optimum or *gbest*). By making use of the two optima values found up to the current moment,

the particles generate a vector of velocities that is used to update their positions; the equations describing this behaviour are as in (3.7).

$$\begin{aligned} x_i(t) &= x_i(t-1) + v_i(t) \\ v_i(t) &= wv_i(t-1) + c_1 r_1(p_i - x_i(t-1)) + c_2 r_2(p_g - x_i(t-1)) \end{aligned} \tag{3.7}$$

where $v_i(t)$ represents the velocity of the i-th particle at generation t, p_i and p_g are the previous best positions of the particle (*pbest*) and of its neighbours (*gbest*) respectively, c_1 and c_2 are two positive constants, w the inertia weighting and r_1 and r_2 two random numbers in the range [0,1]. Usually, the value of w decreases together with the generation number, fact that allows a better exploration at the beginning of the process which gradually changes into a fine tuning.

The pseudocode for a standard particle swarm method is described in Algorithm 6.

Algorithm 6 Standard particle swarm algorithm

Require: A search/optimization problem
Ensure: The best obtained individual(s)
begin
Randomly initialize the particle population and the vector with the velocities.
while the stop condition is not met **do**
Evaluate the particles;
for each particle i **do**
Update the best evaluation of particle i;
Update *pbest*;
Update velocity and position for particle i according to (3.7);
end for
end while
return best obtained solution(s)
end

The particle swarm technique for multi-modal optimization, as proposed in [Parrott and Li, 2004], uses a radius that delimits regions where *gbest* optima are employed; all particles that lie at a smaller distance than the given radius use their own *gbest* value, fact that encourages the particles to converge towards local optima.

A parameter *pmax* that gives that maximum number of particles within a subpopulation (particles that share the same *gbest*) is introduced. If there are more than *pmax* particles in a subpopulation, the redundant ones are reinitialized in other locations of the search space.

All particles in a subpopulation use the personal best value (*pbest*) and the *gbest* of the subpopulation. The subpopulations are reconstructed in each generation. The random reinitialization of the weaker individuals that belong to a subpopulation that has more than *pmax* particles is probably a very strong point as diversity is constantly reintroduced.

As demonstrated in [Parrott and Li, 2004], the technique allows the particles to follow multiple optima in a dynamic multi-modal environment.

Genetic Chromodynamics

Genetic Chromodynamics (GC) [Dumitrescu, 2000b] belongs to the family of radii-based multi-modal evolutionary frameworks, as it builds and maintains subpopulations connected each to the local or global optima of the problem to be solved. This is achieved by introducing a set of restrictions such as the way selection is applied or the way recombination takes place. For selection, each individual represents a *stepping-stone* for the forming of the new generation – each individual is taken into account for reproduction. If a chromosome has no similar individuals to it, then it mutates. For reproduction, a local interaction principle is considered, meaning that only individuals similar under a given threshold recombine. After either recombination or mutation takes place, the offspring fights for survival with the stepping-stone parent.

For the subpopulations to become better and better separated with each iteration, a new operator, called merging, is introduced. It merges very similar individuals and is applied after perturbation takes place. Let an individual c be given. If the distance between c and another individual is very small, under a given radius, then the latter is considered part of the merging region of c. From the set of all individuals in the merging region of c, only one is kept. Generally, it is decided such that the one with the best fitness evaluation is kept in the population. Alternatively, other merging schemes may be used (for instance, the mean of the individuals in the merging region). The merging procedure is outlined in Algorithm 7.

Algorithm 7 Merging procedure

Require: The current population
Ensure: The population after merging
  begin
  repeat
    an individual c is considered the current one;
    select all m chromosomes in the merging region of c, including itself;
    remove all but the best chromosome from the selection;
  until merging can not be applied at all
  return the population after merging
  end

By means of merging, subpopulations become better and better separated with each iteration and the population size is reduced in a way that preserves the position of each subpopulation. The interplay between merging and mating regions strongly impacts convergence properties: a large merging radius leads to the deletion of potential mates of the surviving individuals and thus blocks crossover. Therefore, it is usually chosen to be much smaller than the mating area. Along with separation, the worse individuals are removed step by step, and the process gradually transforms from a globally–oriented population–based to a parallel local search technique. Therefore, in the end of the evolution, only one individual remains connected to every optimum. Each such individual now corresponds to a single hill–climber that only uses mutation as variation operator. Figure 3.3 illustrates the GC radii-based mating and merging, while Algorithm 8 outlines the

pseudo-code.

GC is able to concentrate search on many basins of attraction in parallel, so that several optima are found simultaneously. The evolutionary process takes place as follows. First, the initial population is randomly generated. Next, every individual is taken into account in the forming of the new generation; mating regions around each individual are determined by a radius. Therefore, only neighbouring individuals are recombined. When no mate is found in the mating region of the current individual, the latter produces one offspring by mutation, with a step size that still keeps the descendant in the mating region of its parent. If there is more than one individual *near* the current, the mate is determined using proportional selection. Then, if the offspring has better fitness than the current individual, it replaces the latter in the population.

By reducing the potential partners for crossover of an individual to those lying in its mating region, only individuals that are close to each other recombine, favouring the appearance and maintenance of subpopulations. Offspring replaces the current individual only if fitter. Thus, after a few generations, the candidate solutions will concentrate on the most promising regions of the search space, i.e. those connected to the optima.

Algorithm 8 GC algorithm

Require: A search/optimization problem
Ensure: The set of solutions corresponding each to an optimum
begin
$t = 0$;
initialize population P(t);
repeat
evaluate each individual;
for all individuals c in the population **do**
if mating region of c is empty **then**
apply mutation to c;
if obtained individual is fitter than c **then**
replace c;
end if
else
choose one individual from the mating region of c for crossover by proportional selection;
obtain and evaluate one offspring;
if the offspring has better fitness than c **then**
replace c
end if
end if
end for
merging;
$t = t + 1$;
until stop condition
return obtained solutions
end

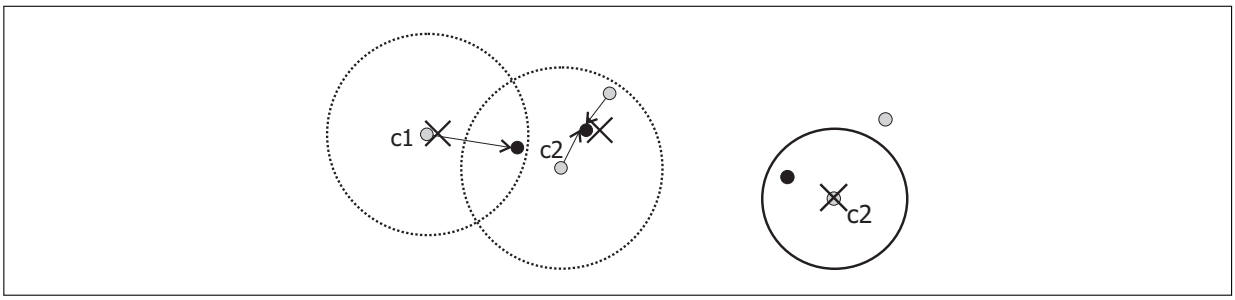

Figure 3.3: Mating (left) and merging (right) within GC. Individual *c1* produces one offspring by mutation; individual *c2* selects another individual from its mating region (dotted circle) and produces one offspring by recombination. Crossed lines indicate replaced individuals with worse fitness. During merging, *c2* is deleted because there is another individual with better fitness in its merging region (solid circle).

Merging is an original and very useful operator as it leads to a better computational time, since, by reducing the size of the population, less fitness evaluations are necessary. Consequently, subpopulations independently evolve and become better separated with each iteration and lead, at convergence, each one to an optimum.

GC proved to be very successful in application to function optimization, clustering and classification. However, it makes use of two parameters like the mating and merging radii which are, for some problems, hard to parameterize. Nevertheless if the mating parameter is computed using the Deb and Goldberg [Deb and Goldberg, 1989] formula (3.4) and smaller values, as compared to it, are tried for the merging radius, one could reach configurations that provide consistent results.

3.6.2 Multinational Algorithms

The technique [Ursem, 1999], [Ursem, 2000] has a very interesting working metaphor: the whole population represents a world and each subpopulation is called a nation; each nation has a government which contains its fittest individuals, referred to as politicians. They define the policy of the nation, which is in fact the centroid of these individuals. Initially, the world consists of a single nation. Then, the governments and, as a consequence, the policies are useful for the differentiation and the dynamics of the subpopulations.

The multinational idea is achieved through two mechanisms: migration and merging. Within every generation, each individual is compared to the policy of the nation it belongs to. If the two follow different optima, then the former migrates to a nation whose policy is tracking the same peak as itself. If there exist no such nation, then the individual forms a new nation, corresponding to a potentially new peak. Conversely, also at every generation, the policies of each nation are checked two by two to ensure that they do not track the same optimum; if this is the case, then the two subpopulations are merged.

Selection can be performed on two levels: either within each subpopulation or in a global but weighted fashion, i.e. the fitness of an individual is divided by the number of members in its corresponding nation, in a fashion that resembles sharing. Finally, crossover is performed only

between individuals of the same nation, as the combination of the genetic material of points that track different optima may lead to the appearance of offspring that are less fit than the parents. It is assumed that mutation is also restricted to each nation, i.e. the offspring is accepted only if it remains within the premises.

The verification of the relationship between two points in the search space, i.e. of the assumption that they track the same optimum, is performed through an approach called the *hill-valley* mechanism. The function takes two individuals (points) as arguments and returns whether or not there is a valley between them in the fitness landscape, i.e. they track different optima. In order to reach that decision, a set of interior points between the two, based on user-defined gradations in the [0,1] interval, is generated. If the fitness of all interior points is higher than the minimal fitness of the two tested individuals, then it is concluded that they track the same optimum. Contrarily, if there exist such a point whose fitness is smaller than the minimal fitness of the two, then it is assessed that they follow different peaks. To conclude, *hill-valley* returns true if the two points follow different optima and false if they follow the same peak. The mechanism is described in Algorithm 9. The fitness evaluation of the individual x is denoted by $f(x)$.

Algorithm 9 The hill-valley mechanism between two individuals x and y

Require: Two individuals, x and y
Ensure: *true*, if x and y follow different optima and $false$ otherwise
begin
$i = 1$;
$found$ = FALSE;
while $i <$ *number of gradations* and not $found$ **do**
for $j = 1$ to *number of dimensions* **do**
$interior_j = x_j + (y_j - x_j) \cdot gradation_j$;
end for
if $f(interior) < min(f(x), f(y))$ **then**
$found$ = TRUE;
end if
end while
return $found$
end

An important advantage of this manner of detecting multi-modality is that a certain optimum is tracked by a single subpopulation, whereas the radii-based mechanisms usually allow the existence of several subpopulations that follow the same peak. The great disadvantage of the method is that it is very expensive with respect to the number of used fitness evaluations.

3.7 Summary

In this chapter, the most important and some of the newly developed evolutionary techniques for multi-modal optimization are presented. The chapter can be summarized as follows:

- A description of the multi-modal optimization concept, along with the motivation for having diversity in the population of candidate solutions, are first envisaged.
- Techniques that have a modified fitness function, like fitness sharing and cooperative coevolution are then presented.
- Methods that are based on topological arrangements of the population, in which interactions are restricted only to groups of individuals (subpopulations) are shown: There are examples of classical techniques like the island or the diffusion models or, more recently developed ones, like the religion-based EA or the multipopulation differential evolution.
- Replacement schemes are then outlined: Cavicchio's preselection is mentioned, but the presentation is more oriented on De Jong's crowding.
- The division of the search space, based on the distances between individuals or just starting from the fitness landscape topology, is then presented. Several radii-based evolutionary techniques are described and the multinational algorithm is outlined as a technique that takes into consideration only the fitness landscape when separating individuals into subpopulations.

Chapter 4

New Variants within the Genetic Chromodynamics Framework

4.1 Objectives of this Chapter

The chapter proposes a new enhanced technique within the Genetic Chromodynamics (GC) framework that speeds up convergence and, at the same time, looks into the search space for more accurate approximations of the solutions. The new technique is applied herein for function optimization and results obtained are compared with those of the classical method in the GC framework and with those of other models applied for the optimization of the considered functions [Stoean et al., 2005c], [Stoean and Dumitrescu, 2005b], [Stoean et al., 2005a].

A second method within the GC framework is discussed; it introduces new copies of the fitter individuals into the population (a process of cloning), a concept depicted from the theory of artificial immune systems; a better exploitation of the search space is thus conducted. The method provided fine results for function optimization [Stoean and Dumitrescu, 2005a], [Stoean and Dumitrescu, 2005b].

4.2 A Crowding Procedure within Genetic Chromodynamics

Belonging to the family of radii-based multi-modal evolutionary frameworks, GC [Dumitrescu, 2000b] has recently proven to be a very powerful optimization tool [Dumitrescu, 2004], [Gorunescu and Millard, 2004], [Stoean et al., 2004b], [Dumitrescu and Stoean, 2006a], [Dumitrescu and Stoean, 2006b]. Recall that, as discussed in chapter 3, GC builds and maintains subpopulations, corresponding each to a global/local optimum of the problem, through the use of a stepping stone search mechanism and a local interaction principle, as selection for reproduction. Selection for replacement takes place between the resulting offspring and the current *stone* individual. A merging operator is used to achieve decrease in the number of individuals.

The work presented here goes further in the development of another EA within this framework. The selection for replacement strategy used in the new technique is generational. In contrast to the stepping stone mechanism, the first parent is selected randomly. The offspring obtained after

recombination does not replace any of the parents particularly, but the worst individual (with respect to fitness values) within its *replacement radius*, a new parameter of the method. The local interaction principle and merging still hold.

The reason for creating the new method was that of preserving the ability of GC to properly locate several or all optima within one go and additionally speed up this process. The new model always achieves the first goal, the second one being accomplished for more complex problems (for instance, for higher dimensional problems).

In the proposed novel technique, the original scheme was modified in order to achieve increased convergence speed based on a better exploitation of the search space. That is obtained especially through the way in which the offspring resulting from recombination enters the population. It does not replace the first parent, but the worst individual in its replacement radius. Therefore, weak individuals are removed more aggressively (alongside with the effect merging has in this respect) from the current population. For this reason, the stepping stone principle is not applied here, but n (where n is the population size) random individuals are selected instead. Now individuals may be replaced by some offspring without ever being selected for recombination. An important aspect of the method is the choice of the replacement radius value. If picked properly, this new parameter may lead to improved convergence speed.

Selection for replacement adopts a generational scheme, as already stated. As the effect of quasi-generational survival selection, used by the classical GC, can be noticed only when another stepping-stone does not find the initial individual in its mating region but sees the offspring instead, the new technique is totally generational. This means that the offspring that replaces its parent might be selected for reproduction many times in the same generation or might vanish within that generation. Thus, the generational scheme leads to increased exploitation.

The local interaction, recombination, mutation and merging principles still hold. Radii-based evolution in the new context of GC with crowding [Stoean et al., 2005c], [Stoean et al., 2005a] is depicted in Figure 4.1 and the complete method is outlined in Algorithm 10.

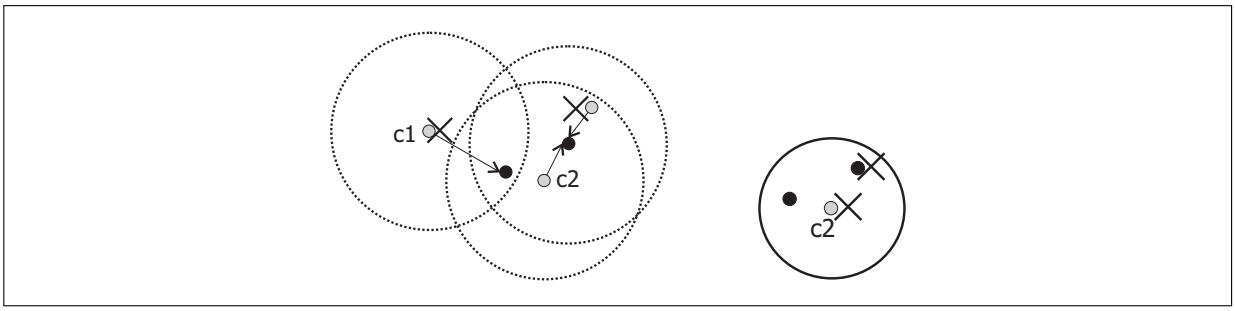

Figure 4.1: Mating (left) and merging (right) within GC with crowding. As in figure 3.3, c1 and c2 each produce one offspring. This time, the second offspring replaces its other parent because the latter is the worst individual in its replacement region. During merging, two individuals are removed, c2 and one offspring, because now three individuals are within merging radius from c2.

Algorithm 10 GC with crowding

Require: A search/optimization problem

Ensure: The set of solutions corresponding each to one local/global optimum of the problem

```
begin
t = 0;
initialize population P(t);
repeat
  evaluate each individual;
  for i = 1 to n do
    randomly choose an individual c;
    if mating region of c is empty then
      apply mutation to c;
      if obtained individual is fitter than c then
        replace c;
      end if
    else
      choose one individual from the mating region of c for recombination by proportional selection;
      obtain and evaluate one offspring d;
      find worst individual w within replacement radius of d;
      if d has better fitness than w then
        replace w;
      end if
    end if
  end for
  merging is applied to all individuals;
  t= t + 1;
until stop condition
return obtained solutions
end
```

{Remark: The offspring d obtained after recombination can replace an individual that does not belong to any of the mating regions of its parents, thus performing faster separation of individuals into clusters.}

4.3 Application to Function Optimization

The proposed technique was tested on three bi-dimensional functions and an n-dimensional one (n = 100 was considered as the upper bound) and obtained results indicate that it has good accuracy, stability, low computational time and thus provides a good method for multi-modal function optimization. Results in this section are reported in [Stoean et al., 2005c], [Stoean et al., 2005a] and [Stoean and Dumitrescu, 2005b].

4.3.1 Benchmark Functions

The functions are selected in such a manner that they have several characteristics and, therefore, several abilities of the evolutionary technique are tested. The considered functions are the six-hump camel back, Schaffer and Himmelblau that are two-dimensional and Schwefel function for which up to 100 variables are considered. First function tests the technique's ability to find and maintain two global optima and four local ones, while for Himmelblau function four global optima that are located on an almost plate platform have to be reached. Only one global optimum, but out of a multitude of local optima, has to be found for both Schaffer and Schwefel functions. As it is usually stated that radii-based evolutionary techniques do not perform well for higher dimensions, the method is tested for the Schwefel function for up to 100 variables. For more details regarding the considered functions, see Appendix A.

4.3.2 Task

The primal task of the experiments is to test the crowding GC technique in solving the optimization problems by making use of parameters that are empirically found. Direct comparison to the original GC method, as well as to other techniques that were tested on the same optimization problems is undertaken.

In order to strengthen the assumption that for the tested high dimensional problem, crowding GC is really faster than the original GC model, a second task is considered, that of applying a recent parameter tuning method named Sequential Parameter Optimization (SPO) [Bartz-Beielstein et al., 2004a], [Bartz-Beielstein, 2005], [Bartz-Beielstein, 2006], with the purpose of having an objective comparison. SPO builds on a quadratic regression model, supported by Latin Hypercube Sampling (LHS) and noise reduction, by an incrementally increased repetition of runs.

4.3.3 Experimental Setup

Table 4.1 contains the empirically determined crowding GC parameter values for the four functions. Mutation with normal perturbation was used in all cases; a value of the gene i of an individual P is changed according to (4.1)

$$P_i = P_i + R \cdot ms, \tag{4.1}$$

where R is a random number with normal distribution and ms is the mutation strength parameter. The value of the mutation strength directly depends on the size of the interval and implicitly on search space size. In case of the Schwefel function, a higher mutation strength was chosen in the beginning, to escape local optima. The value was afterwards decreased by $n/100$ every 100 generations, where n represents the number of dimensions. The values for mating and merging radii are chosen in the same manner, thus depending on the search space size. The last parameter from Table 4.1 gives the number of generations without any improvement which is necessary to determine the termination of the algorithm. For the Schaffer and Schwefel functions, the stop condition was established to be the moment when the technique achieves the desired accuracy because only the global optimum had to be determined.

Intermediate recombination was used – having two randomly selected parents P and Q, the value of a gene i of the offspring O is obtained according to (4.2)

$$O_i = P_i + R \cdot (Q_i - P_i), \tag{4.2}$$

where R is a uniformly distributed random number over [0, 1].

Table 4.1: Parameters of the method for all functions.

Parameters	Six-hump camel back	Schaffer	Himmelblau	Schwefel
Population size	100	2000	150	500
Mutation strength	0.1	0.1	0.5	$25n$
Mating radius	0.14	0.1	2	$15n$
Replacement radius	0.1	0.1	2	$15n$
Merging radius	0.14	0.1	0.02	$12n$
Mutation probability	0.4	0.4	0.4	0.4
No improvement times	100	–	50	–

For each test function, there are reported the accuracy on which the optima, global and local where necessary, are found, the success rate and the mean of the fitness evaluations. The success rate is computed as the ratio between the number of cases when all optima have been located with the desired accuracy and the total number of runs.

4.3.4 Results and Visualization

For the six-hump camel back function, [Lee and Roh, 2002] use several techniques (a genetic algorithm using gradient information, a local optimization method, a multi-start local optimization method and a conventional GA), but each time only one local or global optimum was detected with an accuracy of $\epsilon = 10^{-3}$. Because of lack of information, direct comparisons to the GC techniques could not be conducted. In Table 4.2, results obtained by the original GC model, the one with crowding and three approaches presented in [Zaharie, 2004a]: Crowding based differential evolution (CDE), multiresolution multipopulation differential evolution (MMDE) and multipopulation crowding differential evolution (MCDE). For these the local optima accuracies are not specified. Although the number of generations in GC with crowding was lower than that in the

classical model of GC, the number of evaluations in the novel technique was a little higher; the explanation is that after each recombination, there are some more evaluations that are performed for all individuals in the replacement area of the offspring. To conclude, the two related techniques perform in a very similar manner for this function.

Table 4.2: Comparisons between GC and crowding GC for Six-Hump Camel Back function.

Measures	CDE and MCDE	MMDE	GC	Crowding GC
No. of runs	30	30	30	30
Acc. global opt.	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Acc. local opt.	–	–	10^{-4}	10^{-4}
Success rate	100%	100%	100%	100%
Mean eval calls	62 645	14 610	19 923	19 980

For Schaffer function the parameters are chosen such that the entire search space was covered because the global optimum is surrounded by a high number of local optima that have a very close fitness. As noticeable from Table 4.3, the original GC model, although performing well, needs many more evaluations than GC with crowding.

Table 4.3: Performance of the original and the newly proposed GC on Schaffer function.

Results	GC	Crowding GC
No. of runs	30	30
Accuracy global optimum	10^{-6}	10^{-6}
Success rate	100%	100%
Mean eval calls	658 211	349 712

In [Beasley et al., 1993], the Himmelblau function was tested using a sequential niche technique; again, obtained results cannot be directly compared with those obtained by both GC techniques due to the fact that, in order to detect all the four optima, several runs (6.1, on average) of sequential niching were necessary. Additionally, the desired accuracy was not specified. Using the newly proposed technique, all four optima are detected for each of the 30 runs with the accuracy $\epsilon = 10^{-5}$. A performance comparison between both GC techniques is given in Table 4.4.

Table 4.4: Performance comparison between GC and crowding GC on Himmelblau function.

Measures	GC	Crowding GC
No. of runs	30	30
Accuracy global optima	10^{-4}	10^{-4}
Success rate	100%	100%
Mean eval calls	87 307	97 817

The aim for the Schwefel function is to find the global optimum $f_4(\overline{x}) = 418.9829n$. In this paper, f_4 was considered in turn for $n = 1, 2, ..., 100$. Although radii-based EAs usually have difficulties with high-dimensional test problems, the proposed technique detects the optimum with an accuracy of $\epsilon = 10^{-2}$, even for $n = 100$, in all 100 runs, but only after a high number of

fitness evaluations. The accuracy improves with the decrease in the number of dimensions.

The Schwefel function was also tested with the original GC model which detected the optimum for $n = 100$ with similar accuracy, but needed even more fitness evaluations than crowding GC. Table 4.5 presents a comparison of both techniques and different performance measures for $n = 10$ and $\epsilon = 10^{-2}$.

Table 4.5: GC and crowding GC performance on Schwefel function when n=10.

Numerical results	GC	Crowding GC
No. of runs	30	30
Average evaluations to solution	1 819 593	1 545 752
Mean generations	48 747	39 043
Mean best fitness	4189.827822	4189.827956

Results for the Schwefel function are compared to those in [Gordon and Whitley, 1993]. In the latter, authors presented the behavior of four global EAs, four island algorithms and one cellular algorithm on the shifted Schwefel function with $n = 1, 2, ..., 10$. In the first category, a simple genetic algorithm (SGA), an elitist SGA (ESGA), the parallel CHC (pCHC) and the Genitor algorithm are present. The second category contain Island-SGA (I-SGA), Island Elitist SGA (I-ESGA), Island-pCHC (I-pCHC) and Island-Genitor (I-Genitor). The techniques are run for a set number of generations and the number of runs (out of 30) in which the global optimum is found (denoted by ns) is reported, along with the mean best fitness of the 30 runs (denoted by MBF in Table 4.6). Note that fitness values are normalized so that the global optimum is 0.0.

Table 4.6: Performance of different evolutionary techniques on Schwefel function, as reported by [Gordon and Whitley, 1993].

Technique	Ns	f_4 MBF
SGA	0	17.4
ESGA	16	17.3
pCHC	15	5.9
Genitor	20	13.2
I-SGA	9	6.5
I-ESGA	13	2.6
I-pCHC	28	0.2
I-Genitor	24	0.9
Cellular	26	0.7

Returning to the novel approach, the replacement radius for a given individual is generally chosen to be equal to the mating value of that individual. However, for problems with large plateaus in the fitness landscape, by choosing a different value for the replacement from that of the mating radius, faster convergence can be achieved (see Tables 4.5 and 4.1 vs. Table 4.7 for

Table 4.7: Method design for optimizing the 10-dimensional Schwefel function with GC and crowding GC. The two last columns give the best found configurations, the last lines the resulting AES measures with standard deviations.

Parameter name	N/R	Min	Max	GC	Crowding GC
Population size	N	10	500	10	13
Mating radius	R_+	0.01	5.0	3.6752	2.9366
Replacement radius	R_+	0.01	5.0	–	4.748
Merging radius	R_+	0.01	5.0	1.275	3.9197
Mutation strength	R_+	1.0	250.0	238.45	231.08
Mutation prob.	R_+	0.0	1.0	0.6185	0.8575
Mutation scaledown	N	10	1000	368	625
No improv. times	N	1	200	167	46
AES, 32 runs	–	–	–	1.69E+5	1.45E+5
AES standard dev.	–	–	–	5.92E+4	3.63E+4

Schwefel function). On the other hand, for problems with very close optima, its value must be again chosen as not equal to the mating radius (see Six-hump camel back function in table 4.1), in order to obtain all optima.

SPO was firstly employed in order to tune the parameters for the 10 dimensional Schwefel function. It was run for 6 steps with an initial design size of 50, adding up to a total of around 850 runs (table 4.7). The response value Y (quality criterion) was set to the average number of evaluations (AES) until reaching the global optimum with accuracy 10^{-2}.

4.3.5 Observations

Results show that for both techniques, the AES decreased considerably, to $\approx 10\%$ of the original values reported in Table 4.5. A t-test confirms that the true means for GC and crowding GC are different with 95% confidence (p-value 0.036), so that we conclude that the new GC technique is indeed faster on this problem. However, the difference is quite small — around 15% — and may be statistically significant, but of little importance for practical uses. Surprisingly, many optimized parameter values are very similar for both techniques, e.g. number of individuals, mating radius, and the mutation strength and probability. In contrast to that, the merging radii are chosen differently; larger than the mating radius for crowding GC, and smaller for GC. Nevertheless, other good configurations found during tuning indicate that smaller merging radii also work for GC with crowding.

From these first results, and supported by our previous findings, we learnt that crowding GC may be advantageous for higher dimensional problems. Consequently, a second comparison for the same test function in 20 dimensions was performed, this time allowing for larger radius values. The outcome (Table 4.8) unambiguously favours crowding GC over GC, thereby validating our assumption. However, the maximum radii tested may still be too small; increasing them may lead to even better speedup.

Table 4.8: Method design for optimizing the 20 dimensional Schwefel function with GC and crowding GC; the maximum radii are increased compared to Table 4.7.

Parameter name	N/R	Min	Max	GC	Crowding GC
Population size	N	10	500	24	10
Mating radius	R_+	0.01	15.0	11.695	11.02
Replacement radius	R_+	0.01	15.0	–	9.6561
Merging radius	R_+	0.01	15.0	14.768	3.81
Mutation strength	R_+	1.0	250.0	242.16	237.43
Mutation prob.	R_+	0.0	1.0	0.6705	0.6185
Mutation scaledown	N	10	1000	269	368
No improv. times	N	1	200	8	167
AES, 32 runs	–	–	–	3.32E+5	1.07E+5
AES standard dev.	–	–	–	1.10E+5	2.26E+4

4.3.6 Conclusions

Through the increased exploitative nature of GC with crowding, it seems that a better equilibrium between exploration and exploitation, than that of the standard GC technique is established. This leads to two advantages in using crowding GC.

First, the novel approach proves to be very accurate and stable in very hard multi-modal conditions. A first situation is that of the difficulty in locating the local optima, e.g., the Six-Hump Camel Back function. Another hard configuration would be very distant optima, e.g. the Schwefel function, where other techniques often miss the global optimum. Another issue is having more global optima which are not easily distinguishable in the fitness landscape, like in the case of the Himmelblau function. An even harder case is that of local optima lying too close to the global optimum - where an EA usually experiences the problem of getting stuck into a local optimum. This is the case with the Schaffer function. Several EAs fail in some of these cases, several in other cases. The classical GC model, on the other hand, also performs well in this respect, but its speed of convergence is not very good for functions with many local optima very near to the global optimum, e.g. the Schaffer function and, especially in the case of more complex problems (n-dimensional problems), e.g. the Schwefel function.

As the question of computational time arises, for many low-dimensional multi-modal functions, both techniques are surely not competitive to other, namely non-evolutionary algorithms. For the other cases, i.e. almost indistinguishable local optima and global optima and high n-dimensional function optimization, for instance, both GC techniques perform accurate and stable, however, crowding GC is significantly faster.

Another important feature of proposed method is the new parameter, the replacement radius. Together with the generational scheme, it seems to have brought more power to crowding GC. For instance, for the Schwefel function, where the optima are far from each other, this parameter promotes rapid movement of the individuals through the search space. In the Six-Hump Camel Back function, where the optima are very close to each other, the parameter is responsible for the

small steps necessary not to escape them. Parameter tuning methods as the employed SPO can help to exploit the newly added potential.

4.4 Cloning within Genetic Chromodynamics

A *cloning procedure* is introduced within the GC with crowding method (Algorithm 11) in order to force a better *exploration* of the search space near the optima; this is done by introducing more copies of an individual connected to a current optimum and applying mutation to all of them. The individuals obtained after mutation that are better than the initial one are introduced in the next generation. Through this way of retaining only better individuals, *exploitation* of the search space is also conducted.

Cloning draws its roots from the theory of artificial immune systems ([de Castro and Zuben, 2002], [Dasgupta, 1999]). In the crowding GC technique, cloning appears if there is no individual left in the mating region of the current chromosome. Thus, new individuals can appear in the population of the next generation and will be in the same mating region; therefore recombination will take place for the current individual again.

Algorithm 11 Cloning mechanism

Require: A current individual

Ensure: A number of mutated clones for the individual

begin

an individual is considered to be the current one;

a fixed number of clones (copies) of the current individual is considered;

mutation is applied to all copies;

mutated copies are evaluated and all individuals that have a higher value for the fitness function than the one of the current individual are introduced in the population of the next generation;

return mutated clones

end

In the GC with crowding method, when an individual remains alone in a region, meaning that there is no other individual in its mating area, mutation is applied to it until the algorithm finally stops; this leads to a very high number of mutations applied to that chromosome. By introducing new individuals that are in the same mating region with the first individual, recombination will resume and take place between them and the convergence process will be sped up.

4.5 Reapplication to Function Optimization

The method is applied to function optimization and the considered functions are the three regular multi-modal, two-dimensional functions that were described before, i.e. Six-Hump Camel Back, Schaffer and Himmelblau. Performance comparison is conducted between the results obtained by GC with cloning and those of crowding GC. For each of the three test functions, each algorithm is

Table 4.9: Results obtained for the Six-Hump Camel Back function in 10 runs using GC with cloning.

Technique	SR	Accuracy		Mutations	Rec	Gen	Evals
		Global	Local				
Crowding GC	100	10^{-5}	10^{-4}	8 393	215	563	20 208
GC, 1 clone	100	10^{-5}	10^{-4}	9 759	2 711	408	66 204
GC, 2 clones	100	10^{-5}	10^{-4}	11 925	4 071	359	98 341

Table 4.10: Results obtained for the Schaffer function in 10 runs using GC with cloning.

Technique	SR	Accuracy	Mutations	Rec	Gen	Evals
Crowding GC	100	10^{-6}	30 269	959	137	390 002
GC, 1 clone	100	10^{-6}	3 388	5 239	60	267 136
GC, 2 clones	100	10^{-6}	3 659	6 800	53	381 116
GC, 3 clones	100	10^{-6}	3 828	7 163	24	439 627
GC, 4 clones	100	10^{-6}	4 461	8 190	39	579 287

run 10 times. The experimental setup was created in this way with the purpose of an immediate observation upon the presence or absence of an enhancement targeted by the better exploration.

The numerical results for the Six-hump camel back function are outlined in Table 4.9. *Mutations* represent the mean number of times out of ten runs when mutation was applied. It is obvious that when having more clones, mutation is applied more frequently and, at the same time, there are more evaluations for the mutated clones; evaluations are less in the rest of the algorithm, because the convergence process is faster. *Rec*, *gen* and *evals* represent the mean values obtained after 10 runs for the recombinations, number of generations and number of fitness evaluations, respectively.

As the number of fitness evaluations was increasing very fast depending on the number of the clones, experiments are carried out only for two clones, at the most. The accuracy refers to the global and the local optima, respectively.

GC with cloning was applied for Schaffer function using from one to four clones. In Table 4.10, one can clearly see that, in GC with crowding, mutations perform along the entire search process, while by using one or two clones within the initial proposed technique, results are considerably improved by reintroducing recombination.

The results by the application of the GC with cloning technique on the Himmelblau function are illustrated in Table 4.11. Again, the inclusion of the cloning mechanism makes the technique reach the solutions with the same accuracy as crowding GC, but with a lower expense.

It is mostly the Schaffer function optimization that underlines the major difference between crowding GC and that which includes the cloning procedure; this leads to the conclusion that when having very close optima and fine tuning is necessary, cloning within crowding GC is a very

Table 4.11: Results obtained for the Himmelblau function in 10 runs using GC with cloning.

Technique	SR	Accuracy	Mutations	Rec	Gen	Evals
Crowding GC	100	10^{-4}	771	619	189	104 033
GC, 1 clone	100	10^{-4}	1 456	642	185	98 378
GC, 2 clones	100	10^{-4}	2 176	702	184	100 737
GC, 3 clones	100	10^{-4}	2 894	720	187	102 459
GC, 4 clones	100	10^{-4}	3 696	681	190	101 936

good alternative.

4.6 Summary

In this section, two viable alternatives to the original GC technique are proposed. A crowding procedure is suggested for the integration of the offspring obtained after recombination, fact that also changes the way individuals are selected: They cannot each be considered in turn anymore, like in the original GC, because some might vanish prior to that, but they are instead randomly selected from the continually changing population. The goal of the technique was to maintain the ability of the original GC technique to find and preserve the global/local optima and additionally speed up the process. The experiments on function optimization proved that this is possible and happens especially for highly multi-modal problems and larger number of dimensions.

The second presented approach is constructed on the basis of the first one and it presumes the variation of the population size not only by merging, but also by adding new candidate solutions when subpopulations contain only single individuals. Clones of the single individuals are introduced only as long as fitter solutions may be found, therefore the danger of an uncontrolled population growth cannot exist: The introduced clones suffer mutation and, in case the offspring is not better than the parent, they do not enter the population; thus subpopulations can grow only until the optimum is reached. On the other hand, merging is also responsible for continually reducing the population.

4.7 Future Work

The major weakness of the GC techniques lies in the need to find proper values for mating, merging and replacement radii. Radii self-adaptation is one of the main goals of future work. Another goal would be to try a non-generational selection for replacement strategy instead of the one that is used and see whether is performs better or worse. A third goal would be to test if GC technique can be combined successfully with birth surplus-driven selection schemes as known from evolution strategies.

Chapter 5

Genetic Chromodynamics for Classification

5.1 Objectives of this Chapter

Current chapter aims to seek the means to use multi-modal techniques towards the solving of real-world tasks such as the development of simple and efficient classification techniques. In this respect, it starts with a short presentation of the concepts involved within the design of an evolutionary classifier, in order to subsequently introduce, in section 5.3, a recently developed classification technique based on the previously addressed GC; differences between a classical evolutionary classifier and the new one are properly outlined. All the necessary steps in applying the new classifier to a categorization problem are presented in detail [Stoean, 2004b], [Stoean, 2004c]; then, GC is replaced by the novel crowding variant which previously proved to be more efficient when dealing with multidimensional problems as it is also the case with classification. The new classifier holding the crowding GC mechanism is applied for two benchmark data sets and results demonstrated to be very encouraging [Stoean et al., 2005c], [Stoean and Dumitrescu, 2006], [Stoean and Dumitrescu, 2005c].

Finally, ideas of the way in which proposed classifier might be improved are discussed.

5.2 Other Evolutionary Classifiers

An evolutionary classifier represents a machine learning system that uses an EA as a rule discovery component [Dumitrescu, 2000a], [Michalewicz, 1996]. The rules (or *productions*, which are simple *if-then* rules) represent a population that is evolved by an appropriate EA; the rules cover the space of possible inputs and are evolved in order to successfully be applied to the problem to be solved – the problems may range from data mining to robotics. On a broader sense, an evolutionary classification technique is concerned with the discovery of if-then rules that reproduce the correspondence between given samples and corresponding classes. Given an initial set of training samples, the system learns the patterns, i.e. evolves the classification rules, which are then expected to predict the class of new examples. An if-then rule is imagined as a first-order logic

implication where the condition part is made of attributes and the conclusion part is represented by the class.

There are two important families of evolutionary classifiers: The Pittsburgh and Michigan approaches. In a Pittsburgh-type evolutionary classifier [Michalewicz, 1996], each individual represents an entire set of rules. The individuals compete among themselves and only the strong ones survive and reproduce. The Pittsburgh approach uses a typical EA for the learning problem. What remains to be solved is the representation problem and the way individuals adapt to their environment. Usually, within a chromosome there also appear operators from propositional logic, like disjunction and/or conjunction.

In a Michigan-style evolutionary classifier [Holland, 1986], each individual of the population represents a unique, distinct rule, so the EA evolves a set of rules; the population represents the rule set needed to solve the problem. The goal here is not to obtain the best individual, but to find the best set of individuals (rules) in the end of the algorithm. Usually, chromosomes representation is divided into two parts – one is the condition part and contains the values for the attributes that appear in the *condition* of the rule and the other part consists of the conclusion of the rule. A *credit assignment system* is used in order to *reward* the better rules or, at the same time, to *penalise* the worse ones. When new entities (rules) enter the population through mutation and/or recombination, usually crowding methods are utilised in order to introduce them; in this way, they replace only very similar individuals in the population.

On a different level, by drifting away from the strict meaning of an evolutionary classifier, other approaches to classification that involve EAs can be mentioned. An automated classification by means of evolutionary programming, with the purpose of determining both the optimum number of classes and membership, is discussed in [Luchian et al., 1994]. Hybridization with neural networks [Yao and Liu, 1997] or support vector machines [Stoean et al., 2007d], [Stoean et al., 2007c] are envisaged in the same respect.

Both classical evolutionary classifiers are somewhat complicated, one due to the intricate chromosome representation, while the other because of the demanding credit assignment system. As a consequence, section 5.3 presents a novel evolutionary classifier based on the Michigan approach which, instead of using a typical EA accompanied by the necessary credit assignment system, employs the GC technique described in chapter 3, resulting in a simpler alternative. The new classifier is outlined in application to the difficult problem of text categorization.

5.3 Text Categorization

The texts that are considered for classification are e-mail messages and the two classes of categorization represent unsolicited, commercial e-mails (that will be called *spam* in current chapter) on the one hand, and regular e-mails (also called *ham*), on the other hand. For more details about the considered data set, see Appendix B, section B.4.

Spam e-mails represent big trouble for common Internet users and cost big companies billions

of dollars annually because of employees loss of productivity. There are laws against spam, but nobody knows how to enforce them. It is up to the Internet users to get rid of this problem. Thus, good filters have to be built for labelling e-mails, based on their content, as being either spam or non-spam.

Some efficient filters that have been built so far generally use Naive Bayes engines [Graham, 2002], boosting algorithms [Nicholas, 2003] or pure machine learning techniques [Stoean, 2004b], [Stoean, 2004a].

5.3.1 Keywords Extraction

An e-mail is regarded as a bag of words; an important problem to be solved lies in obtaining a subset of these words that can characterize that e-mail and, implicitly, the category it has assigned.

In conclusion, special keywords have to be extracted for each one of the two categories; they are, of course, taken from e-mails in the training set. For a category, the keywords are taken to be words that appear often in e-mails from that category and rarely in e-mails from the opposite category. To accomplish that, weights for each word appearing in e-mails from the training set with respect to their categories (spam or non-spam) are computed.

First of all, some pre-processing needs to be done to all e-mails, be that they are from the training or test set; that means punctuation and HTML tags are removed and each remaining word is reduced to a word-stem[1].

Each e-mail will be further on represented as two vectors, one containing the word-stems remained after pre-processing, taken only one time, and the other one containing the number of occurrences of each word-stem in that e-mail, divided by the total number of word-stems the considered e-mail has [Stoean, 2004b], [Stoean, 2004a], [Stoean, 2004c]. This representation has to be obtained for all e-mails in the test collection.

Subsequently, one has to compute how *important* word-stems are for a category with respect to the other category; in order to achieve that, for each category, all word-stems in e-mails from the training set that have that category assigned are gathered in a large vector of word-stems. The weights of each word-stem from these e-mails are summed and thus, the second vector, containing weights for word-stems, is obtained.

Same representation has to be obtained for both categories, therefore the following vectors are constructed [Stoean, 2004b], [Stoean, 2004a], [Stoean, 2004c]:

$$WS = (ws_1, ..., ws_p) \text{ and } OS = (os_1, ..., os_p)$$

for *spam*, and

$$WH = (wh_1, ..., wh_q) \text{ and } OH = (oh_1, ..., oh_q)$$

[1]A word-stem is obtained from a word by cutting its suffixes and prefixes; a word-stem represents the root of the word. For instance, the word-stem for the word removing is remov.

for $non - spam$.

Of course, the number of common terms of the two categories is high: these terms have to be penalized, as they are not very much specific to only one of the two categories. That is the reason why their weights are modified for each category with respect to the weights of the same terms in the other category [Stoean, 2004b], [Stoean, 2004c] (5.1).

$$os'_j = \frac{os_j}{1 + oh_k} \qquad oh'_k = \frac{oh_k}{1 + os_j} \tag{5.1}$$

where $ws_j = wh_k$, $j = 1, ..., p$ and $k = 1, ..., q$.

Only the most important n word-stems for each of the two categories are considered: for each category, they are taken to be the n word-stems that have the highest values for their weights with respect to the weights of all word-stems in that category. Therefore, $2n$ word-stems are considered, n from spam and n from the non-spam category. These *keywords* are further on used to represent e-mails in the training and test set ([Stoean et al., 2004a], [Stoean et al., 2004b], [Stoean et al., 2005b]).

5.3.2 Genetic Chromodynamics Approach to the Spam Filtering Problem

Let m denote the number of e-mails in the training set and let v be their set, $v = \{v_1, ..., v_m\}$. Furthermore, let D be the interval from 0 to the average weight of the word-stems in the training set.

Each individual c represents a rule; it is encoded as a list of real numbers of the following form (5.2).

$$(a_1, ..., a_{2n}) : b, \tag{5.2}$$

where $a_1, ..., a_n$ represent the weights of the spam keywords and $a_{n+1}, ..., a_{2n}$ the weights of the non-spam keywords. b represents the outcome and thus it is either 0 (spam) or 1 (non-spam). An individual is encoded thus in the same manner as all e-mails in the test collection, after keywords extraction.

An individual gives the threshold behind which an e-mail can be labeled as either spam or non-spam.

Initial population

The initial population represents the initial set of rules. Let us denote by s its initial size.

The values of the genes are obtained by uniformly generating random numbers in D. Taking the average of the weights of the word-stems as the second extremity of the interval D appeared to be a better choice than the upper bound of the domain of weights, because there are very few maximum values; these would have made the system search also in those few spaces that contained high values and thus consume too much time just to figure out in the end that these

values could not have become thresholds of decision.

Fitness assignment

Let $x = (a_1, ..., a_{2n}) : b_1$ and $y = (c_1, ..., c_{2n}) : b_2$, where $b_1, b_2 \in \{0, 1\}$ two entities of the form (5.2). Manhattan distance is chosen to compute the difference between x and y:

$$d(x, y) = \sum_{i=1}^{2n} |a_i - c_i|$$

The fitness evaluation of a given chromosome c minimizes the distance between the weights of that chromosome and the weights of the selected keywords from all e-mails in the training set that have the same label as the chromosome c. At the same time, the distance between the weights of c and the weights of the keywords from all e-mails in the training set that have the opposite label with respect to chromosome c is maximized.

Let $c = (a_1, ..., a_{2n}) : b$ be the current chromosome.

Suppose there are u e-mails in the training set that are labeled with b.
The following multiobjective problem has to be solved (P):

$f_1 : D^{2n} \to R$, $f_1(c) = \frac{\sum_{i=1}^{u} d(c, v_i)}{u}$ to be minimized

$f_2 : D^{2n} \to R$, $f_2(c) = \frac{\sum_{i=u+1}^{m} d(c, v_i)}{m-u}$ to be maximized

Problem (P) is solved through combining the objective functions f_1 and f_2 in a unique criterion function:

$$f_c(c) = f_1(c) + \frac{1}{f_2(c)} \tag{5.3}$$

One is led now to the minimization of the function in (5.3).

Mating selection operator

The mate for every chromosome is selected within its predefined the mating region. Proportional selection is used in this respect [Dumitrescu, 2000a], [Dumitrescu et al., 2000].

Variation operators

Intermediate crossover is chosen for the first operator [Dumitrescu, 2000a], [Dumitrescu et al., 2000].

As mutation is concerned, a gene oriented one is used. The gene to be mutated suffers a normal perturbation:

$a_i = a_i \pm ms \cdot N_i(0,1),$

where $i = 1, 2, ..., 2n$ and ms denotes the *mutation strength*.

Mutation does not apply to the gene representing the outcome.

Merging operator

The merging operator does not take into account the outcome. Now, it is possible, in the early generations, for chromosomes that are very similar to have different outcomes. As the fitness evaluation takes into account the quality of the chromosome (rule) for the classification task, be that the above mentioned situation happens, only the chromosomes with the proper outcome for the given weights will survive.

Stop condition

The algorithm stops when, after a predefined number of iterations, denoted by t, no new offspring is accepted in the population.

The last population gives the set of rules, which are optimal in number and each rule is optimal for its corresponding outcome.

Parameter values

The values for the involved parameters are given in Table 5.1. These values are experimentally chosen so that GC principles obey.

Table 5.1: Empirically determined parameter values.

Mating region	ms	Merging radius	t	s	n
0.6 * 2n	0.06	0.4 * 2n	100	100	15

The mating region is considered in this way so that the difference between the values of two chromosomes for each gene (keyword) be no higher than 10% of the maximum possible difference between them. The mutation strength is in connection to the mating region, as it is compulsory that the offspring does not fall out of the range of its parent. The merging radius is chosen such that the difference between the values of two chromosomes for each gene (keyword) be no higher than 6% of the maximum possible difference between them.

Table 5.2: GC classifier: Accuracy rates after 10 runs for spam filtering

Data set	Mean	Standard deviation
Non-spam	97.65	0.19
Spam	94.94	0.70
Overall	96.29	0.32

Resulting rules. Interpretation

The final population contains at least two chromosomes, one for each of the two categories. Therefore, at least two rules are finally obtained, one for each category. If there is more than one rule for a certain category then, when applying the rules for an e-mail, at least one of them needs to be satisfied so that the e-mail be labeled with that category. Consider a chromosome in the final population, with the outcome 1, representing thus a rule for non-spam e-mails. The chromosome has the following representation:

$$x = (a_1, a_2, ..., a_n, a_{n+1}, ..., a_{2n}) : 1$$

As the first n genes contain weights of the keywords for spam, only the last n weights are of interest. The rule gives us some minimum values for the weights of the keywords that have to be overtaken by the weights of the same keywords in an e-mail from the test set in order to label that e-mail with non-spam. It is not always the case that every single keyword appears in an e-mail from the test set. This is the reason one cannot compare each value of the weights of the keywords in a rule with each of the values of the weights of these keywords in an e-mail from the test set. Alternatively, the following sum is computed for the non-spam rule:

$$h_1 = \sum_{i=n+1}^{2n} a_i$$

All weights of the non-spam keywords that appear in an e-mail from the test set are also summed; if the obtained sum (denoted by h) is higher than h_1, then the e-mail is concluded to be a non-spam one. If there are more rules for non-spam e-mails, other sums $h_2, h_3, ..., h_j$ are also computed. In conclusion, for an e-mail to be labeled as non-spam, its sum h has to be higher than at least one of the h_j-s. Same goes for spam labeling, but this time taking into account only the first n genes. For all e-mails in the test set, both types of rules are applied and each one of them is labeled as either spam or non-spam.

5.3.3 Experimental Results

As an immediate validation, the obtained rules, at least two (one for each outcome), are applied to the test set and results after ten runs are shown in Table 5.2:

The rule set correctly predicted the category for 96.29% of all e-mails, and, differentiating, rightly categorizing 97.65% of non-spam and 94.94% of spam, giving a failure of 3.71% e-mails which are not placed anywhere.

The classification task is successfully achieved as none of the e-mails in the test set are classified both as spam and non-spam. The fact that the failed e-mails are not labelled in any way is a very good aspect, mainly because no good e-mail is labelled as spam. Obviously, everyone would prefer the presence of spam e-mails in his/her account to the loss of good e-mails to spam: this is the reason it was decided to label the non-classified e-mails as non-spam. In this manner, none of the good e-mails would be lost and the percents would change to 100% non-spam correctly classified and 94.94% spam correctly categorized. Therefore, this leads to an overall result of 97.47% correctly classified e-mails.

A binary encoding is also tried in [Stoean et al., 2004b], but it failed to give results as accurate as the real encoding.

5.4 Crowding Genetic Chromodynamics Classifier

GC with the embedded crowding technique, which previously demonstrated its superiority when targeting high-dimensional problems, is now tested within proposed classifier and applied for two data sets, i.e. Pima-Indian Diabetes and Fisher's iris [Stoean and Dumitrescu, 2005c], [Stoean et al., 2005c], [Stoean and Dumitrescu, 2006], [Stoean and Dumitrescu, 2005b], [Stoean et al., 2005d]. No replacement or deletion of involved data is undertaken. More information about both data sets can be found in Appendix B.

5.4.1 Diabetes Disease Diagnosis

Each individual encodes an if-then rule. An individual therefore contains nine genes, one for each attribute and one for the outcome; first eight genes are real valued, while the last is a binary one and it gives the output of the chromosome (conclusion of the rule encoded). Consequently, the condition of the rule is a conjunction of personal data and symptoms and its conclusion is the diagnosis.

The rules (individuals) are evolved against the training set. The fitness of an individual is computed as its distance to all patients in the training set that have the same outcome. The aim is to minimize distances, conceiving thus good rules for the patients diagnosis. A rule is considered of high-quality if it matches the condition part of the data in the training set with the same outcome as itself.

Having an individual $x = (x_1, x_2, ..., x_n)$ and a patient from the training set $p = (p_1, p_2, ..., p_n)$, the distance between c and p is computed by the same Manhattan distance, only this time it is normalized (5.4):

$$d(x,p) = \sum_{i=1}^{n} \frac{\mid x_i - p_i \mid}{b_i - a_i} \tag{5.4}$$

where a_i and b_i represent the lower and upper bounds of the i-th attribute. As the values for the attributes belong to different intervals, the distance measure has to refer to the interval

bounds. n is equal to 8 in this particular case.

Intermediate recombination is used, with the coefficients biased by the fitness of the two parents involved. Mutation is with normal perturbation. Values for the parameters of the EA are specified in Table 5.4.

dif_i denotes the difference between the bounds of the interval corresponding to attribute i. Replacement radius is taken equal to the mating radius for both applications. The stop condition of the algorithm is given by a number of generations that can pass without any improvement for the solutions; this value is considered to be 10.

The ratio between training and test sets is set to 75%-25%, as established by Prechelt in [Prechelt, 1994] with respect to the diabetes task. Three kind of tests are conducted with different possibilities of choosing the data that would go into training and test, respectively. The two sets are obviously disjoint.

First way of splitting data is by doing test-sample cross-validation. The first 75% of the cases made the training set and the last 25% composed the test set, according to [Prechelt, 1994]. The obtained mean accuracy for the test set in 100 runs is 75.06%.

Second, another test is done according to rules of splitting that should be used for this data set, as established by Prechelt's rules in [Prechelt, 1994]. The data set is sequentially split into 75% training - 25% test to give 4 different combinations of these two sets, *i.e.* first 75% data for training – last 25% for test, first 25% data for test – last 75% for training, first 50% data for training – next 25% for test – last 25% for training again, first 25% data for training – next 25% for test – last 50% for training again. The algorithm is subject to 100 trials again. The mean accuracy obtained for the test set is 69.672%.

Last, random cross-validation is performed, *i.e.* the training set containing 75% data and the test set consisting of 25% data are randomly generated in each run. The algorithm is applied 100 times and the obtained mean accuracy is of 69.515%.

However, in many tests, it was noticed that when the chromosome pool still has four individuals left and has not converged yet, a higher accuracy of 80% is obtained. This leads to the idea that in the structure of each of the two obvious clusters two other subclusters are included. Thus, with the best instead of last accuracy, better results can be obtained.

Results obtained by crowding GC and those obtained by other techniques applied to the same classification problem (a neural network with Prechelt's rules in [Smithies et al., 2004] and an evolved neural network in [Yao and Liu, 1997]) are outlined in Table 5.3.

5.4.2 Iris Plants Identification

There are two ways of dividing the database into training and test sets: on the one hand, two thirds of each of the three classes are considered as training set and the rest as test set; on the other hand, the training set is randomly chosen and the test set consisted of the remaining instances. In the first case, the three classes are equally distributed into training and test sets.

Chromosome representation is similar to the one from the diabetes diagnosis problem, where

Table 5.3: Results of comparable techniques for the diabetes task in relation to crowding GC.

Technique	Runs	Accuracy (%)
Crowding GC with test-sample cross-validation	100	75.06
Crowding GC with Prechelt's rules	100	69.672
Crowding GC with random cross-validation	100	69.515
Crowding GC best accuracy (4 rules)	100	80
Neural Network (NN) with Prechelt's rules	100	65.5
Evolved NN with test-sample cross-validation	30	77.6
Evolved NN best result	30	80.7

the first four genes correspond to the attributes of an iris plant and the last one embodies its class. Same distance as in (5.4) is used to compute differences between individuals and same variation operators are considered; the values that are used for the parameters of the EA are given below in Table 5.4.

Table 5.4: Parameters of the crowding GC technique for both classification problems.

Pop. size	Mut. probability	Mut. strength	Mating radius	Merging radius
100	0.4	$dif_i/100$	0.3	0.03

The final result of the evolutionary classifier has to consist of at least three rules (individuals), that is at least one for each class. The accuracy for the first mode of fixing the training and test sets varies between 94% and 98%, while for the second way of selecting the two disjoint sets the accuracy ranges from 88% to 96%. Besides accuracy from the third row in Table 5.5, where the percent is the best found in 100 runs (obtained even during these runs and not necessarily at the end of them), all other values in the third column are obtained by computing the average for the final accuracies at the end of each of the 100 runs.

Unfortunately, in some literature approaches that used this database there were no clear descriptions of the way training and test sets were chosen, so direct comparisons between their results and those obtained by GC with crowding are not very accurate. For instance, in [Veenman, 1996],

Table 5.5: Results of different techniques for the iris plants recognition in comparison to crowding GC

Technique	Runs	Accuracy (%)
Crowding GC with equally distributed cross-validation	100	95.76
Crowding GC with random cross-validation	100	92.84
Crowding GC best accuracy instead of last	100	98
Genetic programming ([Veenman, 1996])	100	92.7
NN BP with random cross-validation	10	93.2
NN CCA with random cross-validation	10	92.6
NN modified CCA with random cross-validation	10	97

a genetic programming model is tested on the Iris database, while in [Yang and Honavar, 1991], some neural networks techniques (backpropagation algorithm, denoted by BP, and cascade-correlation algorithm, CCA) are applied in this respect; results are outlined in Table 5.5.

5.4.3 Observations

Regarding the diabetes classification problem, one possibility is to build separate rules for different decades; therefore, rules are built depending on the *age* attribute. The natural explanation is that people that are young are very likely to have different reasons to suffer of diabetes than the others or they may have a different type of diabetes.

Experiments are carried out and results showed indeed an important increase in the classification accuracy: 77.08% of the patients in the test set are correctly classified, using cross-validation, in comparison to 75% when plain crowding GC is used. Two decades are considered, so that all patients of age between 21 and 51 are in the first decade and patients of age between 52 and 81 in the second one. Consequently, (at least) four rules are obtained, that is (at least) two for each decade: one for the ill patients and one for the healthy ones.

Another possibility is to weight each of the eight attributes. In this manner, not all attributes have the same importance; the degrees of importance are detected either using a typical genetic algorithm or they may be evolved together with the values for attributes using crowding GC. In the first case, the crowding GC method provided the rules to be applied, that is the values for the eight attributes in each of the two cases; then, independently, a typical genetic algorithm evolved eight values, each between 0 and 1, in order to find the proper weights for the attributes of the rules. The weights are evaluated with respect to the accuracy obtained by application of the rules to the training set – so the higher the accuracy on the training set, the better the weights are. This technique is tested in both situations, with two decades and without them. Without decades, the method provides accuracy around 77% [Stoean and Dumitrescu, 2005c]; what is unusual is that, when using decades, the result is not very stable – it varies from 74.48% to 79.68%. Evolving the weights together with the values for the attributes using crowding GC did not offer good results.

5.5 Summary

The chapter starts with a brief presentation of classical evolutionary classifiers.

GC is then integrated into such a classification technique which is applied for spam filtering. Each component of the classifier is explained, beginning with text preprocessing and continuing with the evolutionary components. Conducted experiments and competitive obtained results are outlined.

In a subsequent episode, a classifier with the crowding GC as engine is presented. This variant has been demonstrated to exhibit a superior performance on multidimensional tasks and consequently the new version of the classifier is applied to two classification problems; the results sustain its employment.

5.6 Future Work

As spam e-mails contain more HTML tags than non-spam e-mails and, at the same time, obviously more classification problems appear with the labeling of spam e-mails, a solution for a more accurate e-mail categorization might rely on a better HTML processing.

For the same GC based classifier presented first, the impact of a size for n higher than 15 is to be studied in future work. Moreover, the study of the ideal size for n has to take into account not only accuracy but also the amount of computing time needed.

A further improvement (for the technique this time) that is still to be done is to adjust the way merging takes place: It should occur only if the obtained set of rules provides a better accuracy on the training set, because, as it is now, potentially promising configurations frequently disappear.

Chapter 6

Coevolution for Classification

6.1 Objectives of this Chapter

The aim of the chapter is to follow a slightly different learning direction and present two other newly developed classifiers [Stoean et al., 2006a], [Stoean et al., 2006c], [Stoean et al., 2008a], [Stoean, 2007] that are based on both the cooperative and the competitive coevolution.

Cooperative coevolution for classification is inspired from the framework proposed in [Potter and Jong, 1994], [Wiegand, 2003]; therein, it is applied for function optimization. Competitive coevolution for classification, on the other hand, draws its roots from the paradigm proposed by [Paredis, 1998].

The chapter starts with an overview of the original cooperative and competitive approaches and then the proposed classifiers are described. Experiments are then driven on several data sets and the obtained results are reported. The chapter encloses with conclusions and ideas for future research directions.

6.2 Overview

The coevolution technique is inspired from the interactive process that occurs between species in nature: on the one hand, species have to fight for the same resources, meaning that they are in competition for a certain goal, and, on the other hand, different species collaborate for a specific purpose. Consequently, two kinds of artificial coevolution are introduced: one is called *competitive*, the other one is *cooperative*.

In competitive coevolution, the evaluation of an individual is determined by a set of competitions between the current individual and other individuals, while in cooperative coevolution, collaborations between two or more individuals are necessary in order to evaluate one complete solution.

The main components of the cooperative coevolution paradigm are next described, together with the successful application of the technique for the optimization of several functions with multiple local optima. The competitive coevolution paradigm description follows afterwards.

6.2.1 Cooperative Coevolution

Individuals in nature evolve by means of adaptation to the environment, part of which consists of other living beings, in particular different groups or species. From this viewpoint, evolution is actually coevolution. Coevolution can be competitive, cooperative or both. Similarly, in the evolutionary computational area, interest has recently grown towards the extension of the powerful EAs to coevolutionary architectures. They are interesting indeed because they bring along a new idea for the fitness evaluation of an individual, i.e. in relation to the other individuals in the population.

The general idea of the cooperative coevolution approach is outlined in chapter 3. Herein some more details regarding attributes specific to the cooperative coevolution are mentioned, as they are further on used in the approach for classification.

When building a cooperative coevolutionary algorithm, there are three attributes [R. P. Wiegand and Jong, 2001] regarding selection that have to be decided on:

1. *Collaborator selection pressure* refers to the way individuals are chosen from each population in order to form complete solutions to the problem, i.e. pick the best individual according to its previous fitness score, pick a random individual or use classic selection schemes in order to select individuals from each of the other populations.

2. *Collaboration pool size* represents the number of collaborators that are selected from each population.

3. *Collaboration credit assignment* decides the way of computing the fitness of the current individual. This attribute appears in case the *collaboration pool size* is higher than one. There are three methods for computing this assignment:

   (a) *Optimistic* - the fitness of the current individual is the value of its best collaboration.

   (b) *Hedge* - the average value of its collaborations is returned as fitness score.

   (c) *Pessimistic* - the value of its worst collaboration is assigned to the current individual.

The explanation for the pessimistic assignment is that it might be best to use a *safe* credit assignment that rewards an individual only as its weakest collaboration. Although in experiments driven for function optimization [R. P. Wiegand and Jong, 2001], both the pessimistic and hedge strategies consistently resulted in significantly poorer performance, they proved to be very effective in the approach proposed for classification in current work.

Algorithm 12 demonstrates the modality of evaluation of an individual c with respect to the three mentioned attributes. We presume that a maximization problem is considered.

In order to evaluate an individual c from a certain population, a number of complete potential solutions are formed according to the chosen collaboration pool size. In order to aggregate a solution, collaborators from each population different from that of c are selected through a certain strategy (collaboration selection pressure). Each solution is evaluated according to the objective

Algorithm 12 Fitness evaluation within cooperative coevolution

Require: A current individual c

Ensure: The fitness evaluation of c

```
begin
for each i = 1, 2, ..., collaboration pool size (cps) do
    select one collaborator d_j, j = 1, 2, ..., number of species from each population different from that of
    c;
    form a complete potential solution;
    compute the fitness f_i of the solution in the terms of the objective criterion;
end for
if Collaboration credit assignment = Optimistic then
    evaluation ← max_{i=1}^{cps}(f_i);
else
    if Collaboration credit assignment = Pessimistic then
        evaluation ← min_{i=1}^{cps}(f_i);
    else
        evaluation ← avg_{i=1}^{cps}(f_i);
    end if
end if
return evaluation
end
```

function of the current problem. Once all candidate solutions are gathered and assessed, the preferred type for the collaboration credit assignment decides the value that will be returned as the performance of the individual c.

Cooperative coevolution is introduced as an alternative evolutionary approach to function optimization [Potter and Jong, 1994]. For this task, one considers as many populations as the number of variables of the function, i.e. each variable represents a component of the solution vector and is separately treated using any type of EA. Several functions with multiple local optima and one global optimum are considered and the cooperative coevolutionary algorithm proved to be effective [Potter and Jong, 1994], [Wiegand, 2003].

The cooperative coevolutionary technique has been recently successfully applied to develop a rule-based control system for agents; two species were considered, each consisting of a population of rule sets for a class of behaviours [Potter et al., 2001].

Other classification evolutionary models for coadapted components are Holland's classifier system [Holland, 1986] and the REGAL system [Giordana et al., 1994], where stimulus-response rules in conjunctive form (such as in present approach) are evolved by EAs. In [Holland, 1986], cooperation is achieved through a *Bucket Brigade* algorithm that awards rules for collaboration and penalizes them otherwise. In [Giordana et al., 1994], problem decomposition is performed by a selection operator, complete solutions are found by choosing best rules from each component, a seeding operator maintains diversity and fitness of individuals within one component depends on their consistency with the negative samples and on their simplicity [Potter and Jong, 2000].

To the best of our knowledge, there has been no attempt in applying cooperative coevolution to classification based on individuals that encode simple conjunctive if-then rules in first order logic.

6.2.2 Competitive Coevolution

Within the competitive model [Paredis, 1999], the complementary evolution between species is achieved through an inverse fitness interaction process. This implies that success attained on one side is regarded as failure among the individuals of the other side; the latter species will have to react in order to maintain its chances of survival.

Competitive coevolution represents a predator-prey complex: The strong evolutionary pressure determines the prey to defend itself better while, as a response, the predator develops better attacking strategies. This results in a stepwise adaptation and complexity of involved species. Therefore, the competitive interaction between species represents the force that drives evolution forward.

Accordingly [Paredis, 1998], one species corresponds to certain tests a solution must satisfy and the other to the potential solutions for the given task. Competition is achieved through encounters between one individual from the tests population and one from the solution species. The two selected individuals are checked against each other and, if the solution passes the test, then the former is rewarded while the latter is penalized; if it fails, credits are assigned in a reverse manner. Moreover, each individual has a history of its encounters which embodies the penalizations/rewards it has received. The fitness of the individual is computed on this basis, as the sum of its most recent behaviours (successes/failures).

An important remark is that, since tests are a priori defined, it is only the population of potential solutions that evolves; the opposite species contains the same individuals (tests) until the end of the evolutionary process. The only fluctuation that appears within the latter population solely regards the ranking of the individuals according to fitness (their satisfiability hardness). It must be however noted that, in certain cases when tests cannot be exhaustively given, the tests population may also evolve.

Canonical competitive coevolution can be described as in Algorithm 13.

The initial evaluation of the individuals in both populations is based on the results of random encounters between solutions and tests. When such an encounter takes place, only the current individual is rewarded / penalized without the inverse score attribution for its competitor happening as well.

An evolution cycle is then entered. A predefined number of encounters between solutions and tests takes place. Those opposite individuals that meet are decided following a ranking selection. As a result, the fittest solutions and tests are more frequently involved in such "tournaments": The best performing solutions must prove their superiority more often, while, concomitantly, the algorithm focuses upon the most difficult tests. As soon as the reward/penalization is established for the two selected competitors, their corresponding history is updated: The score of the most recent encounter replaces that of the oldest one and evaluation is revised.

Algorithm 13 A canonical competitive coevolutionary algorithm

Require: A problem and a population of tests

Ensure: The final solutions population

```
begin
t ← 0;
randomly initialize solutions population Pop_Sol(t);
create history and evaluate individuals in Pop_Sol(t);
create history and evaluate individuals in Pop_Test(t);
while termination condition = false do
    t ← t + 1;
    for i = 1, 2, ..., number of encounters do
        select solution from Pop_Sol(t − 1);
        select test from Pop_Test(t − 1);
        obtain result from encounter between solution and test;
        update history and evaluation of solution according to result;
        update history and evaluation of test according to result;
    end for
    select two solutions from Pop_Sol(t − 1);
    apply variation operators to obtain one offspring;
    evaluate offspring;
    Pop_Sol(t) ← Pop_Sol(t − 1)
    insert offspring into Pop_Sol(t);
end while
return Pop_Sol(t)
end
```

After the considered encounters are finished, a single offspring is created. Two parents are selected according to the same selection scheme and recombination and mutation on the resulting solution are subsequently applied. A personal history of the offspring is created through a number of encounters equal to the defined history length. The tests are again selected according to a ranking scheme. Following such an encounter, only the history of the offspring is modified; unlike a standard encounter between a solution and a test, no simultaneous penalization/reward of the involved test is conducted. This stems from the simple reason that a mediocre offspring might lead to an unreliable change in the behaviour of the considered test. After the offspring is evaluated, it will replace the weakest individual in the solutions population.

During the entire evolutionary process, the tests population suffers no variation.

The two species thus evolve together, through the inverse fitness interaction mechanism: As soon as the potential solutions satisfy certain tests, the latter receive a weaker evaluation score which leads to omission from further selection. As a result, other more difficult tests are subsequently more often selected for tournaments, while the solutions must evolve to adapt to the new requirements that must be fulfilled.

The parameters that are associated with competitive coevolution are the history length of an individual (the number of meetings that provide a measure of its performance) and the number of encounters between solutions and tests within an evolutionary cycle.

The importance of the personal history is manifold [Paredis, 1999]. For one, it offers a continuous evaluation of an individual. Then, its partial nature leads to a major decrease in the computational expense of testing a potential solution against all the given tests, while it offers dynamics and keep of pace between the two species.

The competitive paradigm has been applied to a wide range of problems, i.e. path planning [Paredis and Westra, 1997], constraint satisfaction [Paredis, 1994a] and classification. As classification is concerned, known techniques involve the evolution of neural networks [Paredis, 1994b], decision trees [Siegel, 1994], cellular automata rules [Juille and Pollack, 1996], [Paredis, 1997] and the use of genetic programming for the problem of intertwined spirals [Juille and Pollack, 1998]. Again, it has to be stated that to the best of our knowledge, the competitive coevolution between simple if-then rules and the training set has not been achieved yet.

6.3 Cooperative Coevolution Approach to Classification

The solution to the classification problem is again imagined as to be represented by a set of rules that contains at least one rule for each class. Therefore, the decomposition of each potential problem solution into components is performed by assigning to each species (population) the task of building the rule(s) for one certain class [Stoean et al., 2006a], [Stoean et al., 2006c], [Stoean et al., 2008a], [Stoean et al., 2007b]. Thus, the number of species equals the number of outcomes of the classification problem. A rule is one more time considered to be a first logic entity in conjunctive form, i.e.:

$$\text{if } (a_1 = v_1) \wedge (a_2 = v_2) \wedge ... \wedge (a_n = v_n) \text{ then class } k$$

where $a_1, a_2, , ..., a_n$ are the attributes, $v_1, v_2, , ..., v_n$ are the values in their domain of definition and $k = 1, 2, ..., p$.

6.3.1 Training Stage. The Evolutionary Algorithm Behind

As the task of the cooperative coevolution technique is to build p rules, one for each class, p populations are considered, each with the purpose of evolving one of the p individuals.

Representation

Each individual (or rule) c in every population follows the same encoding as a sample from the data set, i.e. it contains values for the corresponding attributes, $c = (c_1, c_2, ..., c_n)$. As already stated, individuals represent simple if-then rules having the condition part in the attributes space and the conclusion in the classes space. Within the cooperative approach to classification, an individual will not however encode the class, as all individuals within a population have the same outcome.

Initialization

The values for the genes of all individuals are randomly initialized following a uniform distribution in the definition intervals of the corresponding attributes in the data set.

In case the considered data set is normalized, the values for the genes of the individuals are initialized in the interval [0, 1], again following a uniform distribution.

Fitness Evaluation

In order to measure the quality of a rule, this has to be integrated into a complete set of rules which is to be subsequently applied to the training set. The obtained accuracy reflects the quality of the initial rule. Of course, the value of the accuracy very much depends on the other rules that are selected in order to form a complete set of rules: For a more objective assessment of its quality by means of the accuracy value, the rule is tested within several different sets of rules, i.e. different values for the collaboration pool size are considered. For a deep research regarding the effect of the cooperative parameters on the results, see [Stoean, 2006].

The value for the collaboration pool size parameter is further on denoted by *cps*. For evaluating an individual from a certain population–that is a rule of a certain outcome–a collaborator from each of the other populations is selected n times according to the collaborator selection pressure choice. Every time, the set of rules is applied to the entire training collection. Obtained accuracy represents the evaluation of the current individual. The fitness of an individual c may be given by the best of the *cps* acquired accuracies (optimistic assignment), by the worst one of them (pessimistic assignment) or by the average of all *cps* accuracies (hedge assignment). Algorithm 14 describes the way evaluation takes place in these cases [Stoean et al., 2006d], [Stoean et al., 2006a].

Algorithm 14 Fitness evaluation of an individual c by means of either optimistic, pessimistic or hedge collaboration credit assignment

Require: A current individual c

Ensure: The fitness evaluation of c

```
begin
for i = 1 to cps do
    correct_i = 0;
    select a random collaborator from each population different from that of c according to the collaborator
    selection pressure parameter;
    for each sample s in the training set do
        find the rule r from the set of all collaborators that is closest to s; found class for s = r's class;
        if found class for s = real class of s then
            correct_i = correct_i + 1;
        end if
    end for
end for
if optimistic then
    success = max_{i=1}^{n}(correct_i)
else
    if pessimistic then
        success = min_{i=1}^{n}(correct_i)
    else
        success = avg_{i=1}^{n}(correct_i)
    end if
end if
accuracy = 100 * success / number of training samples;
return accuracy
end
```

In addition to the classical cooperative coevolutionary ones, a novel type of assignment is proposed (Algorithm 15, [Stoean et al., 2006d], [Stoean et al., 2006a]). For each sample s in the training set, multiple sets of rules are formed and applied in order to predict its class. All rules within a set have different outcomes. Scores are computed for the sample s, for each of the possible outcomes in the following manner: when a rules set is applied to a sample, a certain outcome is established for it. The score of that outcome is increased by unity. Each of the *cps* sets of rules are applied to s. Finally, the class of s is concluded to be the class that obtains the highest score.

Algorithm 15 Score-based fitness evaluation for an individual c

Require: A current individual c
Ensure: The fitness evaluation of c
begin
for each sample s in the training set **do**
set the score for each possible outcome of s to 0;
end for
for $i = 1$ to *cps* **do**
select a random collaborator from each population different from that of c according to the collaborator selection pressure parameter;
for each sample s in the training set **do**
find the rule r from the set of all collaborators that is closest to s; increase the score of r's class for s by one unit
end for
end for
$success = 0$;
for each sample s in the training set **do**
if the real class of s equals the class that had the higher score for s **then**
s is correctly classified;
$success = success + 1$;
end if
end for
$accuracy$ = 100 * $success$ / number of training samples;
return $accuracy$
end

Independently of the chosen algorithm for calculating fitness evaluations, the distance between individuals and samples from the data set has to be once more computed when one decides which rule is *closer* to each sample in the training set. In the conducted experiments, normalized Manhattan is adopted as the distance measure (6.1). However, other distance measures may be as well employed, depending of the considered problem. Note that the distance does not depend on the class of the sample/individual.

$$d(c, x_i) = \sum_{j=1}^{n} \frac{\mid c_j - x_{ij} \mid}{b_j - a_j} \tag{6.1}$$

We denoted by $x_i = (x_{i1}, x_{i2}, ..., x_{in})$ a sample from the training set, while by $c = (c_1, c_2, ..., c_n)$ an individual (or rule). a_j and b_j represent the lower and upper bounds of the j-th attribute.

In both algorithms 14 and 15, the fitness of an individual is computed as the percent of correctly classified samples from the training set (variable *success* in the algorithms specifies the number of samples that are successfully labelled).

In Algorithm 15, situations may appear when, for a certain sample, there exist more classes that have the same maximum score. In this case, one class has to be decided and it is considered to choose the first one in the order of outcomes. As herein all combinations of rules count in the determination of accuracies, we might state that the new choice of assignment is closer to the classical hedge type.

Selection and Variation Operators

The selection operator presently discussed refers to the selection for reproduction within each population, not to the collaborators selection. Fitness proportional selection is employed, but any other selection scheme may be successfully applied.

Intermediate recombination is used. The obtained offspring individual replaces the worst of its two parents.

Mutation with normal perturbation is used for the experiments performed in current paper – a value of the gene i of an individual P is changed according to (6.2).

$$P_i = P_i + R \cdot (b_i - a_i)/ms, \tag{6.2}$$

where R is a random number with normal distribution, b_i and a_i are the upper and lower bounds of the i-th attribute in the data set and ms is the mutation strength parameter. As the domains for the values of the attributes in the data set have different sizes, we again have to refer to the size of the interval for each attribute when the values of the genes are perturbed through mutation. In case the data set is normalized, the way the value of the gene i is modified changes to (6.3).

$$P_i = P_i + R \cdot ms \tag{6.3}$$

We cannot imagine any obstacle for using any other recombination or mutation operators [Eiben and Smith, 2003].

Stop Condition

In the experiments, a fixed number of generations for the evolutionary process is set.

6.3.2 Cooperative Coevolution Parameters

In order to achieve the optimal configurations for the parameters of the cooperative approach, experiments are carried out as follows.

Concerning the *collaborator selection pressure* attribute, random selection is used, on the one hand, and, on the other hand, a fitness proportional scheme is employed.

All the three types of fitness *assignment* presented in Algorithm 14 together with the one based on scores are tested.

As for the *collaboration pool size*, the number of collaborators is varied in order to find the optimum balance between accuracy and runtime.

6.3.3 Test Stage. Rules Application

After the stop condition is reached, we dispose of p populations of rules that are evolved against the training set. In order to form a complete set of rules, an item from each population is chosen. The rules may be selected randomly, the best ones can be considered or a selection scheme can be used. In the last two cases, the final fitness evaluations of the individuals are taken into account. It is not always the case that, by selecting the fittest rule from each population, the best accuracy on the test set is obtained. Even if these best rules would give very good results on the training set, they may be in fact not general enough to be applied to previously unseen data.

In the conducted experiments, for a number of *cps* times, one rule from each population is randomly selected in order to form *cps* complete sets of rules. Each time, the rule set is applied to the test data in a similar manner to the fitness calculation in Algorithm 15 and the classification accuracy is acquired.

6.4 Competitive Coevolution Approach to Classification

Similarly to the cooperative approach for classification, the aim of the competitive classifier is to construct, based on a training set, a set of rules that model the data and which will be subsequently applied to the test set. Within proposed competitive approach, a population of tests is represented by the samples in the training data, while another population, that of solutions, will contain only the rules that are to be evolved.

6.4.1 Training Stage. The Evolutionary Algorithm Behind

Keeping the same notations as in the cooperative approach, the task in this case will be once more to build p rules, one for each class. Consequently, in order to form a solution to the classification problem, a complete set of rules has to be selected from the solutions population, which must therefore contain rules for every outcome.

Representation

The same representation for the individuals (rules) as in the cooperative approach is adopted. The only difference is that herein a better tracking of the class for each rule has to be kept. Since, in the cooperative case, the outcome of each rule is identified with the population class, in present methodology all rules, indifferent of the class they have, lie in the same population. As a result, this time the rules will also encode the class, i.e. $c = (c_1, c_2, ..., c_n \mid k)$, $k = 1, 2, ..., p$.

Initialization

As previously stated, at least p rules have to be obtained. Recombination will take place only between individuals with the same outcome, therefore, the size of the solutions population has to be of at least $2p$ individuals, i.e. differentiating two individuals per class. However, we only state here the minimum size of the rules population; a higher number of individuals would obviously bring a better covering of the search space.

Fitness Evaluation

For every individual and for every sample from the tests population, a *history* of the scores they obtained during encounters has to be constructed. The actual fitness evaluation of each individual/sample will be equal to the sum of all scores in their history.

The main question is: How are the scores given? When an encounter between a rule and a sample takes place, the distance (the same normalized Manhattan distance is used) between them is computed. The task for the rules is to be as similar as possible to the samples in the training set, therefore the aim is to minimize the distance between them and the samples from the training set with the same outcome. The score that is attached to a rule is given by the negative value of the distance; the maximum score a rule aims to attain is thus 0, meaning that the rule is identical to the sample it encountered.

Conversely, for a sample in the tests population, the actual value of the distance between it and the rule that it met is attached. As rules get closer to a certain pattern of samples, they will subsequently encounter other samples that have larger fitness values (and as a consequence higher chances of being selected for encounters) because they are very different from those rules. Thus, new fitter samples are continuously selected in order to adapt the rules so that they will resemble them too, i.e. evolve the solutions according to a high variety of tests.

Immediately after the initialization of the rules population, the fitness evaluations for both the rules and the samples have to be computed. In this respect, for each rule, a sample with the same outcome as its label is randomly selected and the encounter takes place: This has to be performed for a number of times equal to the history length. Each individual will thus posses a history and, as a result, an evaluation. In a similar manner, for a number of times equal to the history length, each sample from the training set will be considered and random individuals with the same outcome are selected in order to form the encounters that will complete their histories.

After the initial fitness calculation, each time an encounter takes place, solutions and samples are chosen from each population by means of ranking selection. As encounters take place only between solutions and samples with the same outcome, they will occur separately and in turn for each class (Algorithm 16, [Stoean et al., 2006d], [Stoean et al., 2006c], [Stoean et al., 2006a]). After the encounter, the new score is added to the history queue and the oldest score is removed, such that the same history length is maintained.

An individual that is obtained after the variation operators is evaluated as follows: for a

number of times equal to the history length, a sample with the same outcome as its own is selected using ranking selection and encounters take place. Its fitness may now be computed by summing all the scores in the history. It is then included in the population by replacing the individual with the worst fitness evaluation.

Algorithm 16 The competitive coevolution approach to classification

Require: A classification problem and the training samples
Ensure: The final rules population

```
begin
t ← 0;
randomly initialize solutions population Pop_Sol(t);
create history and evaluate individuals in Pop_Sol(t);
create history and evaluate individuals in Pop_Test(t);
while termination condition = false do
  t ← t + 1;
  for j = 1, 2, ..., number of classes do
    for i = 1, 2, ..., number of encounters do
      select solution labelled by j from Pop_Sol(t − 1);
      select test labelled by j from Pop_Test(t − 1);
      obtain result from encounter between solution and test;
      update history and evaluation of solution with -result;
      update history and evaluation of test with +result;
    end for
    select two solutions with class j from Pop_Sol(t − 1);
    apply variation operators to obtain one offspring;
    evaluate offspring;
    insert offspring into Pop_Sol(t);
  end for
end while
return Pop_Sol(t)
end
```

Selection and Variation Operators

In our experiments the ranking scheme is tried, as it is usually advised in the general framework of competitive coevolution.

As regards the variation operators, the same types of recombination and mutation as in the cooperative approach are employed. Recombination takes place only between individuals within the same class and, therefore, the offspring inherits the outcome of the parents. The mutation operator does not apply to the class gene.

Again, any other variation operators may be successfully tried.

Remark: Variation and replacement take place for every class in turn (Algorithm 16).

Stop Condition

Competitive techniques usually require more iterations than a canonical EA, as in each generation there is only one new descendant that enters the population. However, in our approach for classification, we apply the variation operators for individuals of every class in the same generation, a change that makes several individuals (descendants), i.e. p instances (one for each class), enter the population within that iteration.

The stop condition used refers to a fixed number of generations, just like in the cooperative approach.

6.4.2 Competitive Coevolution Parameters

There are two important parameters related to the competitive coevolution technique: The history length and the number of encounters that take place within one generation. The larger the values for both of them, the more accurate the fitness evaluation is for an individual/sample. Unfortunately, together with the raise in the values for either of the two, the runtime of the algorithm also increases.

The value for the number of encounters parameter directly depends on the population size of the two species: If there are many individuals in any of the populations, then a high value for the number of encounters have to be set in order to update the fitness evaluations of a great amount of the individuals.

A value that is too small for the history length parameter could make the fitness evaluation of an individual/sample change too drastically after each encounter and thus the fitness evaluation would not objectively reflect the quality of the individual/sample in contrast to the other population.

6.4.3 Test Stage. Rules Application

After the evolutionary process stops, one rule for every class is selected and these are applied to the test set. For each sample in the test set, the dissimilarity to each of the rules is computed. The found outcome of the sample is taken from the rule it resembles the most.

6.5 Experiments. Application to Real-world Problems

For each of the two coevolutionary approaches for classification, the same data sets are considered for experiments: Two data sets concerning benchmark classification problems coming from the University of California at Irvine (UCI) Repository of Machine Learning Databases, i.e. Wisconsin breast cancer diagnosis and iris recognition, are selected for reasons of validation and comparison. Besides, the former is a two-class instance, while the latter represents a multi-class task, which should reveal whether the classification techniques remain flexible and feasible with some increase in the number of outcomes.

Finally, a real-world data set, courtesy of the University Hospital in Craiova, Romania, is also considered with the purpose of testing and application on an unpredictable environment that is usually associated with raw data. For all grounds mentioned above, the selection of test problems certainly contains a variety of situations that is necessary for the objective validation of the new framework of coevolution in application to classification. More information about the considered data sets can be found in Appendix B.

In all conducted experiments, for each parameter setting, the training set is formed of randomly picked samples and the test set contains the rest of the samples. In order to prove the stability of the approaches, each reported average result is obtained after 30 runs of the algorithm.

6.5.1 Experiment 1: Cooperative Classification Validation

Pre-experimental planning: In preliminary experiments, different settings for the coevolutionary parameters are tested in order to verify their suitability for the classification problems. However, in these initial experiments, the technique is only tested on the breast cancer and iris data sets.

No major differences are observed between results obtained when different types of collaboration credit assignments are used: Slightly better results appeared to be achieved for the score-based fitness.

As it was expected, when the collaboration pool size value is increased, the runtime of the algorithm also raises. As concerning the results, they also seem to be improved to some extent by the increase of the value for this parameter. The technique had been tested from one up to seven collaborators.

As regards the collaborator selection pressure parameter, random selection is initially employed which drove the coevolutionary process to very competitive results. Then, the best individual from each population is chosen for collaborations, but, surprisingly, the obtained results are worse than in the case of a random selection pressure. The next step is that of using a selection scheme for choosing the collaborators. Proportional selection is employed and it proves to be efficient as results are slightly better than those obtained through random selection.

Task: To evaluate whether the cooperative approach produces viable results if compared to those obtained by other approaches (information on these is given in subsection 6.5.3) and how appropriate parameters are chosen.

Setup: The values for all the parameters are manually tuned. Table 6.1 contains the values for both the parameters of the EA and the coevolutionary ones. The population size refers to the number of individuals from each of the considered species. We only outline the values for which the best average results in 30 runs are obtained. In each of the runs, the training and test sets are formed from randomly selected samples.

The only normalized data are the ones regarding breast cancer.

For all considered implementations, we use fitness proportional selection, intermediate recom-

Table 6.1: Parameter values for the cooperative coevolution approach in application to real-world tasks

	Breast Cancer	Iris	Hepatic Cancer
Evolutionary Parameters			
Population size	100	150	100
Recombination probability	0.5	0.4	0.5
Mutation probability	0.6	0.6	0.6
Mutation strength	0.01	150	100
Generations	120	200	100
Cooperative Parameters			
Collab. pool size	3	3	5
Collab. selection pressure	proportional	proportional	proportional
Collab. credit assignment	hedge	hedge	worst

Table 6.2: Average results after 30 runs for the cooperative coevolution approach in application to real-world tasks

Data set	Average accuracy (%)	Standard deviation (%)
Breast cancer	94.5	1.8
Iris	95.4	3.0
Hepatic cancer	90.5	2.4

bination - two-parent and one offspring - and mutation with normal perturbation. The offspring replaces the parent only if fitter.

Results: Table 6.2 outlines the average results obtained by the cooperative approach with the chosen parameter values. For all three data sets, the technique proved to be viable and stable.

Observations: In order to verify how many generations are necessary for the technique to reach a constant and good result, we apply the evolved rules to the test set from an early stage of the evolution process. This test is performed solely on the breast cancer data set. In the initial (approximately 10) generations, the results are very unstable, jumping from 20% to 80% and in-between, reaching then a certain stability of about 80% accuracy and growing slowly, but constantly. In the final generations, there are only minor modifications of the test accuracy of up to one percent. However, the results greatly depend of the way the training/test sets are generated as there exist cases when the accuracy starts with 80% even from the early stages of the evolutionary process.

It has to be noted that there is not a very strong dependence between the EA parameters and obtained results as very competitive accuracies are obtained for a large scale of their values.

Concerning the coevolution parameters, changes within the collaboration credit assignments do not bring vital modifications to the final average results: The differences between various settings as regards the final accuracies do not overcome one percent.

As previously stated, the collaboration pool size parameter directly influences the runtime of the program that implements the approach; the final test accuracy is also affected by the increase

in this value, but a balance has to be established between runtime and accuracy. For the two considered test problems in the pre-experimental stage, better results are obtained for an odd number of collaborators. Another important observation is that when a certain threshold for the collaboration pool size parameter is surpassed no further gain in accuracy is reached. We generally achieved the best results when three collaborators were considered.

The cooperative parameter that brought the most important change in the final result of the algorithm is the collaborator selection pressure. A selection scheme is preferred to a random selection; selecting only the best collaborator seems to be the worst of the three choices.

Discussion: The proposed classification approach based on cooperative coevolution provides very accurate results in a relatively small amount of time. For instance, on the breast cancer data set, the runtime lasts from 7 seconds per run when the collaboration pool size is one, up to 24 for three collaborators and to around 37 seconds when five collaborators are used. Note that for experiments, we used a computer with the following characteristics: Pentium IV CPU 3.0 GHz and 1 GB of RAM.

6.5.2 Experiment 2: Competitive Classification Validation

Pre-experimental planning: The same two benchmark data sets from the UCI repository are used for preliminary experiments. The first observation in these tests refers to the high amount of time necessary for the algorithm to run: The explanation lies in the fact that the tests population is very large and, at the beginning of the evolutionary process, all samples are evaluated. This means that for each sample, for a number of times equal to the history length, an individual is selected and an encounter takes place between the two, assigning a score to the sample.

We also noticed at this stage the importance of the two competitive coevolution parameters: History length and the number of encounters. They also significantly influence the runtime of the program that implements the algorithm. However, a more exact evaluation of an individual or sample is obtained if the history length value is large and, on the other hand, a high value for the number of encounters updates the evaluations of individuals/samples.

Task: It will be investigated if the competitive classification technique can perform as well as the cooperative approach.

Setup: The values for the parameters are manually tuned, like in the cooperative case. Found values are indicated in Table 6.3.

In the current experiment, in order to enhance the speed of the algorithm, we tried to reduce the population size as much as possible: Less individuals means they will have more encounters with samples from the other species and their fitness will be updated very often.

None of the considered data sets is normalized.

Results: The average results that are obtained after 30 runs by applying the competitive classification technique are illustrated in Table 6.4.

Observations: The average results show that the competitive approach is significantly weaker

Table 6.3: Parameter values for the competitive coevolution approach in application to real-world tasks

	Breast Cancer	Iris	Hepatic Cancer
Evolutionary Parameters			
Population size	100	50	100
Mutation probability	0.5	0.5	0.4
Mutation strength	8	1	10
Generations	100	300	150
Competitive Coevolution Parameters			
History length	30	30	30
Number of encounters	20	30	20

Table 6.4: Average results after 30 runs for the competitive coevolution approach in application to real-world tasks

Data set	Average accuracy (%)	Standard deviation (%)
Breast cancer	92.9	2.9
Iris	91.1	3.5
Hepatic cancer	84.7	3.4

than the cooperative one. Not only the final results prove that this approach is much poorer than the cooperative one, but there is also a great difference as concerns runtime: To make an objective comparison, we measured the average runtime for the same breast cancer data set, by using the competitive approach with the parameters indicated in Table 6.3; the obtained value is of around 480 seconds, that is almost 13 times slower than the cooperative approach with 5 collaborators.

The standard deviation of the results is also significantly higher, which indicates the fact that this technique is not as stable as the cooperative approach. Nevertheless, it has to be stated that in an objective judgement, there are not too many perturbations that appear in 100, or even 300 generations, since only two individuals per class recombine during one generation and mutation is applied solely to the obtained offspring. To conclude, at least 1000 generations would probably be necessary for the variation operators to considerably change the population. That would, on the other hand, slow down the program even more.

Discussion: The obtained results and the large runtime of the competitive technique indicate the cooperative approach as much more viable as compared to the state-of-the-art techniques for classification. Note however that this is only the first time the competitive approach for classification is proposed and we believe that it represents a good starting point, as there is definitely potential within this technique as well. To outline some ideas for future research concerning the competitive technique for classification, perhaps a preprocessing technique step could be first applied to the training data in order to substantially reduce them. In conjunction with that (or by itself), we presume that employing a chunking technique in order to pick only small parts from the training set and use them as the static species could significantly improve runtime (maybe even the accuracy). Then, after the rules are specialized on the selected samples, the tests species

Table 6.5: Comparison to accuracies of data mining techniques reviewed in [Bennett, 1997], [Duch et al., 1999] and [Iacus and Porro, 2006]

Task	Worst reported accuracy	Best reported accuracy
Breast cancer	94.2%	97.2%
Iris	93.47%	96.31%

could bring new ones, while the dynamic population of rules could resume the evolution.

An important enhancement could be brought if the very good rules that are evolved at a certain point could be blocked for further modifications: Make one such individual a *tabu rule* and maybe move it in a rules archive that will be applied when the termination condition is reached. In the way the technique is now built, these good rules have the highest chances to be selected over and over again, therefore modified many times, maybe for the worse.

In the end of the evolutionary run, the best rule of each class in the final population is taken and the entire formed set is applied to the test data. Obviously, a different way of choosing the rules could be imagined, e.g. take several rules for one class or apply an archive variant as suggested above.

6.5.3 Comparison to Standard Data Mining Approaches

Comparison of obtained results can be made only for the breast cancer and iris data sets, since they are public benchmark problems. The resulting rules of the hepatic data set are however confronted with the specialized opinion of the physician and are found to be consistent with the medical diagnosis.

A summary of best and worst accuracies in literature concerning the two considered UCI data sets is presented in Table 6.5. These results come from surveys on several canonical data mining techniques in [Bennett, 1997], [Duch et al., 1999] and [Iacus and Porro, 2006]. Comparison cannot be objective, however, as outlined methods either use different sizes for the training/test sets or other types of cross-validation and number of runs or employ various preprocessing procedures for feature or sample selection.

The highest and lowest obtained accuracy for the breast cancer diagnosis problem are reported in [Bennett, 1997], [Duch et al., 1999]. However, the authors used 10-fold cross-validation and removed the samples that had missing values.

On the other hand, for the two results for iris [Iacus and Porro, 2006], a similar way of selecting the training and test sets is used. The difference to our approach is that 80% of the samples from the Iris data are used for training and the rest (less samples than in our approach) for testing and that average accuracies are obtained after 500 runs.

6.6 Summary

The basic ideas of this chapter can be summarized as follows:

1. The idea of coevolution as an inspiration source for the evolutionary community is presented.
2. Cooperative coevolution, together with all its parameters is then outlined.
3. The basics of the competitive counterpart is afterwards presented.
4. Cooperative coevolution is then proposed for building rules for classification purposes.
5. Competitive coevolution is then used for classification.
6. Results of the two techniques are outlined and compared. Cooperative coevolution proved to be more accurate and faster than the competitive approach.

6.7 Future Work

The two types of coevolution within which species either cooperate or compete are herein proposed as tools for solving classification tasks. The cooperative approach, probably even due to the fact that it has been more extensively tested, proved to be more much efficient than the competitive one, as concerns both the accuracy and runtime.

The presented coevolution framework brings a natural manner of targeting classification, with a simple and concise representation and a straightforward fitness assignment as an advantage over existing evolutionary possibilities.

An important drawback of the cooperative technique is however the fact that the number of populations must increase with the number of classes of the problem. Another point for cooperative coevolution approach that is envisaged for the future is to develop a methodology of forming sets of rules that can contain several different rules for the same class.

The competitive classification technique brings an interesting and dynamic perspective of targeting classification. However, the method in its current state did not prove to be as efficient as the cooperative approach. Conversely, an important advantage of the competitive scenario over the cooperative alternative is that the raise in the number of classes does not target the automatic increase in the number of populations and thus complicate the evolutionary system. Further study and enhancement will complete the creation of a general coevolutionary framework, containing both cooperation and competitiveness, that can provide new insights and successes into the demanding and crucial field of classification.

Chapter 7

Topological Species Conservation Hybridized Technique

7.1 Objectives of this Chapter

In this chapter, a hybridization between two recent multi-modal techniques is investigated, resulting in the novel topological species conservation (TSC) [Stoean et al., 2008b]. It is hoped to combine the strengths of both techniques, the species conservation genetic algorithm (SCGA) and multinational algorithms, while avoiding their weaknesses. On the one hand, the SCGA is simpler and more efficient than the multinational paradigm; on the other hand, the latter comes with a more natural decomposition of the population into species. By combining species conservation with an alternative mechanism for subpopulation determination, two goals are envisaged: Preservation of the best local individuals during runtime and the achievement of a natural differentiation between the multiple final solutions of the multi-modal problem.

7.2 Advantages/Disadvantages of the Parent Techniques

Although both parent techniques are presented in Chapter 3, their basic ideas are briefly reconsidered here. Within the SCGA in [Li et al., 2002], the fittest individuals that are different from each other within a certain radius are set as seeds of their subpopulations; all the other individuals (that are not seeds) belong each to the subpopulation of the fittest individual that is found close to them within the given radius. The seeds are conserved from one generation to another in order to avoid the risk that some of them may disappear due to the application of genetic operators. Naturally, the seeds are updated at each generation. The elitist SCGA idea of transferring the seeds of each subpopulation from generation to generation is also adopted in the technique herein proposed. The main disadvantage of SCGA is the use of the species radius parameter which has a value that is problem dependent and therefore it is hard to be set.

An original approach that does not make use of radii and, moreover, of distances between genotypes when separating individuals into subpopulations was developed by Ursem in [Ursem, 1999, Ursem, 2000]. It detects whether two individuals track the same optimum by considering a set of

additional candidate solutions between them and testing if one of these is weaker than both. If this is the case, a valley between the individuals is assumed and consequently, they are presumed to track different optima and shall be distributed to different subpopulations. The method is described in chapter 3.

Besides the great advantage that comes with this way of determining if two individuals follow the same peak or not, that of not using a radius anymore and, moreover, being even more precise at distinguishing between different hills, there comes a disadvantage, that of *consuming* fitness evaluations. The complex model of multinational evolutionary algorithms makes use of the method for detecting multi-modality for each individual in every generation and, moreover, considers it for nations, as well. Overall, multinational algorithms proved to be a powerful tool in finding the optima for several test functions [Ursem, 1999]. However, the main drawback is that the method uses a large amount of fitness evaluations [Stoean et al., 2007a], fact that makes it unsuitable for real-world applications.

7.3 Description of the Proposed Hybridized Technique

The novel proposed algorithm, Topological Species Conservation (TSC) [Stoean et al., 2008b] inherits the ideas of having a dominating individual (seed) for each species and that of conserving the seeds. At the same time, within TSC, subpopulations determination is done through the same procedure as in multinational algorithms. The first attempt of achieving such a hybridization is made in [Stoean et al., 2007a]. Additionally, some further important improvements are introduced, especially as concerns the way individuals are distributed into subpopulations, and a deeper investigation upon the insights of the technique is performed.

7.3.1 Motivation

The efficiency of the SCGA method lies in the elitism it uses, as subpopulations cannot be completely lost, even if selection may skip all individuals within one population or they may disappear because of recombination and mutation. Thus, conservation of the seeds of the found subpopulations prevents them from getting extinct. However, within SCGA, no particular mating selection mechanism is used which has the disadvantage that, after a small number of generations, most of the individuals will be situated in the subpopulations that are connected to the fittest regions in the search space. The local optima are very likely to be followed only by subpopulations containing the seed that is only conserved from one generation to another. Therefore, fine tuning is not performed for the local optima. In order to avoid this situation, a shared fitness for the mating selection is employed and, as a consequence, each optimum has a subpopulation size proportional to its fitness.

Then, replacing the radius-dependent method to differentiate subpopulations in favour of one that employs fitness discrepancies as in multinational algorithms may have two advantages:

- One gets rid of a crucial parameter for which it is very difficult to find a proper value, especially in higher dimensional problems;
- Less fit individuals that are actually not promising, but merely different enough from the others, are rapidly detected. This is obvious especially for large plateaus contained in the fitness landscape or for optima that have very large basins of attraction: While the SCGA method would form a great number of subpopulations, multinational algorithms detect only one peak to follow.

We managed to avoid the expensive behavior of the original multinational algorithms that, due to its subpopulations dynamics, which is achieved through migration and merging, uses a high number of fitness evaluations in a very small number of generations. By incorporating the preservation of multi-modality through seed conservation and efficiently keeping track of each individual subpopulation during evolution, we overcame this great disadvantage.

The number of seeds is restricted to a percentage of the population, fact that did not appear in the SCGA. This is very important for the highly multi-modal functions, where a high number of seeds is formed even from the early stages of the EA. The SCGA does not have such a limit for the number of subpopulations that can be formed and, therefore, the entire population can be blocked into local optima.

In conclusion, the proposed approach borrows strength, while simultaneously solves inefficiencies from these both canonical techniques. The SCGA has the weakness of the use of a radius, while the multinational algorithms have an interesting and working underlying idea, but nevertheless very expensive, if one counts the fitness evaluation calls.

7.3.2 The Mechanics

Within the proposed technique, the main characteristics of a species are the following:

- In the ideal case, all individuals within the same species lie in the basin of attraction of the same optimum. This certitude very much depends on the number of intermediary individuals that are considered for the verification of the multi-modality.
- An individual can belong to only one species.
- Each species has a seed, which is represented by the fittest individual of that species.
- For each species, i.e. for all individuals it contains, a unique positive integer value is assigned as ID. The purpose of the ID is to avoid the repetition of the multi-modal verification over the generations.

The method does not employ a radius for separating subpopulations, however, at certain times it makes use of the dissimilarity between individuals with the purpose of reducing the number of consumed fitness evaluations. In this respect, in conducted experiments, the Euclidean distance

is used: Having two individuals $x = (x_1, x_2, ..., x_m)$ and $y = (y_1, y_2, ..., y_m)$, the distance is defined by (7.1). However, any other distance measure can be successfully used for computing the dissimilarity between different individuals.

$$d(x, y) = \sum_{i=1}^{m} \sqrt{(x_i - y_i)^2} \tag{7.1}$$

In order to outline the way subpopulations are formed, we first need to see how the detect-multi-modal mechanism functions.

The Detect-multi-modal Method

The verification of whether two points in the search space track the same optimum is performed through an approach that was originally referred as the *hill-valley* mechanism, but which, for reasons of simplicity, is renamed to *detect-multi-modal*. The function takes two individuals (points) as arguments and returns whether or not there is a valley between them in the fitness landscape, i.e. they track different optima.

In order to reach a decision, a set of interior points between the two given as arguments is generated. The interior points are chosen based on user-defined gradations in the [0,1] interval. If the fitness of all interior points is higher than the minimal fitness of the two tested individuals, then it is concluded that they track the same optimum. Contrarily, if there exist such a point whose fitness is smaller than the minimal fitness of the two, then it is assessed that they follow different peaks.

In conclusion, *detect-multi-modal* returns true if the two points follow different peaks and false if they follow the same optimum. The mechanism is described in chapter 3, Algorithm 9, where it is referred to by the original *hill-valley* name.

The value for the *number of gradations* in Algorithm 9 actually coincides with the number of interior points that are considered. The vector $gradation_j$ contains equally distant values in the interval (0, 1). If an individual that has the value for the fitness evaluation smaller than the minimum evaluation value of the two initial points is found, the method stops and returns *true*. As a consequence, all the interior points are evaluated only in case the two individuals follow the same peak or if only when the final point has the evaluation smaller than the minimum evaluation of the two.

An important advantage of this manner of detecting multi-modality is that there are not several subpopulations assigned to follow a certain optimum, as it happens when the radii-based mechanism of species conservation is used, but only one subpopulation, regardless of the size of the basin of attraction of that optimum.

Next, the use of conservation in the proposed technique is justified and then the steps that are followed within each iteration are presented.

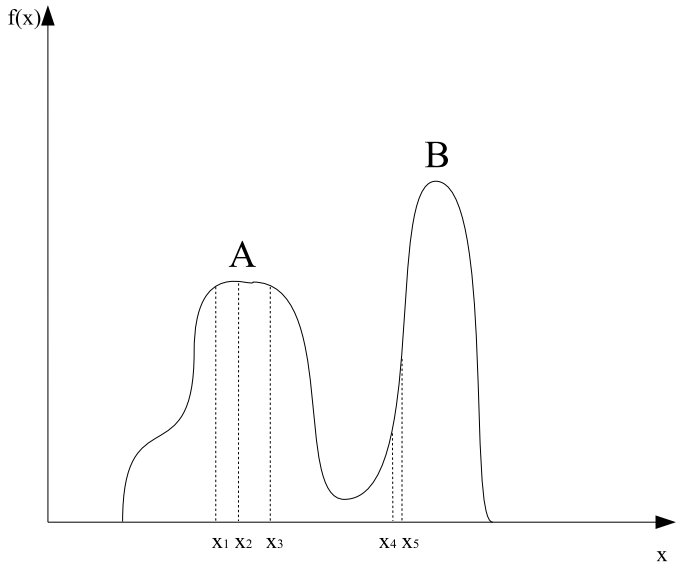

Figure 7.1: Valuable individuals could vanish if not conserved.

Conservation. Is it Necessary?

In each generation, we have a certain number of species, each having its dominating individual and each of them following a different peak. On the one hand, we employ a weighted mating selection, meaning that the fitness of each individual is divided by the size of the species it belongs to. This gives a greater chance to species that have only few individuals to escape extinction, just like in Goldberg and Richardson fitness sharing [Goldberg and Richardson, 1987].

On the other hand, this precaution measure is not always sufficient, as there may exist subpopulations with few individuals that are situated just on the base of an optimum, as it is the case with points x_4 and x_5 in Figure 7.1; they may not be selected for recombination at all, or, if selected, might recombine with individuals from different species and produce fitter offspring in other regions of the search space, which would eventually replace them. Therefore, for each of the subpopulations detected so far, the best individual each contains is retained in the next generation. However, this copying takes place only in the situation when the individual does not already exist in the population, so that several clones of the same individual are not maintained.

Regarding conservation, the fact that we avoid having multiple instances for these individuals is the only difference of the proposed technique in comparison to the SCGA. However, the preservation of the species and especially the fact that the niches are kept occupied by a number of individuals proportional to their resources is also achieved within TSC, by means of weighted selection, mechanism which is not integrated within SCGA.

Determining the Species

Before referring to species determination, we have to indicate the way the seeds are found, as species form by gathering individuals around these seeds. The first generation is the most

expensive one as regards the used number of fitness evaluations because this is the time the $detect - multi - modal$ method is applied for forming the starting subpopulations. In the next generations, till the end of the evolutionary process, the species IDs are used wherever possible. Algorithm 17 [Stoean et al., 2008b] describes how the seeds are selected and, at the same time, the subpopulations are formed around them. We denoted by n the population size and by P_i the i-th individual in the current population P.

Algorithm 17 Seeds selection procedure

Require: The current population P
Ensure: The seeds
begin
Sort population P decreasingly according to the fitness;
$Seeds = P_1$; (fittest individual is a seed)
if not(first generation) **then**
$P_{1_{previousID}} = P_{1_{ID}}$;
end if
$P_{1_{ID}} = 1$; (the ID of the first seed)
$currentID = 2$; ($currentID$ incremented)
for $i = 2$ to n **do**
if first generation **then**
Find the closest seed s in $Seeds$ for which detect-multi-modal(P_i, s) = $false$;
else
Find the closest seed s in $Seeds$ for which $P_{i_{ID}} = s_{previousID}$
end if
if exists such a seed s **then**
$P_{i_{ID}} = s_{ID}$; (P_i belong to species dominated by s)
else
$Seeds = Seeds \cup P_i$; (P_i is a seed)
if not(first generation) **then**
$P_{i_{previousID}} = P_{1_{ID}}$;
end if
$P_{i_{ID}} = currentID$;
$currentID = currentID + 1$;
end if
end for
return the $Seeds$ set
end

The set $Seeds$ is constructed by considering all individuals, in decreasing order of their fitness. The fittest individual represents the first seed that is added to the set. In the first generation, when an individual is considered, it is checked against the other existing seeds, using the $detect-multi-modal$ mechanism, to see whether it follows the same peak or not. In order to save some fitness evaluations for being spent, we try to avoid unnecessary applications of the $detect-multi-modal$ method and choose the seeds by starting from the one closest to the current individual. The species

dominated by this seed is, naturally, the most likely one to follow the same peak as the current individual. If this is not the case, the individual is checked against the next closest seed and so on.

The seeds for all species are updated in every generation. As the entire population is ordered decreasingly at each iteration, the IDs of the subpopulations do not remain identical from one generation to another, hence the need to retain the previous IDs for the newly set seeds, so that the individuals that belong to their species could be identified and have their IDs updated. Thus, after the first generation, when an individual is verified whether it is a seed or if it belongs to a certain species, it is not the $detect - multi - modal$ that is applied to verify if it follows the same peak with any of the already found seeds, but its seed ID is compared to the IDs that the currently detected seeds had attributed in the previous generation for this purpose.

Obviously, not all species are detected from the first generation and kept until the end of the evolutionary process, but new species can be detected and added to the existing ones in each iteration. The evolutionary process continues with the weighted mating selection and then the variation operators are applied. When mutation is applied to an individual, the offspring does not belong to any of the existing species, i.e. it does not have a value for the ID. These candidate solutions are further referred as *free individuals*. In case of recombination, if both parents belong to the same species, the offspring inherits the ID from the parents. Otherwise, the descendants will be free individuals, just like in the case of the mutation offspring. The conservation of the species seeds follows afterwards and subsequently the newly created individuals with no assigned ID are integrated.

Seeds Conservation

The conservation of the seeds is described in Algorithm 18 [Stoean et al., 2008b]. f denotes the fitness function. For each seed, be that it does not already have an instance in the population, the worst individual of its species is searched, i.e. the least fit individual that has the same ID value as the seed. If the seed has a higher value for the fitness evaluation than that individual, then it enters the population instead of it. In case there is no individual in the population belonging to the same species, the seed is introduced instead of the worst, unmarked individual in the population. The marking is necessary in order to avoid deleting already introduced seeds.

Two differences exist as compared to the conservation procedure in the SCGA:

- Before inserting the seeds in the population, we are checking whether a copy of their instance does not already exist, in order to avoid having duplicate individuals in the population.
- No distance is used, as we are verifying whether the species IDs coincide to the ones of the individuals that are to be replaced by the seed.

Algorithm 18 Seeds conservation procedure within TSC

Require: The current population P

Ensure: The population that contains the seeds

```
begin
Mark all individuals in P as unprocessed;
for every s in Seeds do
  if s does not already exist in P then
    Take worst unprocessed w from P, such that s_ID = w_ID;
    if w exists then
      if f(w) < f(s) then
        w = s;
      end if
    else
      Take worst unprocessed w in P;
      w = s;
    end if
    Mark w as processed;
  end if
end for
return the population with the integrated seeds
end
```

Free Individuals Integration

The mechanism of integrating the free individuals is described in Algorithm 19 [Stoean et al., 2008b]. In order to avoid the formation of too many species, which may happen only in case the optimization function is highly multi-modal, a maximum percentage of the population size that can be appointed as seeds is considered: MAX_{Seeds} denotes the actual maximum value of seeds that can be appointed. In all the experiments, $MAX_{Seeds} = 20$, as it is considered that an average of four individuals per species would be convenient.

The first choice for the integration of the individuals outside a species is to test whether they belong to any of the already existing ones. Thus, the solution is to check for each individual whether it follows the same peak as any of the existing seeds through applying the $detect - multi - modal$ method. With the aim to avoid the excessively use of the $detect - multi - modal$ method, for each free individual we check the seeds by starting with the most likely one to follow the same optimum, that is by considering step by step, the closest one to the current individual. If a seed that follows the same peak as the current individual is found, then the individual is set to belong to that seed species, it takes the ID from the seed and the individual is no longer $free$.

If there remain individuals that do not belong to any of the existing species, then the choice for these individuals is to build their own species in which they represent the seeds. That is done by ordering all these individuals in decreasing order of fitness and then establishing the fittest one as a new seed with the ID incremented from the last species ID. Next individual is then checked whether it belongs to the same newly created species. If so, it will have the same ID assigned, otherwise, it will be a new seed as well, having the next ID value. The process continues for all individuals by checking them only against the newly added seeds.

If there still remain free individuals, fact that can happen only in case the maximum number of seeds has been reached, we just assign these individuals to the seeds closest to them. Thus, in case MAX_{Seeds} species are formed at a certain point and a better solution than the existing ones is found, it will enter in the closest seed subpopulation that exist in the search space and, in the next generation, this solution will be chosen as seed of the species. This way, it is conserved from one generation to another and there does not exist the risk of extinction.

Topological Species Conservation Steps

After describing the main steps that have to be followed by TSC, we integrate them altogether in Algorithm 20 [Stoean et al., 2008b]. At each generation, before the mating selection is applied, the species are identified and the IDs of all individuals are updated. A weighted mating selection is chosen in aiming to keep a good proportion between each niche resources and the individuals it contains. Individuals from different species are allowed to recombine as their descendants may appear in unexplored regions of the search space and, in case an optimum lies there, they settle new species. On the other hand, when recombination takes place between individuals from the same species, as intermediate recombination is chosen, the offspring is considered as belonging to

Algorithm 19 Integration of the *free individuals*

Require: A set of free individuals

Ensure: The population and $Seeds$ set with the integrated free individuals

begin
for each free individual x **do**
Find the closest seed s to x for which detect-multi-modal(x, s) = $false$;
if s exists **then**
$x_{ID} = s_{ID}$;
end if
end for
if $Seeds$.length $< MAX_{Seeds}$ **then**
$currentID = Seeds$.length + 1;
Find the fittest free individual x;
$Seeds = Seeds \cup x$; (x is a new seed)
while there are still free individuals and $currentID < MAX_{Seeds}$ **do**
For the fittest free individual x find the closest newly added seed s to x for which $detect - multi - modal(x, s) = false$;
if s exists **then**
$x_{ID} = s_{ID}$;
else
$currentID = currentID + 1$;
$Seeds = Seeds \cup x$;
$x_{ID} = currentID$;
end if
end while
else
for each free individual x **do**
Find the closest seed s to x;
$x_{ID} = s_{ID}$; (integrate the free individual to closest seed species)
end for
end if
return the population and $Seeds$ set with the integrated free individuals
end

the same subpopulation as their parents, i.e. it inherits the ID from them. The seeds that were retained in the set *Seeds* before the variation operators were applied are then integrated in the population and then the assimilation of the descendants that do not yet belong to any species takes place.

Algorithm 20 Topological Species Conservation Structure

Require: A search/optimization problem
Ensure: The set of seeds
begin
Initialize population;
while stop condition is not met **do**
Identify species seeds;
Apply mating selection;
Apply recombination;
Apply mutation;
Integrate the seeds in current population;
Integrate free individuals;
end while
return the set of seeds
end

In addition to the version in [Stoean et al., 2007a], the technique is further enhanced as we avoid introducing duplicate individuals when seeds conservation takes place, make use of distances in order to escape the frequent application of $detect - multi - modal$ and thus save an important amount of fitness evaluations, the free individuals are separately treated (and not during the seed integration process) and a maximum limit of seeds is set.

As compared to SCGA, besides the facts that we do not make use of a radius whose value is hard to be set sometimes and that we do not have to compute so many distances in order to identify the species together with their seeds, within TSC there is no need for a mechanism of identifying the final solutions: all the seeds provided in the end by TSC represent the solutions, in case the aim is to find several global and local optima. This is due to the fact that within TSC, all individuals that follow a certain optimum lie in the basin of attraction of that optimum, and there does not exist the case that different species follow the same optimum. Plus, the parameter that gives the number of interior points to be considered for TSC, the number of gradations, which is a positive integer, is naturally easier to be tuned than the positive real-valued radius within SCGA. However, in the experiments section, direct comparison of how dependent the models are on the two parameters is conducted.

7.4 Application to Function Optimization

In the following experiments, we try to fill all cells of a table made of the different modality conditions, one, few, and many optima, and of the different search space sizes, low and high dimensional. In order to have relatively difficult problems when dealing with functions with low number of dimensions, we consider as easiest case the optimization of functions with two variables.

The optimization functions considered are $F1$ (Waves), which is also considered in [Ursem, 1999] and $F2$ (Six-hump Camel Back) that is used in both [Li et al., 2002] and [Ursem, 1999] and we add $F3$ (De Jong), $F4$ (Shifted Rastrigin), $F5$ (Rotated hybrid composition function) and a shifted version of $F2$ which is referred here as $F6$.

Recent simulation model based investigations [Preuss, 2006] lead to the conjecture that complex multi-modal optimization algorithms may perform better than simple multistart methods only if the number of optima is relatively low. F3 is tested to show that even in case of only one optimum, the considered methods still perform sufficiently well.

Having equally distant optima would advantage a radii-based EA as a proper value for the radius would aid in detecting all peaks. Plus, as a real-world problem does not usually have a regular fitness landscape, we rescaled the original Six-hump Camel Back function in order to have the optima, two by two more distant from each other ($F6$). Waves function already is a function that is asymmetric and has many peaks, some of which being even more difficult to find as they lie on the border or on flat hills.

More information regarding all considered test functions can be found in Appendix A.

7.4.1 Direct Performance Comparison

Pre-experimental planning: In a previous version of the method a maximum number of seeds was not set. This parameter proved to be vital when we dealt with the $F5$ function: the results were very poor, even when the test function was considered for two variables. The number of seeds was increasing very fast as generations were passing. Having a population of 200 individuals, about 180 seeds were chosen in less than 30 generations, meaning that 90% of the population is blocked from the initial steps of the algorithm. However, after setting this MAX_{Seeds} value, this situation was tackled.

Another circumstance we got aware of during the initial experiments regards the number of duplicates that are introduced through species conservation: By verifying the existence of the seeds instances in the current population before introducing other copies, especially for the highly multi-modal fitness landscapes where the number of found seeds is very high, the addition of many duplicates which would have blocked other potential solutions is avoided.

Task: Perform a direct and objective comparison between SCGA and the TSC technique. The multinational algorithm is not considered for comparison because it proved to be less efficient than both SCGA and a previous form of TSC in [Stoean et al., 2007a].

Experimental Setup: In order to have an objective comparison, the only user interaction in

setting the parameters for the two techniques appears in defining the ranges for the parameter values. A Latin Hypersquare Design (LHD) is performed for all test functions in setting the exact values for all the parameters of the two techniques.

For all considered test functions and for both techniques, the same budget of $3 \cdot 10^4$ fitness evaluations was set. The upper bound of the population size is restricted for both methods to 200. The mutation and crossover probabilities are selected by the LHD in the interval [0, 1] in all experiments. The upper bound value for the mutation strength parameter is identically set for both techniques, but differently for each benchmark function: For $F2$, $F3$ and $F6$ the interval is set between 0 and 5, while for the other cases the interval is [0, 20].

Regarding the intervals for the two specific parameters of the techniques, for the number of gradations in TSC we considered for all benchmark problems values in the set {1, 2, ..., 15}. For the SCGA radius, the interval sizes are set depending on the problem to be solved: The actual radius value was computed for each function using the Deb and Goldberg formula that is outlined in chapter 3, Equation (3.3) and then, for each test function, the interval was set as approximately double that value, in order to make sure a proper configuration is included into the ones generated by the LHD. For $F1$ and $F5$ with 2 variables, the upper bound is set to 25, for $F2$, $F3$ and $F6$ it is set to 5 and for $F5$ with 10 variables, the value is set to 40.

There are 30 LHD points considered for each test function and for each of the considered techniques and every configuration is replicated 30 times: The mean, best and worst values of the detected peaks are recorded.

The semi-automated tuning method of sequential parameter optimization (SPO), see [Bartz-Beielstein et al., 2004b], is utilised for the optimization of $F5$ function with 10 variables as there was space for improvements in the results given by LHD. The method starts with 100 LHD points, each repeated 4 times. The total budget of runs was set to 1000 and the model reduces the noise by incrementally increasing the repetition of runs up to the maximum value of 30.

A technique is considered to be successful for a test function when there exists at least one individual in the population of the last generation that is situated in the basin of attraction of that optimum with an accuracy closer than 10^{-1} to it. If the purpose for a certain function is to reach one optimum (out of many), then the average fitness evaluation out of the 30 repeats is outlined for direct comparison, while when several optima are to be found, the average number of detected optima is used as the comparison criterion.

Results and Visualization: Table 7.1 outlines the results obtained from the considered LHDs. All functions are considered for maximization, meaning that higher values correspond to better results. The average results from the best and worst configurations, as well as the average over all design points are shown.

Results for the Waves function ($F1$) prove that TSC performs significantly better than SCGA, not only in average, where TSC detected 7.77 peaks and SCGA only 1.09, but also the average result of the best TSC configuration is much better than the corresponding one from the SCGA (9.9 and 2.8 respectively). The average results from all LHDs for $F2$, as well as for $F6$ indicate

Table 7.1: Average, best and worst results obtained in 30 LHD points, each replicated 30 times. While for $F1$, $F2$ and $F6$ the average number of detected peaks are reported, for $F3$, $F4$ and $F5$ the best average fitness value is presented.

Test	TSC			SCGA		
function	Best	Mean	Worst	Best	Mean	Worst
$F1$	9.9	7.77	1	2.8	1.09	0.96
$F2$	5.96	4.63	1	5.8	1.91	1
$F3$, 2 var	2.8e+12	9.5e+09	2.3e+07	1.7e+10	6.4e+08	2.9e+04
$F3$, 10 var	3.4e+07	9.6e+02	1.31	5.1e+07	6.26	36.63
$F4$, 2 var	329.99	329.51	327.18	329.99	329.98	329.7
$F4$, 10 var	329.99	299.11	232.41	329.99	311.91	212.92
$F5$, 2 var	-360.2	-480.09	-834.47	-390.88	-531.76	-709.9
$F5$, 10 var	-991.67	-1268.11	-1659.12	-1036.95	-1289.49	-1867.09
$F6$	5.93	4.62	1	4.06	1.35	1

the fact that TSC performs better than SCGA.

When the aim is to find the global optimum, the two techniques are very competitive, with minor differences for the highly multi-modal function $F5$ in favour of TSC. None of the two methods found the optimum in any run for $F5$ with 10 variables.

The percent of runs out of the 30 repeats when the goal has been achieved is also measured (Table 7.2). When a zero value is set in the table, it does not necessarily mean that the method completely failed: For instance, if the goal is to find 6 optima, if only 5 of them are reached, the run is considered as failed, but the average number of optima is recorded in Table 7.1. The same idea in strengthened, i.e. when the task is to find several optima, TSC results outperform the ones given by SCGA. The only test case when SCGA proved to be more accurate than TSC is the $F4$ function.

As regards the results obtained by SPO for $F5$ with 10 variables, the global optimum still could not be obtained, but the average results obtained by the LHD is improved for both techniques: The best average value for TSC is -857.44, while the corresponding one for SCGA is -988.06. The parameters found for TSC are the following:

- Population size: 161
- Crossover probability: 0.16
- Mutation probability: 0.38
- Mutation strength: 9.57
- Number of gradations: 2

while the detected ones for SCGA are:

Table 7.2: The percent of runs in which the goal has been achieved. Outlined results are obtained from the same configurations given by LHD, each repeated 30 times.

Test	TSC			SCGA		
function	Best	Mean	Worst	Best	Mean	Worst
$F1$	93.33	22.78	0	0	0	0
$F2$	96.66	33.33	0	80	8.33	0
$F3$, 2 var	100	100	100	100	99.88	96.66
$F3$, 10 var	100	86.44	0	100	70.88	0
$F4$, 2 var	100	78.22	0	100	98.88	66.67
$F4$, 10 var	100	4.33	0	100	33.67	0
$F5$, 2 var	76.67	20.22	0	76.67	21.22	0
$F5$, 10 var	0	0	0	0	0	0
$F6$	96.67	32.22	0	6.67	0.22	0

- Population size: 167
- Crossover probability: 0.08
- Mutation probability: 0.51
- Mutation strength: 11.74
- Radius: 23.68

Discussion: The main advantage of the TSC over SCGA is that the former does not make use of a radius for separating the individuals in the population into subpopulations, but only uses the fitness evaluations of some interior solutions to realise what is the geometrical form of the landscape and, therefore, create subpopulations which include individuals that track the same peak. On the other hand, TSC has the disadvantage that it uses a higher number of fitness evaluations just for separating the subpopulations, while the SCGA does not spend any of the fitness evaluations in its budget for this purpose. Note, however, that in the undertaken experiments, the same budget of $3 \cdot 10^4$ fitness evaluations was set and the results clearly indicate that TSC performs better than the SCGA.

TSC does not make use of a radius, but it has an integer parameter, which is the number of gradations. It is obvious that the number of gradations parameter is easier to be set than the radius in the SCGA case, as the former is a positive integer value, while the latter is a positive real number. However, further on, this dependence of the model on its specific parameter is investigated.

7.4.2 Model Dependence on Radius/Number of Gradation Parameters

Pre-experimental planning: The formation of subpopulations by only taking the fitness landscape into consideration and not relating to a radius can be a good advantage. In order to ensure

that, two of the previous test functions are considered for both TSC and SCGA, e.g. $F2$ and its shifted version, $F6$. While TSC should behave in a similar manner for the two functions, as there is the same fitness landscape, it is expected that SCGA is sensitive to the fact that the optima in $F6$ are not equally distant.

On the other hand, as SCGA did not perform well for $F1$ function in the first experiment, finding a proper value for the radius and then investigating how dependent the method is on the values of this parameter is of great interest.

Task: Investigate how sensitive the two compared techniques are on the specific parameters: radius and number of gradations. On the one hand, the results given by the two techniques for $F2$ and $F6$ test functions are recorded and, on the other hand, different values are tried for the radius/number of gradations in case of $F1$ test function.

Experimental Setup: The same LHD points are considered for comparing the results for $F2$ and $F6$.

In order to find the best values for the radius/number of gradations, the best LHD configuration found in the previous experiment is used for all the parameters, except the two investigated. While for TSC, the number of gradations is tried for all possible values (1, up to 15), for the SCGA, the radius is exponentially scaled in the following manner: the starting value for the radius is computed using the Deb and Goldberg formula (3.3) in chapter 3 and then the value is multiplied with integer powers of 2 taken from the interval {-7, -6, ..., 6, 7}.

Results and Visualization: Regarding the first comparison of the results obtained by TSC and SCGA over $F2$ and $F6$ respectively, one can visualize the results in Tables 7.1 and 7.2. While TSC has approximately the same results for the two functions, SCGA shows a decrease in results when the optima are not equally distant, i.e. in the case of $F6$ function.

Results obtained for $F1$ by SCGA with different values for the radius parameter are shown in Figure 7.2. For each configuration, 30 repeats are performed. The actual value found using Deb and Goldberg formula is approximately 0.5. It can be clearly seen however from the figure that the best average result is not obtained for this value, but for smaller ones. None of the considered values for the radius was proper for detecting all 10 optima the test function has. Moreover, when the value for the radius is higher than 1, the number of detected optima is decreased to only approximately one optimum.

Figure 7.3 outlines the average number of optima detected when trying different values for the number of gradations parameter within TSC. As it can be seen, by increasing the value of the number of gradations from 2 up to 13, more than 9.6 optima are detected in average for 30 runs per configuration.

Discussion: Both assumptions within the current experiment proved to be verified: On the one hand, TSC is independent of the fact of whether the optima are equally distant or not, while SCGA is very sensitive to such changes of the peak location. This is an important drawback of the SCGA as there can be very seldom the case that a real-world problem has equally distant optima. On the other hand, it could be noticed that TSC does not depend very much on the

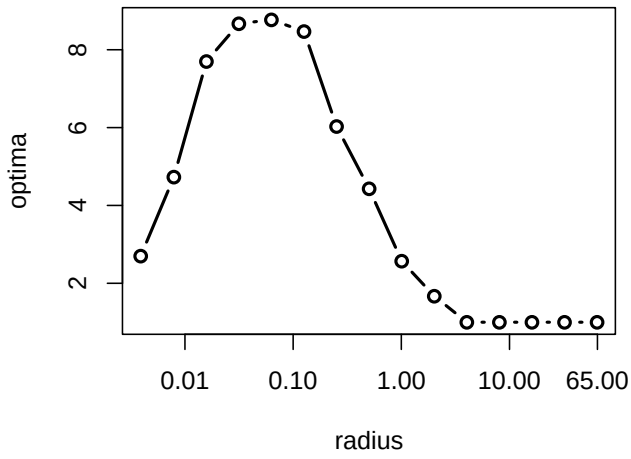

Figure 7.2: Average number of optima detected for different radius values within SCGA.

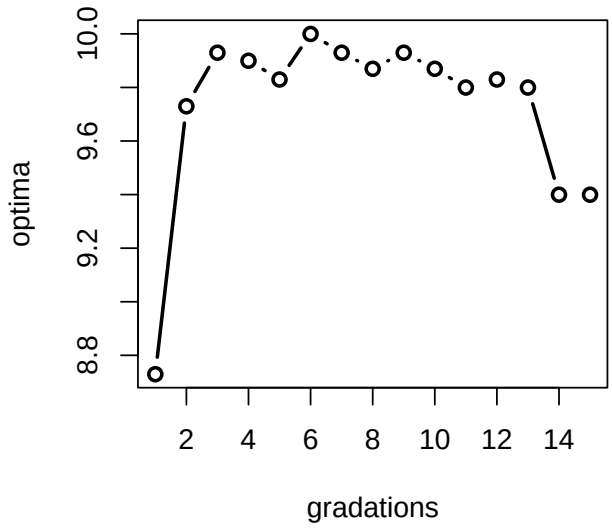

Figure 7.3: Average number of optima detected for different number of gradations within TSC.

value that is chosen for the number of gradations parameter, while the picking of the right value for the radius parameter within SCGA is vital in obtaining good results.

7.5 Summary

In current chapter a new hybridized evolutionary technique for multi-modal optimization, namely topological species conservation, is proposed. The method is described in detail, it is tested for the optimization of several benchmark functions and results are directly compared to the better one of the parent techniques. Results proved that the new technique both gets rid of a crucial parameter and performs significantly better than the parent approach.

7.6 Future Work

A first step to extend the current work would be to consider more irregularly formed and/or spaced test problems in order to see where the turning point between the TSC and the original SCGA is. It may also be useful to drive the hybridization even further and insert techniques based on relative (instead of absolute) distances into the TSC. Currently, only main parameter effects have been investigated. Once the parameter interactions are understood better, adding mutation step size adaptation mechanisms will surely open a path leading to substantially increased performance.

Chapter 8

Conclusions and Future Work

Current work is concentrated on author's research within the field of evolutionary computation. Two goals were envisaged: the development of new, powerful evolutionary techniques for multimodal optimization problems and the application of newly or existing evolutionary methods for real-world problems, precisely for classification.

8.1 Achievements

The thesis firstly presents an evolutionary computing background and then continues with the outline of the most commonly used and recently developed evolutionary techniques for multimodal optimization. It subsequently goes on by gradually introducing the author's research in the field since he engaged into PhD training.

- Genetic chromodynamics represents a source of inspiration for the development of two new, more efficient methods for multimodal optimization. One of them integrates within the original algorithm a procedure that resembles crowding, fact that changes the consideration of each individual in turn into a random selection of solutions.
- Additionally, some more fine tuning for the final solutions is envisaged in another variant of the technique that is built upon the already developed one: When solutions remain alone in each niche, mutated clones of them are inserted, only if they are fitter than the existing solutions.
- The first novel genetic chromodynamics variant is then used as an engine for a classifier that is applied to real-world data sets. The classifier is employed to create if-then rules that are trained using a part of the data set and are then tested on previously unseen data.
- Cooperative and competitive coevolution are considered herein as mechanisms for two different classifiers.
- A hybridization between two multi-modal techniques, the species conservation genetic algorithm and the multinational algorithms, is then investigated, resulting in the novel topological species conservation.

8.2 Remarks

A set of conclusions arise from the undertaken experiments:

- The goal of the GC with crowding technique was to maintain the ability of the original GC to find and preserve the global/local optima and additionally speed up the process. The experiments on function optimization proved that this especially happens especially for highly multi-modal problems and larger number of dimensions.
- Although accuracy is further increased in the case of the cloning GC variant, the method proves to be more expensive, especially when the number of considered clones is high.
- The coevolutionary classifiers are validated against several benchmark data sets and even on other raw collections. Their performance proved to be viable as compared to existing state-of-the-art techniques with the exception of the competitive approach that has not been largely tested in order to identify and remove its weaknesses.
- Topological species conservation proved to be independent of the fact of whether the optima in the fitness landscape are equally distant or not, while the species conservation genetic algorithm (the more efficient of the two parent techniques) is very sensitive to such changes of the peak location. Plus, the proposed method is not hard to be parameterized, fact that also places it above the parent techniques.

8.3 Further Enhancements

There are some problems that remain open for the future research. As one of the targets of the thesis is to develop evolutionary techniques for classification problems, the creation of complete and as easy to adjust as possible techniques is desired, i.e. with a minimum number of evolutionary parameters to be set. One of the parameters that is very much dependent of the problem is the mutation strength: Adding an adaptation mechanism for it would not only open a path leading to substantially increased performance, but also ease the work of the user.

The identification of the drawbacks that exist within the competitive approach for classification is also envisaged. A new classification tool that would incorporate both the cooperative and competitive approaches could be a winning combination as it would resemble the situation from the evolution in nature more.

Another task that remains for future work is the development of a tool for detecting whether there exist a very high number of local and/or global optima or not. Knowing this information in advance can be very valuable for a technique dealing with a specific problem: It can help in setting then the proper values for parameters or even in deciding the method that should be employed for solving the problem. The topological species conservation could represent a good starting point for this purpose as it keeps track of all the different detected peaks and, if it were

not for the maximum number of seeds parameter, it would decide from the very early stages of the evolutionary process how many different niches exist in the solutions search space.

Appendix A

Considered Test Functions Suite

In the undertaken experiments, all functions were considered for maximization. However, in this section, they are outlined in the form they are usually encountered in the literature, that is, for most of them, for minimization. The only function among the presented ones that is proposed for maximization is the Waves function ($F5$).

1. Sphere (De Jong) function

$$F1(\overrightarrow{x}) = \sum_{i=1}^{n}(x_i^2) \tag{A.1}$$

where $-5.12 \leq x_i \leq 5.12$.

Although most of the tests are carried out to outline the ability of the techniques to deal with multimodal functions, F1 is considered to show that even in case of only one optimum, the considered methods still perform sufficiently well.

2. Himmelblau function

$$F2(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{A.2}$$

where -4 $\leq x \leq$ 4 and -6 $\leq y \leq$ 6.

Himmelblau function has four global optima, $F2(x, y)$ = -200. The peaks are located on an almost plate platform and the four hills are barely observable.

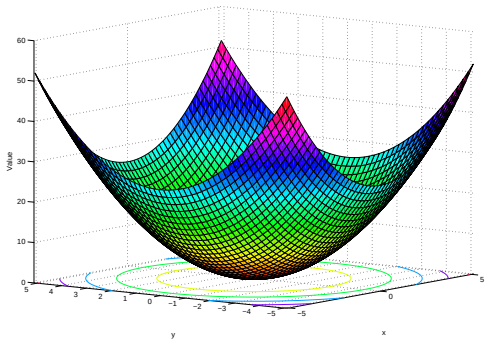

Figure A.1: De Jong function for $n = 2$.

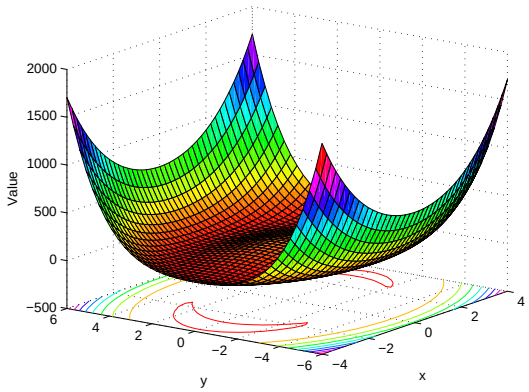

Figure A.2: Himmelblau function.

Table A.1: All 6 optima of the Six-Hump Camel Back function

Optimum type	Point	$F3$ value
Global	(-0.0898, 0.7126)	1.0316
Global	(0.0898, -0.7126)	1.0316
Local	(-1.7036, 0.7961)	0.2155
Local	(1.7036, -0.7961)	0.2155
Local	(-1.6071, -0.5687)	-2.1043
Local	(1.6071, 0.5687)	-2.1043

3. Six-Hump Camel Back function

$$F3(x,y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + 4(y^2 - 1)y^2 \tag{A.3}$$

where -1.9 $\leq x \leq$ 1.9 and -1.1 $\leq y \leq$ 1.1.

The aim for the *Six-Hump Camel Back* function is to locate the two global optima and the four other local optima. (Table A.1). The main issue here is that the two local optima with the same value -2.1043 are not much higher than their neighboring regions and thus can easily be missed by a multimodal evolutionary algorithm.

4. Shifted Six-Hump Camel Back function

$$F4(x,y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + 10xy + 4(100y^2 - 1)100y^2 \tag{A.4}$$

where -1.9 $\leq x \leq$ 1.9 and -0.11 $\leq y \leq$ 0.11.

Having equally distant optima would advantage a radii-based techniques as a proper value for the radius would aid in detecting all peaks. Plus, as real-world problems do not usually have a regular fitness landscape, $F3$ is rescaled to $F4$ in order to have the optima, two by two more distant from each other. However, the graphical representation looks very similar to that of the standard formulation.

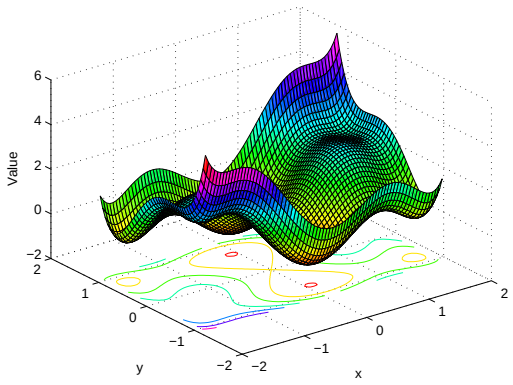

Figure A.3: Six-Hump Camel Back function.

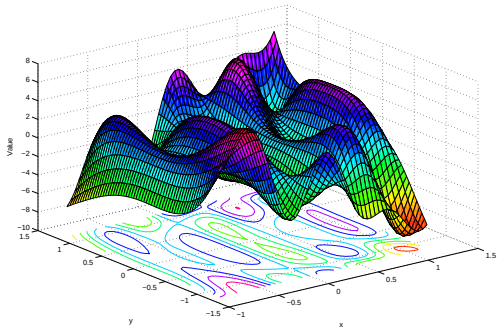

Figure A.4: Waves function.

5. Waves function

$$F4(x,y) = (0.3x)^3 - (y^2 - 4.5y^2)xy - 4.7cos(3x - y^2(2+x))sin(2.5\Pi x)) \tag{A.5}$$

where $-0.9 \leq x \leq 1.2, -1.2 \leq y \leq 1.2$.

Waves function contains 10 optima to be detected, some of which being even more difficult to find as they lie on the border or on flat hills. $F5$ function is asymmetric and contains an irregularly formed basin structure which proves to be very misleading especially for radius based evolutionary techniques (see Chapter 7).

6. Schaffer function

$$F6(x,y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2} \tag{A.6}$$

where -20 $\leq x \leq$ 20 and -20 $\leq y \leq$ 20.

The aim for the *Schaffer* function is to detect the global optimum $F_6(x,y) = 0$ that can be escaped because of the high number of local optima around it and because the difference between the values of the local optima and the value of the global optimum is very small (of order 10^{-3}).

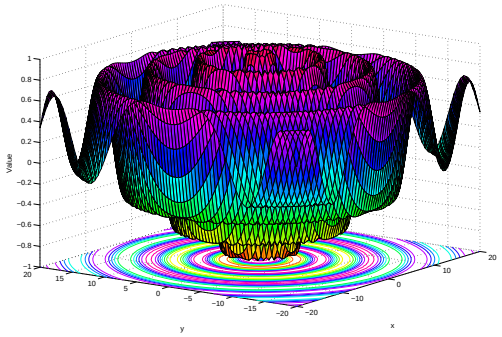

Figure A.5: Schaffer function for $n = 2$.

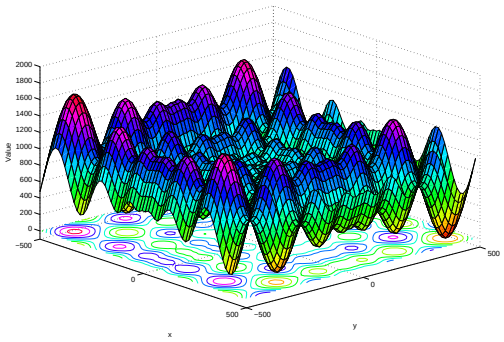

Figure A.6: Schwefel function for $n = 2$; right most, the global optimum.

7. Schwefel function (100D)

$$F7(\overrightarrow{x}) = 418.9829n + \sum_{i=1}^{n} -x_i sin\sqrt{|x_i|} \tag{A.7}$$

where -500 $\leq x_i \leq$ 500 and $i \in \{1, 2, ..., n\}$.

Schwefel function is a well known benchmark function which is very misleading for evolutionary algorithms; it has a high number of local optima and the global one ($F7(\overrightarrow{x}) = 0$) can easily be missed.

8. Rastrigin function

$$F8(\overrightarrow{x}) = -20 - \sum_{i=1}^{n}(-x_i^2 + 10cos(2\Pi x_i)) \tag{A.8}$$

where $-5.12 \leq x_i \leq 5.12$.

The difficulty with the Rastrigin function is that the global optimum is surrounded by a large number of very close local optima with only a small difference in their values as compared to it.

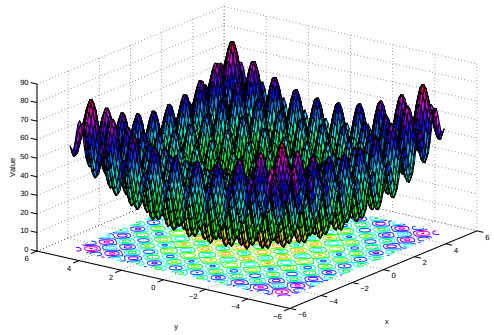

Figure A.7: Rastrigin function for $n = 2$.

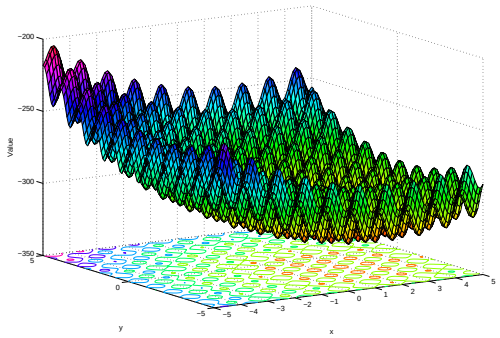

Figure A.8: Shifted Rastrigin function for $n = 2$.

9. Shifted Rastrigin function

$$F9(\overrightarrow{x}) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\Pi x_i) + 10) + f\_bias \tag{A.9}$$

where $-5 \leq x_i \leq 5$.

The complete description for both the shifted Rastrigin and the rotated hybrid composition functions can be found in [Suganthan et al., 2005], as they are part of a set of 25 benchmark problems used in a contest during CEC 2005 (shifted Rastrigin can be identified as F_9 in the cited technical report). The difficulty with the shifted Rastrigin function is again that the global optimum is surrounded by a large number of very close local optima with only a small difference in their values as compared to it.

10. Rotated hybrid composition function

$F10$ function represents a composition of 5 functions: Ackley, Rastrigin, Sphere, Weierstrass and Griewank, where $-5.12 \leq x_i \leq 5.12$. According to the description in [Suganthan et al., 2005], it has a huge number of optima, different functions properties are mixed together, sphere function adds some flat areas and a local optimum is set on the origin. 11 algorithms were

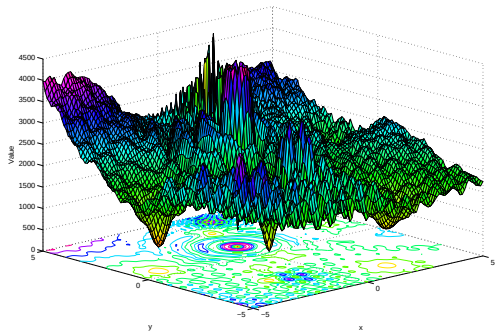

Figure A.9: $F11$ for $n = 2$.

tested in the CEC 2005 contest mentioned above and none of them was able to find the global optimum in any run when 10 dimensions were considered.

Appendix B

Real-world Problems Addressed in the Thesis

There are five data sets for classification considered in current work, three coming from the University of California at Irvine (UCI) Repository of Machine Learning Databases[1] and two with raw information.

The three widely used benchmark classification problems from UCI are Fisher's Iris, Pima-Indian Diabetes and Breast Cancer data sets. Some of their features are described in the next four subsections.

With the purpose of testing on an unpredictable environment that is usually associated with raw data, a practical data set containing e-mails of both kind, regular and undesired ones (spam), represents the fifth problem which is further described. Additionally, a real-world medical data set, courtesy of the University Hospital in Craiova, Romania, is also considered.

B.1 Fisher's Iris data set

There are 150 samples with 3 possible classes pertaining to this data set. Each class refers to a type of iris plant, namely Iris Setosa, Iris Versicolour and Iris Virginica. Each sample consists of 4 attributes which denote the length and width for petals and sepals of the iris flowers. Samples are equally distributed among classes and the intervals of the four attributes are balanced.

B.2 Pima-Indian Diabetes data set

The Diabetes data set was given to the UCI repository by the Johns Hopkins University. Prior to that, the university selected cases from a larger database owned by the National Institute of Diabetes and Digestive and Kidney Diseases to create it.

All objects in the data set represent females of at least 21 years age, of Pima Indian heritage, living near Phoenix, Arizona, USA. For each object in the data set, there are 8 attributes (either discrete or continuous) containing personal data, e.g. age, number of pregnancies, and medical

[1] Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html

Table B.1: Attributes and their corresponding ranges in Pima Diabetes problem and a brief statistical analysis of the attribute values

Attribute	Interval	Mean	Standard deviation
Number of times pregnant	[0, 5]	3.8	3.4
Plasma glucose concentration in an oral glucose tolerance test	[0, 199]	120.9	32
Diastolic blood pressure	[0, 122]	69.1	19.4
Triceps skin fold thickness	[0, 99]	20.5	16
2-Hour serum insulin	[0, 846]	79.8	115.2
Body mass index	[0, 67]	32.0	7.9
Diabetes pedigree function	[0.078, 2.42]	0.5	0.3
Age	[21, 81]	33.2	11.8

data, e.g. blood pressure, body mass index, result of glucose tolerance test etc (see Table B.1). The last attribute (the outcome) is a discrete one - the diagnosis, which is either 0 (negative) or 1 (positive). 34.9% of the cases in the data set are assigned diabetes positive. The total number of cases is 768.

B.3 Breast Cancer data set

The data set was obtained from the University of Wisconsin Hospital, Madison, USA, from Dr. William H. Wolberg. It contains 699 observations on 9 discrete cytological factors. The factors that influence the diagnosis are: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial, cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses; all attributes have the same domain between 1 and 10. The objective is to identify whether a sample corresponds to either a benign or a malignant tumour. Class distribution is 65.5% for benign and 34.5% for malignant. There are 16 missing values for attribute 6. In all experiments, they are replaced by the average value for that attribute.

B.4 Spam raw data set

The considered data is extracted from e-mails in the test collections available at http://spamassassin.org/publiccorpus. The data on which the experiments were conducted comprises of 1000 e-mails, 500 for each category, spam or non-spam.

B.5 Hepatic Cancer Early Diagnosis

Data is collected from the University Hospital in Craiova, Romania. Hepatocellular carcinoma (HCC/hepatoma) is the primary malignancy (cancer) of the liver that ranks fifth in frequency among all malignancies in the world. In patients with a higher suspicion of HCC, the best method of diagnosis involves a scan of the abdomen, but only at a high cost. A cheap and effective alternative consists in detecting small or subtle increases for serum enzymes levels. Consequently,

based on a set of 14 significant serum enzymes, a group of 299 individuals and two possible outcomes (HCC and non-HCC), the aim is to provide an efficient computational means of checking the consistency of decision making in the early detection of HCC at a significantly low expense.

Bibliography

[Bäck, 1996] Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK.

[Bäck et al., 1997] Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK.

[Bartz-Beielstein, 2005] Bartz-Beielstein, T. (2005). *New Experimentalism Applied to Evolutionary Computation*. PhD thesis, University of Dortmund, Dortmund, Germany.

[Bartz-Beielstein, 2006] Bartz-Beielstein, T. (2006). *Experimental Research in Evolutionary Computation – The New Experimentalism*. Natural Computing Series. Springer, Berlin.

[Bartz-Beielstein et al., 2004a] Bartz-Beielstein, T., Parsopoulos, K. E., and Vrahatis, M. N. (2004a). Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433.

[Bartz-Beielstein et al., 2004b] Bartz-Beielstein, T., Parsopoulos, K. E., and Vrahatis, M. N. (2004b). Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433.

[Beasley et al., 1993] Beasley, D., Bull, R., and Martin, R. R. (1993). Sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125.

[Bennett, 1997] Bennett, K. P. (1997). A support vector machine approach to decision trees. Technical Report 97–100, Rensselaer Polytechnic Institute Troy.

[Cavicchio, 1970] Cavicchio, D. J. (1970). *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, Ann Arbor.

[Dasgupta, 1999] Dasgupta, D. (1999). *New Ideas in Optimization*, chapter Information processing in the immune system, pages 161–166. McGraw-Hill Ltd., UK, London.

[de Castro and Zuben, 2002] de Castro, L. N. and Zuben, F. J. V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 6(3):239–251.

[Deb and Goldberg, 1989] Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California. Morgan Kaufman.

[Duch et al., 1999] Duch, W., Adamczak, R., Grabczewski, K., and Zal, G. (1999). Hybrid neural-global minimization method of logical rule extraction. *Journal of Advanced Computational Intelligence*, 3(5).

[Dumitrescu, 2000a] Dumitrescu, D., editor (2000a). *Algoritmi Genetici si Strategii Evolutive Aplicatii in Inteligenta Artificiala si in Domenii Conexe*. Editura Albastra, Cluj-Napoca, Romania.

[Dumitrescu, 2000b] Dumitrescu, D. (2000b). Genetic chromodynamics. *Studia Universitatis Babes-Bolyai Cluj-Napoca, Ser. Informatica*, 45(1):39–50.

[Dumitrescu, 2004] Dumitrescu, D. (2004). Evolutionary clustering using adaptive prototypes. *Studia Universitatis Babes-Bolyai Cluj-Napoca, Ser. Informatica*, 49:15–20.

[Dumitrescu et al., 2000] Dumitrescu, D., Lazzerini, B., Jain, L. C., and Dumitrescu, A. (2000). *Evolutionary computation*. CRC Press, Inc., Boca Raton, FL, USA.

[Dumitrescu and Stoean, 2006a] Dumitrescu, D. and Stoean, C. (2006a). Genetic chromodynamics a novel evolutionary heuristic for multimodal optimization. In Guerrero-Bote, V., editor, *First International Conference on Multidisciplinary Information Sciences and Technologies (InSciT2006)*, volume 2, pages 238–243, Merida, Spain. Current Research in Information Sciences and Technologies, Open Institute of Knowledge.

[Dumitrescu and Stoean, 2006b] Dumitrescu, D. and Stoean, C. (2006b). Genetic chromodynamics metaheuristic for multimodal optimization. *WSEAS Transactions on Information Science and Application*, 3(8):1444–1452. indexed by INSPEC (IEE).

[Eiben and Smith, 2003] Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Heildelberg, NY.

[Fogel, 1997] Fogel, D. B. (1997). *Handbook of Evolutionary Computation*, chapter Why Evolutionary Computation? Oxford University Press.

[Giordana et al., 1994] Giordana, A., Saitta, L., and Zini, F. (1994). Learning disjunctive concepts by means of genetic algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 96–104.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[Goldberg and Richardson, 1987] Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and Their Application*, pages 41–49. J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum.

[Gordon and Whitley, 1993] Gordon, V. S. and Whitley, L. D. (1993). Serial and parallel genetic algorithms as function optimizers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 177–183, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Gorunescu and Millard, 2004] Gorunescu, R. and Millard, P. H. (2004). An evolutionary model of a multidisciplinary review panel for admission to long-term care. In *ICCC 2004*, page 181185, Baile Felix, Oradea.

[Graham, 2002] Graham, P. (2002). A plan for spam. http://www.paulgraham.com/spam.html.

[Holland, 1986] Holland, J. H. (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. *Machine Learning*, 2:593–623.

[Iacus and Porro, 2006] Iacus, S. and Porro, G. (2006). Missing data imputation, classification, prediction and average treatment effect estimation via random recursive partitioning. Technical report, UNIMI - Research Papers in Economics, Business, and Statistics. Statistics and Mathematics.

[Jong, 1975] Jong, K. A. D. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor.

[Juille and Pollack, 1996] Juille, H. and Pollack, J. B. (1996). Dynamics of co-evolutionary learning. In *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 526–534.

[Juille and Pollack, 1998] Juille, H. and Pollack, J. B. (1998). Coevolutionary learning: a case study. In *Proceedings of the 15th International Conference on Machine Learning*, pages 251–259.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimisation. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ.

[Lee and Roh, 2002] Lee, K. Y. and Roh, M. I. (2002). An efficient genetic algorithm using gradient information for ship structural design optimization. Technical report, Schiffstechnik Bd. 48 2001/Ship Technology Research.

[Li et al., 2002] Li, J. P., Balazs, M. E., Parks, G. T., and Clarkson, P. J. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234.

[Luchian et al., 1994] Luchian, S., Luchian, H., and Petriuc, M. (1994). Evolutionary automated classification. *Proc. of International Conference on Evolutionary Computation*, 2:585–588.

[Mahfoud, 1995] Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois, Urbana, Illinois.

[Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK.

[Nicholas, 2003] Nicholas, T. (2003). Using adaboost and decision stumps to identify spam e-mail. *Stanford Natural Language Processing Group, CS224N/Ling237*.

[Paredis, 1994a] Paredis, J. (1994a). Coevolutionary constraint satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, pages 46–55.

[Paredis, 1994b] Paredis, J. (1994b). Steps towards coevolutionary classification neural networks. *Artificial Life IV*.

[Paredis, 1997] Paredis, J. (1997). Coevolving cellular automata: Be aware of the red queen! In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 393–399.

[Paredis, 1998] Paredis, J. (1998). *Handbook of Evolutionary Computation*, chapter Coevolutionary Algorithms. Oxford University Press.

[Paredis, 1999] Paredis, J. (1999). *New Ideas in Optimization*, chapter Constraint Satisfaction with Coevolution, pages 359–366. McGraw-Hill Ltd., UK, London.

[Paredis and Westra, 1997] Paredis, J. and Westra, R. (1997). Coevolutionary computation for path planning. In *Proceedings EUFIT*, pages 394–398.

[Parrott and Li, 2004] Parrott, D. and Li, X. (2004). A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Proceeding of the 2004 Congress on Evolutionary Computation (CEC 2004)*, pages 98–103.

[Potter and Jong, 1994] Potter, M. A. and Jong, K. A. D. (1994). A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature (PPSN 1994)*, pages 249–257, Berlin, Germany. Springer-Verlag.

[Potter and Jong, 2000] Potter, M. A. and Jong, K. A. D. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1).

[Potter et al., 2001] Potter, M. A., Meeden, L. A., and Schultz, A. C. (2001). Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the 17th International Conference on Artificial Intelligence*, pages 1337–1343.

[Prechelt, 1994] Prechelt, L. (1994). Proben 1 a set of benchmark and benchmarking rules for neural network training algorithms. Technical Report 21/94, University of Karlsruhe, Institute for Program Structures and Data Organization (IPD).

[Preuss, 2006] Preuss, M. (2006). Niching Prospects. In *Bioinspired Optimization Methods and their Applications (BIOMA 2006)*, pages 25–34. Jozef Stefan Institute, Ljubljana, Slovenia.

[R. P. Wiegand and Jong, 2001] R. P. Wiegand, W. C. L. and Jong, K. A. D. (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of GECCO 2001*, pages 1235–1245.

[Schwefel, 1997] Schwefel, H. P. (1997). *Handbook of Evolutionary Computation*, chapter Why Evolutionary Computation? Oxford University Press.

[Siegel, 1994] Siegel, E. V. (1994). Competitively evolving decision trees against fixed training cases for natural language processing. *Advances in Genetic Programming*, pages 409–423.

[Smithies et al., 2004] Smithies, R., Salhi, S., and Queen, N. (2004). Adaptive hybrid learning for neural networks. *Neural Computation*, 16(1):139–157.

[Stoean, 2004a] Stoean, C. (2004a). Hierarchical learning text categorization. In *ICCC 2004*, pages 383–387, Baile Felix, Oradea.

[Stoean, 2004b] Stoean, C. (2004b). A new technique for text categorization. application to spam filtering. In *Proceedings of the Zilele Academice Clujene Symposium*, volume 1, pages 95–100, Babes-Bolyai University of Cluj-Napoca.

[Stoean, 2004c] Stoean, C. (2004c). On classifying good and bad e-mails. In *4th International Conference on Artificial Intelligence and Digital Communications (AIDC 2004)*, pages 79–85, Craiova, Romania. Research Notes in Artificial Intelligence and Digital Communications, N. Tandareanu (Ed.), Reprograph Press.

[Stoean, 2006] Stoean, C. (2006). Various collaborator selection pressures for cooperative coevolution for classification. In Tandareanu, N., editor, *6th International Conference on Artificial Intelligence and Digital Communications (AIDC 2006)*, pages 45–53, Thessaloniki, Greece. Research Notes in Artificial Intelligence and Data Communications, Reprograph Press.

[Stoean, 2007] Stoean, C. (2007). Competitive coevolution for classification. In *7th International Conference on Artificial Intelligence and Digital Communications (AIDC 2007)*, pages 28–39, Craiova, Romania. Research Notes in Artificial Intelligence and Digital Communications, N. Tandareanu (Ed.), Reprograph Press.

[Stoean and Dumitrescu, 2005a] Stoean, C. and Dumitrescu, D. (2005a). Cloning within genetic chromodynamics. In *Proceedings of the Colocviul Academic Clujean de Informatica*, pages 9–14, Babes-Bolyai University of Cluj-Napoca, Cluj-Napoca, Romania.

[Stoean and Dumitrescu, 2005b] Stoean, C. and Dumitrescu, D. (2005b). A new algorithm within genetic chromodynamics. applications to function optimization and classification. Technical report, Department of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University.

[Stoean and Dumitrescu, 2005c] Stoean, C. and Dumitrescu, D. (2005c). On improving classification accuracy for diabetes diagnosis by evolving weights. *Scientific Bulletin, University of Pitesti, Mathematics and Computer Science Series*, 1(11):67–74.

[Stoean and Dumitrescu, 2006] Stoean, C. and Dumitrescu, D. (2006). Elitist generational genetic chromodynamics as a learning classifier system. *Annals of University of Craiova, Mathematics and Computer Science Series*, 33(1):132–140.

[Stoean et al., 2006a] Stoean, C., Dumitrescu, D., Preuss, M., and Stoean, R. (2006a). Cooperative coevolution for classification. In D. Dumitrescu, L. P., editor, *Bio-Inspired Computing: Theory and Applications (BIC-TA 2006)*, pages 289–298, China.

[Stoean et al., 2008a] Stoean, C., Dumitrescu, D., Preuss, M., and Stoean, R. (in press, 2008a). Cooperative coevolution as a paradigm for classification. *Journal of Universal Computer Science*.

[Stoean et al., 2005a] Stoean, C., Gorunescu, R., and Dumitrescu, D. (2005a). A new evolutionary model for the optimization of multimodal functions. In Teodorescu, H. N., Watada, J., Aluja, J. G., and Mihaila, M., editors, *The Anniversary Symposium Celebrating 25 Years of the Seminar Grigore Moisil and 15 Years of the Romanian Society for Fuzzy Systems and A.I., Selected Papers*, pages 65–72, Iasi, Romania. Performantica Press.

[Stoean et al., 2004a] Stoean, C., Gorunescu, R., Preuss, M., and Dumitrescu, D. (2004a). Evolutionary detection of rules for text categorization. application to spam filtering. In *Third European Conference on Intelligent Systems and Technologies (ECIT'2004)*, pages 87–95, Iasi, Romania. Intelligent Systems, Selected papers, H. N. Teodorescu (Ed.), Performantica Press.

[Stoean et al., 2004b] Stoean, C., Gorunescu, R., Preuss, M., and Dumitrescu, D. (2004b). An evolutionary learning spam filter system. In Petcu, D., Negru, V., Zaharie, D., and Jebelean, T., editors, *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2004)*, pages 512–522, Timisoara, Romania. Mirton Press.

[Stoean et al., 2005b] Stoean, C., Gorunescu, R., Preuss, M., and Dumitrescu, D. (2005b). An evolutionary learning classifier system applied to text categorization. *Annals of West University of Timisoara, Mathematics and Computer Science Series*, XLII(special issue 1):265–278.

[Stoean et al., 2006b] Stoean, C., Preuss, M., Dumitrescu, D., and Stoean, R. (2006b). A cooperative coevolutionary algorithm for multi-class classification. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006)*, pages 7–14, Timisoara, Romania.

[Stoean et al., 2006c] Stoean, C., Preuss, M., Dumitrescu, D., and Stoean, R. (2006c). Cooperative evolution of rules for classification. *IEEE Post-proceedings Symbolic and Numeric Algorithms for Scientific Computing 2006*, pages 317–322.

[Stoean et al., 2005c] Stoean, C., Preuss, M., Gorunescu, R., and Dumitrescu, D. (2005c). Elitist generational genetic chromodynamics - a new radii-based evolutionary algorithm for multimodal optimization. In *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 1839–1846, Edinburgh, UK. IEEE Computer Society Press.

[Stoean et al., 2007a] Stoean, C., Preuss, M., Stoean, R., and Dumitrescu, D. (2007a). Disburdening the species conservation evolutionary algorithm of arguing with radii. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1420–1427, New York, NY, USA. ACM Press.

[Stoean et al., 2008b] Stoean, C., Preuss, M., Stoean, R., and Dumitrescu, D. (submitted, 2008b). Multimodal optimization by means of a topological species conservation algorithm. *IEEE Transactions on Evolutionary Computation*.

[Stoean et al., 2007b] Stoean, C., Stoean, R., and El-Darzi, E. (2007b). Breast cancer diagnosis by means of cooperative coevolution. In *3rd ACM International Conference on Intelligent Computing and Information Systems*, pages 493–497.

[Stoean et al., 2005d] Stoean, C., Stoean, R., Preuss, M., and Dumitrescu, D. (2005d). Diabetes diagnosis through the means of a multimodal evolutionary algorithm. In *Proceedings of the First East European Conference on Health Care Modelling and Computation (HCMC 2005)*, pages 277–289, Craiova, Romania.

[Stoean et al., 2006d] Stoean, C., Stoean, R., Preuss, M., and Dumitrescu, D. (2006d). A cooperative evolutionary algorithm for classification. In *International Conference on Computers and Communications (ICCC 2006)*, pages 417–422, Baile Felix Spa - Oradea, Romania.

[Stoean et al., 2006e] Stoean, C., Stoean, R., Preuss, M., and Dumitrescu, D. (2006e). Spam filtering by means of cooperative coevolution. In Teodorescu, H. N., editor, *4th European Conference on Intelligent Systems and Technologies, (ECIT 2006), Advances in Intelligent Systems and Technologies Selected Papers*, pages 157–159, Iasi, Romania. Performantica Press.

[Stoean et al., 2008c] Stoean, C., Stoean, R., Preuss, M., and Dumitrescu, D. (2008c). Coevolution for classification. Technical Report CI-239/08, Collaborative Research Center on Computational Intelligence, University of Dortmund.

[Stoean et al., 2007c] Stoean, R., Preuss, M., Stoean, C., and Dumitrescu, D. (2007c). Concerning the potential of evolutionary support vector machines. *Proc. of the IEEE Congress on Evolutionary Computation*, pages 1436–1443.

[Stoean et al., 2007d] Stoean, R., Preuss, M., Stoean, C., El-Darzi, E., and Dumitrescu, D. (submitted, 2007d). An evolutionary resemblant to support vector machines for classification and regression. *Journal of the Operational Research Society*.

[Storn and Price, 1995] Storn, R. and Price, K. (1995). Differential evolution; a simple and efficient heuristic for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA.

[Suganthan et al., 2005] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-. P., Auger, A., and Tiwari, S. (2005). Problem definition and evaluation criteria for the cec 2005 special session on real-parameter optimization. In *Technical Report*, Nanyang Technological University, Singapore.

[Thomsen, 2004] Thomsen, R. (2004). Multimodal optimization using crowding-based differential evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, volume 2, pages 1382–1389.

[Thomsen et al., 2000] Thomsen, R., Rickers, P., and Krink, T. (2000). A religion-based spatial model for evolutionary algorithms. In *Proceedings of the Sixth Conference on Parallel Problem Solving from Nature (PPSN 2000)*, pages 348–356, Berlin, Germany. Springer-Verlag.

[Ursem, 1999] Ursem, R. K. (1999). Multinational evolutionary algorithms. In *Congress of Evolutionary Computation (CEC 1999)*, volume 3, pages 1633–1640, Piscataway, NJ. IEEE Press.

[Ursem, 2000] Ursem, R. K. (2000). Multinational gas: Multimodal optimization techniques in dynamic environments. In *Genetic and Evolutionary Computation Conference (GECCO 2000)*, volume 1, pages 19–26. Morgan Kaufmann.

[Veenman, 1996] Veenman, C. J. (1996). Positional genetic programming: Genetic algorithms with encoded tree structures. Master's thesis, Vrije Universiteit, Amsterdam.

[Wiegand, 2003] Wiegand, R. P. (2003). *Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, Department of Computer Science, George Mason University.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*.

[Yang and Honavar, 1991] Yang, J. and Honavar, V. (1991). Experiments with the cascade-correlation algorithm. Technical Report 91-16, Iowa State University.

[Yao and Liu, 1997] Yao, X. and Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713.

[Zaharie, 2004a] Zaharie, D. (2004a). Extensions of differential evolution algorithms for multimodal optimization. In Petcu, D., Negru, V., Zaharie, D., and Jebelean, T., editors, *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2004)*, pages 523–534, Timisoara, Romania. Mirton Press.

[Zaharie, 2004b] Zaharie, D. (2004b). A multipopulation differential evolution algorithm for multimodal optimization. In Matousek, R. and Osmera, P., editors, *10th International Conference on Soft Computing (Mendel'04)*, pages 17–22, Brno, Czech Republic.

Index