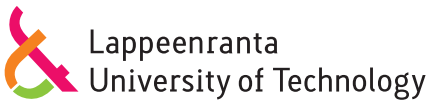


Saku Kukkonen

GENERALIZED DIFFERENTIAL EVOLUTION FOR GLOBAL MULTI-OBJECTIVE OPTIMIZATION WITH CONSTRAINTS

Thesis for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium of the Student Union House at Lappeenranta University of Technology, Lappeenranta, Finland on the 19th of May, 2012, at noon.

Acta Universitatis
Lappeenrantaensis **475**

Supervisors Professor Jouni Lampinen
Laboratory of Information Technology
University of Vaasa
Finland

Docent Jorma K. Mattila
Laboratory of Applied Mathematics
Lappeenranta University of Technology
Finland

Reviewers Professor Kaisa Miettinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Professor Daniela Zaharie
Faculty of Mathematics and Computer Science
West University of Timisoara
Romania

Opponent Professor Carlos A. Coello Coello
Computer Science Department
Center for Research and Advanced Studies
of the National Polytechnical Institute
Mexico

ISBN 978-952-265-235-5
ISBN 978-952-265-236-2 (PDF)
ISSN 1456-4491

Lappeenrannan teknillinen yliopisto
Digipaino 2012

To my father
LL.M. Pauli Sakari Kukkonen
(1945 – 2004)

Preface

The work presented in this thesis has been carried out mainly at the Laboratory of Information Processing in the Department of Information Technology at Lappeenranta University of Technology, Finland, during the years 2003–2007. A part of the work has been done at the Kanpur Generic Algorithms Laboratory (KanGAL) of Indian Institute of Technology Kanpur, India, during the years 2005–2006. The work has been finalized at the Laboratory of Applied Mathematics of Lappeenranta University of Technology.

The biggest compliments belong to my main supervisor, Jouni Lampinen; I am truly grateful for an excellent research topic as well as his tremendous support and guidance during this thesis project. I also wish to thank Kalyanmoy Deb for giving me the opportunity to visit his Kanpur Genetic Algorithms Laboratory. It has been a great honor to be able to work with one of the greatest researchers in the field. Finally, I thank Jorma Mattila for helping me to complete this work.

I thank my reviewers, Kaisa Miettinen and Daniela Zaharie, for their critical review of my thesis manuscript and their valuable comments. Kaisa Miettinen, in particular, made a vast number of detailed and thoughtful suggestions that helped me to improve the thesis significantly.

I would also like to thank my fellow researchers co-authoring papers closely related to this thesis work. These co-authors are Jani Rönkkönen, Jouni Sampo, Kenneth Price, Nirupam Chakraborti, Sujit Jangam, Ankur Sinha, and Junhong Liu.

I gratefully acknowledge the financial support of the East Finland Graduate School in Computer Science and Engineering (ECSE), the Research Foundation of Lappeenranta University of Technology (Lappeenrannan teknillisen yliopiston tukisäätiö), Finnish Foundation for Technology Promotion (Tekniikan edistämissäätiö), the Finnish Cultural Foundation (Suomen Kulttuurirahasto) and the South Karelia Regional Fund (Etelä-Karjalan rahasto), the Finnish Information Processing Association (Tietotekniikan Liitto ry), and the Centre for International Mobility (CIMO).

I thank all those people who have supported and helped me during these years. Colleagues at work have provided support in various ways and shared many refreshing moments (*e.g.*, coffee breaks and sauna evenings). Some people that I would like to mention by name are Ville Kyrki, Leena Ikonen, Lasse Lensu, Pekka Paalanen, Satu Alaoutinen, Taina Juuti, Ilmari Laakkonen, Timo Aittokoski, Vitaliy Feoktistov, Ashish Anand, Santosh Tiwari, Shamik Chaudhuri, Peter Jones, Evan Hughes, and Mariya Loginova.

I am grateful for my existence to my parents, Tuula and Pauli, who mixed their genes at the beginning of my birth year 1975. Also, I wish to thank my two sisters, Anu and Anne, for sharing and tolerating many years.

My roots lie in the small village of Kerimäki in Eastern Finland. I would like to acknowledge two elementary and high school teachers, Raimo Malinen and Matti Martikainen, for inspiring me with an interest in the natural sciences.

Lastly, I would like to thank myself for finally completing this work after many years.

Consummatum est, di bene vortant.

Lappeenranta, May 2012

Saku Kukkonen

Abstract

Saku Kukkonen
Generalized Differential Evolution for Global Multi-Objective Optimization with Constraints
Lappeenranta, 2012
108 p.

Acta Universitatis Lappeenrantaensis 475
Diss. Lappeenranta University of Technology

ISBN 978-952-265-235-5
ISBN 978-952-265-236-2 (PDF)
ISSN 1456-4491

The objective of this thesis work is to develop and study the Differential Evolution algorithm for multi-objective optimization with constraints. Differential Evolution is an evolutionary algorithm that has gained in popularity because of its simplicity and good observed performance. Multi-objective evolutionary algorithms have become popular since they are able to produce a set of compromise solutions during the search process to approximate the Pareto-optimal front.

The starting point for this thesis was an idea how Differential Evolution, with simple changes, could be extended for optimization with multiple constraints and objectives. This approach is implemented, experimentally studied, and further developed in the work. Development and study concentrates on the multi-objective optimization aspect.

The main outcomes of the work are versions of a method called Generalized Differential Evolution. The versions aim to improve the performance of the method in multi-objective optimization. A diversity preservation technique that is effective and efficient compared to previous diversity preservation techniques is developed. The thesis also studies the influence of control parameters of Differential Evolution in multi-objective optimization. Proposals for initial control parameter value selection are given. Overall, the work contributes to the diversity preservation of solutions in multi-objective optimization.

Keywords: evolutionary algorithms, Differential Evolution, multi-objective optimization, diversity preservation, constraint handling, influence of control parameters
UDC 004.021:004.421:519.85:519.6

$\prec$	Pareto-dominance relation
$\preceq$	Weak Pareto-dominance relation
$\prec_c$	Constraint-dominance relation
$\preceq_c$	Weak constraint-dominance relation
$\aleph$	Cardinality of a non-dominated set
Δ	Spread
η_c	Distribution index for real variable crossover operation in NSGA-II
η_m	Distribution index for real variable polynomial mutation in NSGA-II
c	Change of the population standard deviation between successive generations due to the crossover and mutation operations
$C(A, B)$	Set coverage metric between two non-dominated sets A and B
CD	Crowding distance
CR	Crossover control parameter of DE
D	Number of decision variables
$\bar{D}$	Maximum spread, normalized version
d_i	Distance measure for measuring the distance of a particular solution i to its neighbor solutions
ER	Error ratio
F	Mutation control parameter of DE
f_m	mth objective function
G	Generation number
GD	Generational distance
g_k	kth constraint function
G_{max}	Maximum number of generations
$HV(A, B)$	Hypervolume between two non-dominated sets A and B
$I_\epsilon(A, B)$	ϵ-indicator between two non-dominated sets A and B
IGD	Inverted generational distance
K	Number of constraints
M	Number of objectives
NP	Population size
p_c	Crossover probability of real variable in NSGA-II
p_m	Mutation probability of real variable in NSGA-II
S	Spacing

$\boldsymbol{u}$	Trial vector
$\boldsymbol{x}$	Old decision vector
$\boldsymbol{z}^*$	Ideal solution
$\boldsymbol{z}^{**}$	Utopian solution
ACO	Ant Colony Optimization
ADE	Adaptive Differential Evolution
APDE	Adaptive Pareto Differential Evolution
AS-MODE	Adaptive Multi-Objective Differential Evolution with Stochastic Coding Strategy
BNH	Constrained bi-objective test problem by Binh and Korn
CEC	Congress on Evolutionary Computation
CFDE	Cluster-Forming Differential Evolution
CMA-ES	Covariance Matrix Adaptation Evolutionary Strategy
DE	Differential Evolution
DE/best/1/bin	DE strategy based on mutation of the best individual of the population
DEMO	Differential Evolution for Multiobjective Optimization
DEMORS	Differential Evolution for Multiobjective Optimization with Random Sets
DE/rand/1/bin	DE strategy based on mutation of a random individual of the population
DTLZ	Deb-Thiele-Laumanns-Zitzler test problem set
EA	Evolutionary algorithm
ENNS	Equal-average Nearest Neighbor Search
ϵ-CCDE	ϵ-Constraint method with Cultured DE
ϵ-ODEMO	Differential Evolution algorithm based on ϵ-dominance and an orthogonal design method
EMO	Evolutionary Multiobjective Optimization
ϵ-MOEA	ϵ-dominance Multi-Objective Evolutionary Algorithm
ϵ-MyDE	A DE algorithm by Santana-Quintero and Coello Coello
ES	Evolutionary Strategy
GA	Genetic Algorithm
GDE	Generalized Differential Evolution
GDE1	First version of Generalized Differential Evolution
GDE2	Second version of Generalized Differential Evolution
GDE3	Third version of Generalized Differential Evolution

GP	Genetic Programming
IBEA	Indicator-Based Evolutionary Algorithm
MCDM	Multiple Criteria Decision-Making
MODE	Multi-Objective Differential Evolution
MODEA	Multi-Objective Differential Evolution Algorithm
MODE/D	Multiobjective Differential Evolution based on decomposition
MODE-LD+SS	Multi-Objective Differential Evolution with Local Dominance and Scalar Selection
MOEA	Multi-objective evolutionary algorithm
MOEA/D	Multiobjective evolutionary algorithm based on decomposition
MOGA	Multiple Objective Genetic Algorithm
MOOP	Multi-Objective Optimization Problem
NASA	National Aeronautics and Space Administration
NFL	No Free Lunch theorem by Wolpert and Macready
NIMBUS	Non-differentiable Interactive Multi-objective BUndle-based optimization System
NN	Nearest neighbor
NPGA	Niched-Pareto Genetic Algorithm
NSGA	Non-Dominated Sorting Genetic Algorithm
NSDE	Non-dominated Sorting DE
NSGA-II	Elitist Non-Dominated Sorting Genetic Algorithm
OSY	Constrained bi-objective test problem by Osyczka and Kundu
PAES	Pareto-Archived Evolution Strategy
PDE	Pareto(-frontier) Differential Evolution
PDEA	Pareto Differential Evolution Approach
PESA	Pareto Envelope-based Selection Algorithm
PSO	Particle Swarm Optimization
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Improved version of Strength Pareto Evolutionary Algorithm
SPDE	Self-adaptive Pareto Differential Evolution
SRN	A constrained bi-objective test problem by Srinivas and Deb
TNK	A constrained bi-objective test problem by Tanaka
VEDE	Vector Evaluated Differential Evolution
VEGA	Vector Evaluated Genetic Algorithm
WFG	Walking Fish Group test problem toolkit
ZDT	Zitzler-Deb-Thiele test problem set

I. Kukkonen, S., Lampinen, J., "Comparison of Generalized Differential Evolution Algorithm to other Multi-Objective Evolutionary Algorithms", *Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, Jyväskylä, Finland, July, 2004, 20 pages.

II. Kukkonen, S., Lampinen, J., "An Empirical Study of Control Parameters for Generalized Differential Evolution", *Proceedings of the Sixth Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005)*, Munich, Germany, September, 2005, 12 pages.

III. Kukkonen, S., Lampinen, J., "An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints", *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Lecture Notes in Computer Science (LNCS)*, Vol. 3242, Birmingham, England, September, 2004, pages 752–761.

IV. Kukkonen, S., Lampinen, J., "GDE3: The third Evolution Step of Generalized Differential Evolution", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, September, 2005, pages 443–450.

V. Kukkonen, S., Deb, K., "Improved Pruning of Non-Dominated Solutions Based on Crowding Distance for Bi-Objective Optimization Problems", *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July, 2006, pages 3995–4002.

VI. Kukkonen, S., Lampinen, J., "An Empirical Study of Control Parameters for The Third Version of Generalized Differential Evolution (GDE3)", *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July, 2006, pages 7355–7362.

VII. Kukkonen, S., Deb, K., "A Fast and Effective Method for Pruning of Non-Dominated Solutions in Many-Objective Problems", *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX), Lecture Notes in Computer Science (LNCS)*, Vol. 4193, Reykjavik, Iceland, September, 2006, pages 553–562.

In this thesis, these publications are referred to as *Publication I*, *Publication II*, *Publication III*, *Publication IV*, *Publication V*, *Publication VI*, and *Publication VII*. Other relevant papers by the author related to the present thesis are: [79–89].

1 Introduction **15**

2 Background and Related Studies **19**
2.1 Evolutionary Algorithms 19
2.2 Multi-Objective Optimization with Constraints 21
2.2.1 Basic Concepts 21
2.2.2 Constraint Handling 25
2.2.3 Constraint-Domination Relation 27
2.2.4 Classical Methods for Multi-Objective Optimization 28
2.2.5 Multi-Objective Evolutionary Algorithms 31
2.2.6 Performance Evaluation in Multi-Objective Optimization 36
2.3 Differential Evolution 42
2.3.1 Basic Differential Evolution, DE/rand/1/bin 42
2.3.2 Differential Evolution for Multiple Objectives 45
2.3.3 Differential Evolution and Constraints 50

3 Generalized Differential Evolution **51**
3.1 First Version, GDE1 52
3.2 Second Version, GDE2 53
3.3 Third Version, GDE3 56
3.3.1 Diversity Preservation for Bi-Objective Problems 63
3.3.2 Diversity Preservation for Many-Objective Problems 63
3.4 Numerical Comparison of the GDE Versions 70
3.5 Study of Control Parameter Values for GDE 70
3.6 Constrained Optimization with GDE 79

4 Conclusions and Discussion **83**

5 Errata and Additional Information to Publications **89**

Bibliography **93**

Appendices

I Test Problems **109**

II Program Codes for Performance Metrics **115**

III Publications **125**

Chapter I

Introduction

"It is needless to say that most real-world optimization problems are naturally posed as a multi-objective optimization problem."
– Kalyanmoy Deb

Optimization, a task confronted in daily life, is a search for one or multiple feasible solutions corresponding to the best possible values of one or multiple objectives of a problem [33, p. 1]. Feasible solutions satisfy possible constraints inherent in the problem. Multi-objective optimization means optimization of more than one objective simultaneously. Many practical problems are multi-objective by nature, and multi-objective optimization has, therefore, become an important research topic in the field of optimization. Since multiple objectives usually conflict with each other, the result for a multi-objective optimization problem (MOOP) is usually not a single solution but a set of solutions. These solutions represent the best compromises between the different objectives.

Evolutionary algorithms (EAs) are population based stochastic optimization methods that are inspired by Darwin's Theory of Evolution. EAs are able to deal with difficult objective functions which are, *e.g.*, discontinuous, non-convex, multi-modal, non-linear, and non-differentiable, and which pose difficulties to most traditional optimization methods. Since many practical problems include such difficult objectives, EAs have become popular during the last couple of decades. Developments in computer technology have also facilitated the use of EAs.

A further reason for the increased popularity of EAs in multi-objective optimization is that EAs are capable of providing multiple solution candidates during the search process, which is a desirable quality with MOOPs. Two of the most well known multi-objective EAs (MOEAs) are the elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) [35] and the improved Strength Pareto Evolutionary Algorithm (SPEA2) [161], which are both described in more detail in the next chapter. Research on MOEAs and multi-objective optimization has been very active during the last two decades. The research has concentrated on developing new MOEAs in order to find as good solution sets as possible. Several other topics in the research field are also the subject of much work. These include:

- How to define and measure the quality of a result.

- How to test performance (including test problem development).
- How to improve methods computationally.
- How to solve a problem when the number of objectives increases.
- How to adjust the control parameters of the methods.
- How to deal with computationally expensive objectives.

Differential Evolution (DE) is a relatively new EA and has been gaining in popularity in recent years because of its simplicity and good observed performance. Several extensions of DE for multi-objective optimization have already been proposed. Older approaches just convert a MOOP to a single-objective problem and use DE to solve the single-objective problem [5, 16, 143], whereas more recent and advanced approaches mainly use the concept of Pareto-dominance [106].

The aim of this thesis is to develop and study DE for multi-objective optimization. The topic seems promising since MOEAs have become important and DE appears a promising EA for use in multi-objective optimization. The work focuses on diversity preservation among solutions.

The starting point of the work was an idea how DE could easily be extended to handle multiple constraints and objectives merely by modifying the selection rule. The idea had been proposed in [90] but its properties had not been studied. The approach was, therefore, implemented and tested by the author of this thesis. When the results of this initial investigation were published in [81], the method was named Generalized DE (GDE). The name GDE was chosen because the method can handle multiple objectives and constraints, and the method is identical to basic DE in the case of an unconstrained single-objective problem. These properties were retained during different development phases of the method. Already during preliminary studies, it was found that for good performance the method needed different control parameter values than usually used with single-objective DE, and the diversity of the obtained solutions could have been better [81]. Therefore, the work continued by studying the effect of control parameter values and by developing the diversity preservation part of the method in the publications included in the thesis. Different GDE versions differ mainly in their ability to maintain the diversity of the solutions. The last GDE version, GDE3, has been noted to be effective and efficient for solving constrained multi-objective problems. It has also performed well in comparison with several other MOEAs [136].

The thesis is divided into five chapters. Chapter 2 contains literature study and provides background information about EAs, multi-objective optimization, MOEAs, and DE. DE extensions for multi-objective optimization are also covered. Chapter 3 describes the work presented in the publications of this thesis. Different development phases of GDE are described with experimental illustrations. General conclusions and discussion of the thesis work are presented in Chapter 4. Chapter 5 contains corrections to errors observed in the publications and additional information regarding the publications.

Summary of the Publications

The author of this thesis has made the main contribution to all the articles included in the thesis. The co-authors have been involved in a supervisory role in this thesis work. In the following, the individual papers are briefly summarized.

In *Publication I*, the first version of GDE was evaluated against several multi-objective evolutionary algorithms using a set of common bi-objective test problems. Based on the results, the performance of the first version of GDE was found very comparable to the performance of other multi-objective evolutionary algorithms. However, it was noticed that the control parameter values controlling mutation and crossover should be drawn from a rather narrow range for some problems. Moreover, the diversity obtained could have been better. The author of this thesis implemented the optimization method, performed the tests, and wrote the article.

In *Publication II*, a control parameter value study for the first version of GDE was performed empirically using a set of common bi-objective test problems and performance metrics. A non-linear relationship between the main control parameter (CR and F) values was observed according to a theorem in [148] for single-objective DE about the relationships between the control parameter values and the evolution of the population standard deviation. Based on this relation, a recommendation for control parameter value selection was given. It was also noticed that good results were obtained mainly with small control parameter values. The author of this thesis selected the test problems, performed the tests, and wrote the article.

In *Publication III*, the second version of GDE was proposed. Unlike the first version, the second version incorporated a diversity preservation technique. The performance of the proposed method was evaluated experimentally, and an improvement in the extent and distribution of solutions was observed. However, the second version was noted to be rather sensitive to the selection of control parameter values. The author of this thesis developed and implemented the method, performed the tests, and wrote the article.

In *Publication IV*, the third version of GDE was proposed. This version improved the previous version by using non-dominated sorting and a better diversity preservation technique. The method was observed to be more robust to the selection of control parameter values compared to the earlier GDE versions and provided at least comparable results compared to NSGA-II. The author of this thesis developed and implemented the method, performed the tests, and wrote the article.

In *Publication V*, the diversity preservation technique of the third version of GDE has been described in detail. The diversity preservation technique is an improved version compared to the one used in NSGA-II. It was noted that the diversity preservation technique does not provide good diversity when the number of objectives is three instead of two. One of the main contributions of the article was an explanation why the crowding distance metric used in NSGA-II and GDE versions in *Publication III* and *Publication IV* fails to approximate crowding of solutions when the number of objectives is more than two. The author of this thesis developed and implemented the diversity preservation method, performed the tests, made the observations, and wrote the article.

In *Publication VI*, a similar control parameter value study as in *Publication II* was performed for the third version of GDE. The set of problems was supplemented with

a set of non-separable problems. A similar observation as in *Publication II* about the non-linear relationship between the control parameter values was made. Compared to the results in *Publication II*, the third version of GDE was observed to be more robust to the selection of control parameter values and provided better results based on the performance metrics used. The author of this thesis selected the test problems, performed the tests, and wrote the article.

In *Publication VII*, a diversity preservation technique for problems with more than two objectives was proposed because the need for such a method was noticed in *Publication V*. The technique is a further development of the technique used in *Publication V*. The crowding distance metric is no longer used. Instead, a crowding estimation technique based on distance to the nearest neighbors is implemented. The efficiency of the method is based on an efficient nearest neighbors search algorithm and the use of a priority queue. The proposed technique was observed to provide good diversity and to be relatively fast. The author of this thesis developed and implemented the diversity preservation method, performed the tests, and wrote the article.

Chapter II

Background and Related Studies

"The year 1975 was a particularly good one for genetic algorithms."
– David E. Goldberg

This chapter provides background information for this thesis. First, evolutionary algorithms are described, and then multi-objective optimization and constraint handling are discussed. The chapter concludes with a description of Differential Evolution and its modifications for multi-objective optimization and constraint handling.

2.1 Evolutionary Algorithms

Evolutionary algorithm (EA) [6] is a generic term for a set of *population based and stochastic* optimization methods inspired by the theory of evolution by natural selection. EAs consist of such mainstream methods as Evolution Strategies (ESs) [12], Genetic Algorithms (GAs) [55], and Genetic Programming (GP) [8]. The first methods that can be considered as EAs were proposed already in the 1950's [55, pp. 89–104] but a more important date was the year 1975 when Holland published his *Adaptation in Natural and Artificial Systems* book [61] and De Jong completed his thesis [32] about the topic [55, p. 106]. During recent decades, EAs have become popular and important. Prominent and relatively recent EAs are Ant Colony Optimization (ACO) [42], Particle Swarm Optimization (PSO) [21], and Differential Evolution (DE) [119].

A common structure of EAs can be given. A general EA consists of the following steps:

1. Initialize a population
2. While a termination criterion is not met
3. Select members for reproduction
4. Reproduce new member candidates
5. Select members for the next generation

In the first step, the population is initialized. When there is no prior information, the decision variable values of the population are usually selected from some priorly defined value range according to uniform random distribution. If some prior knowledge exists

about the location of the optimum, then initialization can be done, *e.g.*, based on Gaussian distribution centered at the presumed location of the optimum.

After the initialization, the population is evolved iteratively; each iteration is called a *generation*. The usual termination criterion for the iterative process is some predefined upper limit for the number of generations. The termination criterion may also be based on the development of the population during previous generations or some knowledge about the value of the optimum [44, pp. 23–24].

The first step in the iteration loop is the *selection* of members of the population (often called *parents*) for reproduction. The selection is done based on the quality, *fitness*, of each member. Thus, there must be a way to evaluate the fitness (or cost) value of a particular decision variable value combination. Usually better members according to the fitness value are preferred in the selection but some EAs (*e.g.*, DE) select all the members for reproduction.

The reproduction is often divided into *recombination* (often called *crossover*) and *mutation* operations. Recombination combines two or more selected members/parents to reproduce one or more *children*. The idea behind recombination is that combining parents might produce a child that combines good features from the parents and is better than any of the parents. This idea is often referred to as the building block hypothesis [55, pp. 41–45]. Of course (like in nature) the recombination might produce a child which is worse than the parents. Mutation means a slight alternation of a single member and its purpose is to create variability in the population and to reduce the risk of stagnation.

The final step in the iteration loop is to select members for the next generation from the members of the previous generation and the reproduced members. As earlier, the selection usually prefers better members. If the selection guarantees that the best member found so far is selected for the next generation, then the selection is *elitist* and the EA is an elitist optimization method. The degree to which the selection prefers better members is known as the *selection pressure*, and although the selection should prefer better members, the selection pressure should not be too high as it might lead to a lose of diversity in the population and to premature convergence [33, p. 104]. Premature convergence means that the population has converged to a sub-optimal solution and the search no longer advances. Diversity is lost since population members are close to each other in the decision variable space, *i.e.*, the search space.

Balancing the exploitation effect of the selection and the exploration effect caused by the reproduction is one of the key issues in an EA and determines how greedy, fast, or reliable the EA is. Very greedy algorithms are typically fast for easy problems but lead to premature convergence with more demanding problems. On the other hand, methods that are successful with harder problems typically have a slower convergence rate.

Since EAs are stochastic methods with several simultaneously evolving solution candidates, their mathematical analysis is difficult. However, theoretical analysis has been performed using the Markov chain theory and it has been proven that an EA fulfilling two conditions will converge to a global optimum [6, p. 129]. Prerequisite conditions are:

1. The method is elitist.

2. Any solution candidate in the search space can be created using reproduction operations.

A method fulfilling these conditions will definitely converge to a global optimum if there is an infinite amount of time. It is questionable how much significance this has in practice because time is limited in the real world.

Due to the difficulty of theoretical analysis, comparisons between methods are usually performed empirically. Since EAs are stochastic methods, the results of an EA are with high probability different for different repetitions of the search. Therefore, experiments used for performance evaluation usually have several repetitions.

Experimental evaluation is not without its own difficulties because the result of the comparison depends on at least the following factors:

1. The problems used in the testing.
2. The control parameter values used for the optimization methods.
3. The evaluation criteria used to evaluate the quality of the obtained results.

With suitable selection of these factors, substantially different results can be obtained for the same set of optimization methods, as implied by the No Free Lunch (NFL) theorem. The NFL theorem states that no single optimization method can be the best for all problems, and all optimization methods are equally good with respect to the set of all problems [144]. Therefore, experimental results should be examined in the context of the target class of problems to be solved and cannot be generalized to the set of all problems.

Many classical optimization methods rely on assumptions about the mathematical properties of the fitness function, such as convexity, linearity, differentiability, continuity, and modality. However, these assumptions do not hold for many practical problems. EAs are free from most assumptions about the fitness function, thus the fitness function can be considered as a "black box" without information about the properties of the fitness function. This is probably the main reason why EAs have become popular during the last decades.

2.2 Multi-Objective Optimization with Constraints

2.2.1 Basic Concepts

Multi-objective optimization means optimization of more than one objective or goal at the same time. Many practical problems have multiple objectives and several factors create constraints to problems. For example, mechanical design problems may have several objectives such as obtained performance and manufacturing costs, and available resources may be limited. Constraints can be divided into boundary constraints and constraint functions. Boundary constraints are used when the value of a decision variable is limited to some range. Constraint functions represent more complicated constraints, which are expressed as functions on one side of an inequality equation.

Mathematically, a constrained multi-objective optimization problem (MOOP) can be presented in the form [108, p. 37]:

$$\begin{array}{ll} \text{minimize} & \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\} \\ \text{subject to} & g_1(\boldsymbol{x}) \leq 0 \\ & g_2(\boldsymbol{x}) \leq 0 \\ & \quad \vdots \\ & g_K(\boldsymbol{x}) \leq 0 \\ & \boldsymbol{x} \in \mathbf{R}^D. \end{array} \tag{2.1}$$

Thus, there are M objective functions f_m to be minimized, K constraint functions g_k limiting the search space, and D decision variables. Decision variables, which define values of objectives and constraints, form together a decision vector $\boldsymbol{x}$. The goal of multi-objective optimization is to find a decision vector $\boldsymbol{x}$ which minimizes the objectives without violating any of the constraints. Maximization problems can be converted to minimization problems by multiplying the objective function by -1 because maximization of $f_m(\boldsymbol{x})$ is equivalent to minimization of $-f_m(\boldsymbol{x})$. All constraints can be converted into the form $g_k(\boldsymbol{x}) \leq 0$ in the following way: $g_k(\boldsymbol{x}) \geq 0 \Leftrightarrow -g_k(\boldsymbol{x}) \leq 0$, $g_k(\boldsymbol{x}) = 0 \Leftrightarrow g_k(\boldsymbol{x}) \leq 0 \wedge -g_k(\boldsymbol{x}) \leq 0$. Boundary constraints can also be presented in the form of constraint functions, *e.g.*, $x_i^{(LO)} \leq x_i \leq x_i^{(HI)} \Leftrightarrow x_i^{(LO)} - x_i \leq 0 \wedge x_i - x_i^{(HI)} \leq 0$. Thereby the formulation in Equation 2.1 is without loss of generality. However, because boundary constraints are usually handled separately, the general definition is given in the form:

$$\begin{array}{ll} \text{Minimize} & \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\}, \\ \text{subject to} & g_1(\boldsymbol{x}) \leq 0 \\ & g_2(\boldsymbol{x}) \leq 0 \\ & \vdots \\ & g_K(\boldsymbol{x}) \leq 0 \\ & x_1^{(LO)} \leq x_1 \leq x_1^{(HI)} \\ & x_2^{(LO)} \leq x_2 \leq x_2^{(HI)} \\ & \vdots \\ & x_D^{(LO)} \leq x_D \leq x_D^{(HI)} \ . \end{array} \tag{2.2}$$

It should be noted that a problem does not necessarily contain all the constraints in the above definition, *i.e.*, some of the decision variables might be unbounded.

Usually, the objectives conflict and it is not possible to find a single solution that would be optimal for all the objectives. Therefore, the task becomes a search for a set of solutions which represent the best possible compromises between different objectives. For such solutions it holds that none of the objectives can be improved without impairing at least one other objective. This definition was given by Edgeworth in 1881 but nowadays it is commonly known as the definition of Pareto-optimality after another nineteen century scientist, Vilfredo Pareto, who developed the definition further [108, p. 11]. Finding Pareto-optimal solutions for a MOOP is sometimes called Pareto-optimization [77]. If there exist such decision vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that

$$\forall m \in \{1, 2, \ldots, M\} : f_m(\boldsymbol{x}_1) \leq f_m(\boldsymbol{x}_2) \ , \tag{2.3}$$

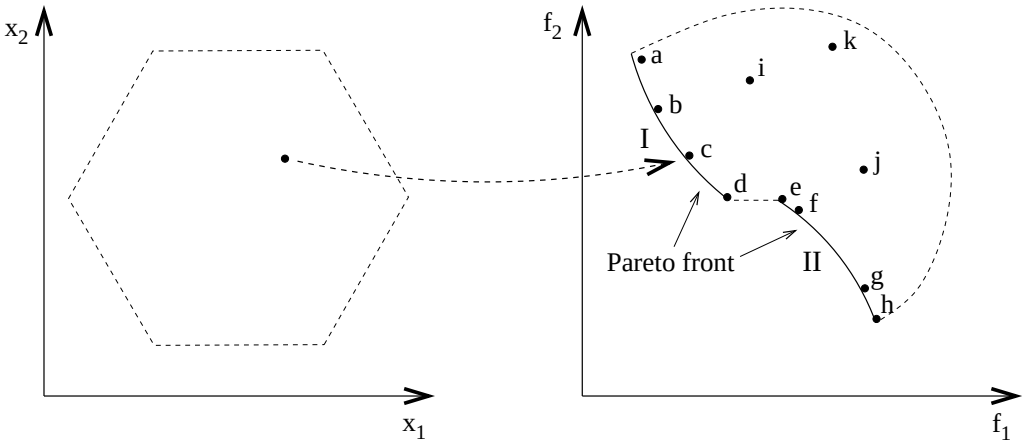


Figure 2.1: An example of Pareto-optimal front and solutions candidates for a bi-objective MOOP with two decision variables. The feasible area is bounded with a dashed line.

then $\boldsymbol{x}_1$ weakly (Pareto-)dominates $\boldsymbol{x}_2$ and this is expressed as $\boldsymbol{x}_1 \preceq \boldsymbol{x}_2$. If

$$\boldsymbol{x}_1 \preceq \boldsymbol{x}_2 \quad \wedge \quad \exists m \in \{1, 2, \ldots, M\} : f_m(\boldsymbol{x}_1) < f_m(\boldsymbol{x}_2) \ , \tag{2.4}$$

then $\boldsymbol{x}_1$ (Pareto-)dominates $\boldsymbol{x}_2$ and this is expressed as $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$ [164]. No solution exists which dominates a Pareto-optimal solution. Pareto-optimal solutions form a Pareto-optimal set in the decision variable space and a Pareto-optimal front in the objective space [26, pp. 11–12]. This is illustrated in Figure 2.1, which shows the decision variable space and objective space for a MOOP with two decision variables and two objectives. The region of the spaces which satisfies the constraints is bounded with a dashed line. Points inside the region are feasible and points outside the region are infeasible.

In Figure 2.1, a point drawn in the decision variable space will correspond to the point c in the objective space. This is one solution candidate for the MOOP. Figure 2.1 also shows other solution candidates $a, b, d-k$ in the objective space. Some of them are closer to the Pareto-optimal front than others, and the Pareto-optimal front in this case consists of two disconnected curves (*i.e.*, the Pareto-optimal front is not necessarily continuous). Part I of the Pareto-optimal front has a convex shape and part II is concave. Therefore, the whole Pareto-optimal front is considered to be non-convex. Solution candidates $a - h$ do not dominate each other and none of solution candidates $i - k$ dominate them. Therefore, solution candidates $a - h$ form a non-dominated set that is considered to be an approximation of the Pareto-optimal front for the MOOP. Solution candidates $i - k$ in Figure 2.1 are dominated by at least one solution from the set of the non-dominated solutions, and therefore solutions $i - k$ are discarded as solutions for the MOOP.

Solution candidates can be sorted based on dominance using non-dominated sorting [33, pp. 40–44]. When solution candidates $a - k$ are sorted, non-dominated solutions $a - h$ form the first non-dominated class, non-dominated solutions of the remaining solutions, *i.e.*, solutions $i - j$ form the second non-dominated class, and solution k forms alone the third non-dominated class. The first class is considered the best and the third class

is considered the worst, based on dominance. Solutions which belong to the same non-dominated class do not dominate each other.

A classical approach for multi-objective optimization is to convert a MOOP into a single-objective form by predefining weighting factors for different objectives, expressing the relative importance of each objective *a priori*. The weights then define what kind of compromise solution is sought by a decision-maker. The decision-maker is a person (or group of persons) who does the ultimate selection of the final solution among the set of non-dominated solutions that are equally good in the sense of Pareto-dominance [108, p. 14–15]. Other ways than weights also exist to express the preference of the decision-maker *a priori*, *e.g.*, ϵ-constraint and goal programming methods. Some of the most common methods are discussed in Section 2.2.4.

The decision-maker may not want or not be able to provide the relative importance of objectives beforehand. In such cases, the approach is to find a set of solution candidates and let the decision-maker pick a solution which provides a suitable compromise between the objectives. This can be viewed as *a posteriori* articulation of the preferences of the decision-maker concerning the relative importance of each objective. Besides *a priori* and *a posteriori* approaches, no-preference and interactive approaches also exist [108]. The no-preference approaches provide a solution without any preference information and the interactive approaches involve the decision-maker interacting/guiding the solution process.

Only the convergence aspect of multi-objective optimization has been considered in the above discussion, *i.e.*, the obtained solutions should be as close to the Pareto-optimal front as possible. However, *a posteriori* methods provide several solution candidates and besides convergence, good diversity is also desired [33, pp. 22–23]. Diversity is usually considered in the objective space (not the decision variable space). Good diversity means that the spread of extreme solutions is as high as possible and the relative distance between solutions is as equal as possible. For example, there is good diversity among non-dominated solutions $a - d$ in Figure 2.1 since they are uniformly distributed and cover well part I of the Pareto-optimal front. The distribution of solutions $e - h$ is less optimal since the solutions are not uniformly distributed although they cover the full extent of the Pareto-optimal front part II. Although the goal of good diversity is common in *a posteriori* multi-objective optimization, the goal has not been defined precisely and obtained diversity has been a subjective measure. Only recently has a formal definition for good diversity been proposed, in [79]. According to this definition, obtaining a good diversity for n solutions forming a non-dominated set P is equal to the optimization problem:

$$\begin{array}{ll} \text{Maximize} & \min_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \,, \\ \text{subject to} & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{P}, |\mathcal{P}| = n \,. \end{array} \tag{2.5}$$

Thus, the optimum for this problem is a set of n solutions for which the minimal distance between any two solutions in the objective space is maximal. The natural distance metric between two solutions is the Euclidean distance and different objective values should be normalized before distance calculations.

2.2.2 Constraint Handling

Most practical problems contain constraints that must be treated in the optimization process. In the following section, only very brief background information about constraint handling is given, since constraint handling has not been under development in this work, although it is a part of Generalized Differential Evolution. More comprehensive surveys about constraint handling techniques used with EAs are given in [22, 103].

Boundary Constraints

Decision variable values of a problem are usually constrained. Many optimization algorithms are capable of creating solution candidates outside the original initialization bounds of the decision variables. If a problem definition contains boundaries for decision variable values, then some technique must be used for handling boundary constraint violations. It is possible just to reject the violating solution candidate and create a new one, however, decision variable values can be corrected according to some rule to ensure they are inside boundaries as described in [119, pp. 202–206]. Three boundary constraint violation correction techniques are described here.

The first technique sets the value of the violating decision variable x_i to the corresponding boundary. This can be presented formally:

$$x_i = \begin{cases} x_i^{(LO)} & \text{if} \quad x_i < x_i^{(LO)} \\ x_i^{(HI)} & \text{if} \quad x_i > x_i^{(HI)} \end{cases} , \tag{2.6}$$

where $x_i^{(LO)}$ and $x_i^{(HI)}$ are corresponding lower and upper bounds, respectively. This rule is very simple to implement but reduces the diversity of the decision variables.

The second commonly used technique is to generate a new decision variable value between the decision variable boundary values:

$$x_i = x_i^{(LO)} + rand[0,1] \cdot \left(x_i^{(HI)} - x_i^{(LO)} \right) \quad \text{if} \quad x_i < x_i^{(LO)} \vee x_i > x_i^{(HI)} , \tag{2.7}$$

where $rand[0,1]$ denotes a random number generated from the uniform distribution $[0,1]$.

The third technique reflects any violation back into the feasible solution area by the same distance by which the boundary was violated.

$$x_i = \begin{cases} 2x_i^{(LO)} - x_i & \text{if} \quad x_i < x_i^{(LO)} \\ 2x_i^{(HI)} - x_i & \text{if} \quad x_i > x_i^{(HI)} \end{cases} . \tag{2.8}$$

If the amount of violation is larger than the distance between the upper and lower limits, then the corrected value is still outside the variable value range but the amount of violation is less than originally. The rule above can be repeated as many times as needed to make the variable value feasible.

The third technique does not need a random number generator as required by the second technique and provides better diversity in the population than the first technique. For this reason the technique has been used (if necessary) in all the experiments in this thesis.

Constraint Functions

More complicated constraints are presented in the form of functions on one side of the inequality and 0 on the other side [1]. Also in this case, an infeasible solution could simply be rejected and a new solution generated. However, the classical method to handle constraint functions has been an approach based on the use of a penalty function [112]. The idea is to penalize an infeasible solution by increasing the value of the corresponding objective function by the constraint violation. This can be presented formally as follows:

$$F_m(\boldsymbol{x}) = f_m(\boldsymbol{x}) + \sum_{k=1}^{K} w_k \max(g_k(\boldsymbol{x}), 0) \ . \tag{2.9}$$

Thus, the constraint function g_k increases the value of F_m if $g_k(\boldsymbol{x}) > 0$. The value of F_m is used instead of f_m when different solutions are compared. Each constraint function has a penalty parameter/weight w_k, which defines the importance of the corresponding constraint. The requirement for setting penalty parameter values is the biggest drawback of the penalty function approach, since determination of suitable penalty parameter values is not trivial and different penalty parameter values lead to different results. Therefore, alternative approaches (*e.g.*, dynamic and adaptive penalties) have been developed to overcome the problem [22,112].

Parameter free approaches also exist and have been becoming popular lately since they do not have the problem of choosing or adjusting appropriate parameter values. These techniques are often based on the following simple principles when comparing two solutions at a time [33, pp. 131–132]:

1. A feasible solution is better than an infeasible solution.
2. Among two feasible solutions, the better one has a better objective value.
3. Among two infeasible solutions, the better one violates the constraints less.

A couple of variations exist for the third principle. One popular approach is to add the constraint violations and compare the sums, as used by Deb [33, p. 131]. Another approach is to use the dominance-relation in the space of constraint violations [92]: If the constraint violations of solution $\boldsymbol{x}$ dominate the constraint violations of solution $\boldsymbol{y}$ [2], then $\boldsymbol{x}$ is considered to be better (see Figure 2.2).

In the first approach, where constraint violations are summed, constraints having a larger magnitude will direct the search since their violations have more effect on the total constraint violation. This approach also permits worsening of individual constraint function values because sums of the constraint violations are compared instead of individual constraint violations.

The second approach does not permit worsening of individual constraint function values and is therefore more strict. It can lead to a situation where neither of the solutions can

[1] Boundary constraints can be handled as constraint functions, as noted in the previous section, but this is not rational since boundary constraints can be handled more easily than constraint functions.

[2] We define that $\boldsymbol{x}$ dominates $\boldsymbol{y}$ with respect to constraints iff $\forall k : g'_k(\boldsymbol{x}) \leq g'_k(\boldsymbol{y}) \wedge \exists k : g'_k(\boldsymbol{x}) < g'_k(\boldsymbol{y}), \quad g'_k(\boldsymbol{z}) = \max(g_k(\boldsymbol{z}), 0)$

be judged to be better. This happens if both solutions are infeasible and the constraint violations do not dominate each other. This kind of situation can be solved using additional rules (*e.g.*, preferring the old solution if the new one is not better). This second constraint handling approach has been noted to give very competitive results with a set of benchmark problems having dozens of constraints [83, 103].

The two parameter free methods described above are the simplest, but still effective approaches to constraint handling.

2.2.3 Constraint-Domination Relation

In addition to objective values, the dominance relationship can be extended to take into consideration constraint values. *Constraint-domination* $\prec_c$ is defined here such that $\boldsymbol{x}_1$ constraint-dominates $\boldsymbol{x}_2$, *i.e.*, $\boldsymbol{x}_1 \prec_c \boldsymbol{x}_2$ iff any of the following conditions is true [90]:

1. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are infeasible and $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$ in respect to constraint violations.
2. $\boldsymbol{x}_1$ is feasible and $\boldsymbol{x}_2$ is not.
3. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are feasible and $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$ in the objective function space.

This approach is a special case of a unified formulation of goals and priorities proposed by Fonseca and Fleming [49]. A slightly different constraint-domination definition is given in [33]. It differs in such a way that two infeasible solutions are compared based on the sum of constraint violations instead of dominance in the constraint function violation space.

The definition for weak constraint-domination $\preceq_c$ is analogous when the dominance relation is changed to weak dominance in the above definition. The weak constraint-domination relation can be formally defined as:

$$\boldsymbol{x}_1 \preceq_c \boldsymbol{x}_2 \quad \text{iff} \quad \left\{ \begin{array}{l} \left\{ \begin{array}{l} \exists k \in \{1,\ldots,K\} : g_k(\boldsymbol{x}_1) > 0 \\ \wedge \\ \forall k \in \{1,\ldots,K\} : g'_k(\boldsymbol{x}_1) \leq g'_k(\boldsymbol{x}_2) \end{array} \right. \\ \vee \\ \left\{ \begin{array}{l} \forall k \in \{1,\ldots,K\} : g_k(\boldsymbol{x}_1) \leq 0 \\ \wedge \\ \exists k \in \{1,\ldots,K\} : g_k(\boldsymbol{x}_2) > 0 \end{array} \right. \\ \vee \\ \left\{ \begin{array}{l} \forall k \in \{1,\ldots,K\} : g_k(\boldsymbol{x}_1) \leq 0 \wedge g_k(\boldsymbol{x}_2) \leq 0 \\ \wedge \\ \forall m \in \{1,\ldots,M\} : f_m(\boldsymbol{x}_1) \leq f_m(\boldsymbol{x}_2) \end{array} \right. \end{array} \right. , \tag{2.10}$$

where $g'_k(\boldsymbol{x}_i) = \max(g_k(\boldsymbol{x}_i), 0)$ represents a constraint violation of $\boldsymbol{x}_i$ with respect to the kth constraint. It is good to note that in the case of no constraints and a single objective, $\boldsymbol{x}_1 \preceq_c \boldsymbol{x}_2$ iff $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2)$.

An example of constraint-domination is given in Figure 2.2, where the constraint violation and objective spaces are shown for a set of solution candidates. Solution candidates $a-c$ violate neither of the constraints. Thus these solutions constraint-dominate infeasible

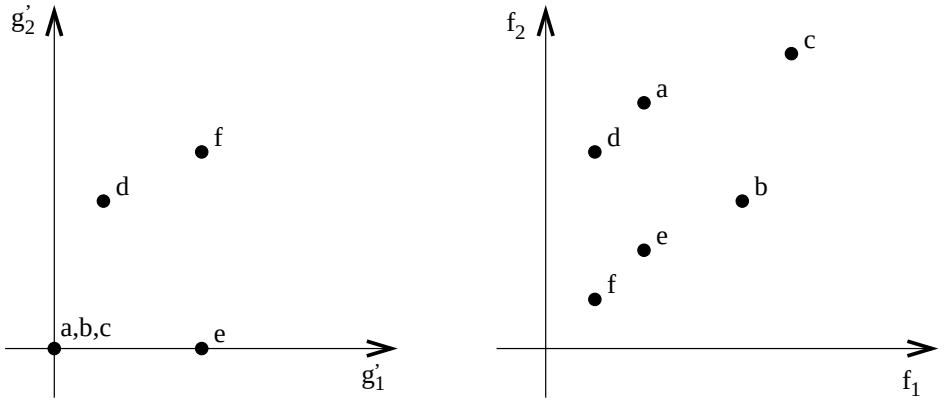


Figure 2.2: Example of the constraint violation and objective space for a set of solution candidates.

solution candidates $d - f$ although solution candidates e and f are better in the objective space. In the same way, $e \prec_c f$ because both are infeasible and comparison is done in the constraint violation space, where e dominates f, and $a \prec_c c$ because both are feasible and a dominates c in the objective space.

2.2.4 Classical Methods for Multi-Objective Optimization

Although this thesis concerns evolutionary multi-objective optimization and is not directly concerned with non-evolutionary methods, a brief overview of the most common non-evolutionary multi-objective optimization methods is given below.

Multi-objective optimization is traditionally called Multiple Criteria Decision-Making (MCDM) [108, p. xiii]. The general goal in MCDM is to help the decision-maker to find a solution or several solutions for a MOOP.

In the following, the methods are listed according to [33, pp. 49–79]. A more comprehensive study can be found in [108].

Weighted Sum Method

The weighted sum method is the simplest and probably the most commonly used traditional multi-objective optimization method. The idea is to calculate a scalarized objective function that is a weighted sum of individual objective functions. More formally, the scalarized objective value is calculated as:

$$F(\boldsymbol{x}) = \sum_{m=1}^{M} w_m f_m(\boldsymbol{x}) \ . \tag{2.11}$$

Thus, there is a non-negative weight w_m connected to each objective defining the relative importance of the objective.

The weighted sum approach is simple to implement and calculate even for a large number of objectives. Furthermore, it guarantees that the optimum of the scalarized objective function is a Pareto-optimal solution. However, if a MOOP is non-convex, not all the Pareto-optimal solutions can be found using the weighted sum method.

ϵ-Constraint Method

In the ϵ-constraint method, one objective is optimized and rest of the objectives are regarded as constraints. The modified problem is in the form:

$$\begin{array}{ll} \text{Minimize} & f_\mu(\boldsymbol{x}) \\ \text{subject to} & f_m \leq \epsilon_m, \quad m = 1, \ldots, M \wedge m \neq \mu \ . \end{array} \tag{2.12}$$

User-defined constraint values ϵ_m limit the value of all but one objective. The ϵ-constraint method can be used to identify all the Pareto-optimal solutions for any kind of problem (also non-convex problems).

Weighted Metric Methods

Weighted metric methods are slightly more complicated versions of the weighted sum method. Different objectives are combined together into a single objective function which has the form:

$$l_p(\boldsymbol{x}) = \left(\sum_{m=1}^{M} w_m \left| f_m(\boldsymbol{x}) - z_m^* \right|^p \right)^{1/p} , \tag{2.13}$$

where w_m is a weight, $p \in [1, \infty)$, and z_m^* is the mth element of an ideal solution [3]. When $p = 1$, the method is equivalent to the weighted sum method and, therefore, all the Pareto-optimal solutions cannot be found for non-convex problems. When the value of p is increased, a greater quantity of Pareto-optimal solutions can be found, but at the same time, the weighted metric becomes non-differentiable and many optimization methods are not applicable.

When $p = \infty$, the objective function has the form:

$$l_\infty(\boldsymbol{x}) = \max_{m=1,\ldots,M} \left[w_m \left| f_m(\boldsymbol{x}) - z_m^* \right| \right] \ . \tag{2.14}$$

This problem formulation has its own name – the weighted Tchebycheff. All the Pareto-optimal solutions can be found with this formulation if a utopian objective vector $\boldsymbol{z}^{**}$ [4] is used instead of $\boldsymbol{z}^*$ [108, p. 155].

[3] The ideal solution $\boldsymbol{z}^*$ is defined such that the mth element of $\boldsymbol{z}^*$ is the optimum for f_m, thus $z_m^* = f_m^*$.

[4] If minimization of objectives is assumed, $\boldsymbol{z}_m^{**} = \boldsymbol{z}_m^* - \epsilon_m$ with $\epsilon_m > 0$ for all $m = 1, 2, \ldots, M$, thus, $\boldsymbol{z}^{**}$ has slightly smaller objective values than $\boldsymbol{z}^*$ has.

Value Function Method

The value function method assumes that the decision-maker is able to express her/his preference as a mathematical function U, which scalarizes multiple objectives into a single value. This value function is then maximized:

$$\text{Maximize} \quad U\left(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\right) \ . \tag{2.15}$$

In a sense, the value function method is optimal if a decision-maker is able to define the value function. However, it is often impossible for a decision-maker to define her/his absolute preference beforehand. The value function must also fulfil certain properties so that its optimum is Pareto-optimal [108, p. 115]. Therefore the value function method is little used in practice and has more importance as a theoretical concept.

Goal Programming Methods

Goal programming methods try to satisfy goals set for objectives. For example, the decision-maker could demand that objectives are equal, less-than-or-equal, or greater-than-or-equal to certain target values. The decision-maker could also set value ranges for objective values. The optimization method then tries to satisfy these goals as well as possible.

Different ways to express preference exist. One approach is lexicographic ordering. In this approach the decision-maker gives a priority order for the goals. Goals with higher priority will be satisfied first and after that goals with lower priority. Lexicographic ordering has a connection to constrained optimization approaches in which constraints are satisfied first and then objectives are optimized.

Interactive Methods

Interactive methods differ from the above-mentioned methods since they involve the decision-maker during the optimization process. A MOOP is initially solved and then the opinion of the decision-maker about the preferences is requested before the optimization process is continued. The interaction with the decision-maker is continued until a solution satisfying the decision-maker has been found.

Several interactive approaches are described in [108]. One of these is the Non-differentiable Interactive Multi-objective BUndle-based optimization System (NIMBUS), which also has an online version available free for academic use [109–111].

Review of Classical Methods

Many of classical methods have the advantage that their working principle is well understood and Pareto-optimality of found solutions can be guaranteed. The above-mentioned classical methods (with exception of the interactive methods) find a single solution for a given MOOP since the methods convert a MOOP to a single-objective form and solve it. It should be noted that no general single-objective optimization method exists that

would return the solution of any problem in finite time. Therefore, finding a Pareto-optimal solution in finite time for a MOOP cannot be guaranteed in general. As noted above, some classical approaches cannot find certain solutions for a non-convex MOOP. In addition, all the methods discussed above demand some problem knowledge, *e.g.*, for setting suitable weight or target values.

In *a priori* optimization, the assumption is that the decision-maker is able to express her/his preference among different objectives beforehand. As has been already noted in Section 2.2.1, giving the absolute preference of objectives beforehand is often difficult. Finding solutions in some specific region of the Pareto-optimal front by adjusting method parameters (*i.e.*, weight, priority, or target) values might also be difficult. Therefore, without an absolute preference of the decision-maker, an *a posteriori* approach is argued to be more preferable [33, p. 77].

The above-mentioned classical methods can be used to generated a set of different compromise solutions for a given MOOP *a posteriori*. However, this would need several simulation runs of an optimization method since one simulation run typically provides only a single solution. For example, using the method with weighted Tchebycheff, different weights need to be used to obtain different solutions and the optimization process needs to be repeated in order to have different compromise solutions. Moreover, finding uniformly distributed solutions along the Pareto-optimal front is often difficult since uniformly distributed method parameter values do not guarantee uniformly distributed solutions in the objective space.

In the light of these issues, multi-objective evolutionary algorithms (MOEAs) in *a posteriori* multi-objective optimization have gained popularity during last decades since several MOEAs are able to produce a relatively well distributed set of non-dominated solutions approximating the Pareto-optimal front in a single simulation run. Furthermore, MOEAs do not commonly require any additional problem parameters and they are able to provide good results (*i.e.*, well distributed and converged set of solutions) for difficult MOOPs having non-convexity, non-linearity, discontinuity, and non-differentiability.

2.2.5 Multi-Objective Evolutionary Algorithms

Multi-objective evolutionary algorithms (MOEAs) are EAs designed to solve multi-objective problems. Originally, EAs were used in connection with scalarization approaches, such as mentioned in Section 2.2.4, *i.e.*, MOOPs were scalarized and then solved using a single-objective EA. Later, *a posteriori* EA approaches started to emerge and gain popularity since EA can easily provide several solution candidates, a feature that is desirable in *a posteriori* multi-objective optimization. In general, MOEAs are considered as *a posteriori* methods, *i.e.*, their goal is to find a limited number of well converged and distributed solutions approximating the Pareto-optimal front [25, pp. 3–4].

Currently thousands of references about MOEAs exist in the academic literature, including hundreds of doctoral theses [23]. Prominent doctoral theses about MOEAs were completed by Fonseca in 1995 [46], Zitzler in 1999 [156], and Van Veldhuizen 1999 [141]. Several books about MOEAs are also available. The most notable books are written by Deb [33] and Coello Coello *et al.* [25, 26]. In the following, the development history of MOEAs is briefly described.

One of the first MOEAs was the Vector Evaluated GA (VEGA) by Schaffer in 1984 [131]. VEGA is not based on the Pareto-dominance concept but dividing the population to as many subpopulations as there are objectives and doing the selection according to a single objective in each subpopulation. In reproduction, individuals of different subpopulations are allowed to mate with each other. No diversity preservation technique is included. VEGA produces solutions which are good according to a single objective but compromise solutions are typically missing.

After VEGA, the next notable MOEAs were proposed in the 1990's. These methods are based on the non-dominance concept in fitness evaluation and the use of some kind of mechanism to preserve diversity of solutions in the objective space. Notable suggestions were the Multiple Objective GA (MOGA) [47] by Fonseca and Fleming in 1993, the Non-Dominated Sorting GA (NSGA) by Srinivas and Deb in 1994 [132], the Niched-Pareto GA (NPGA) by Horn *et al.* in 1994 [62], and the Strength Pareto EA (SPEA) by Zitzler and Thiele in 1998 [162, 163]. All the above mentioned MOEAs except SPEA have a common property, namely that they are all non-elitist [33]. This means that they do not necessarily maintain the best found solution candidate during the evolutionary process. Non-elitist MOEAs are also sometimes referred to as first generation MOEAs.

Following the introduction of non-elitist MOEAs, elitist (second generation) MOEAs started to emerge. These methods include the above-mentioned SPEA and its improved version, SPEA2, proposed by Zitzler *et al.* in 2002 [161]. Other prominent elitist methods are the elitist Non-Dominated Sorting GA (NSGA-II) by Deb *et al.* in 2000 [34, 35] and the Pareto-Archived Evolution Strategy (PAES) by Knowles and Corne in 2000 [78]. The last mentioned researchers have also been involved in developing two versions of the Pareto Envelope-based Selection Algorithm (PESA) in 2000 and 2001 [28, 30].

A few recent MOEAs are not based on concepts of non-dominance and diversity preservation in the same way as the methods above. Remarkable methods are the ϵ-dominance Multi-Objective Evolutionary Algorithm (ϵ-MOEA) proposed by Laumanns *et al.* in 2002 [98], the Indicator-Based Evolutionary Algorithm (IBEA) proposed by Zitzler and Künzli in 2004 [159], and Multiobjective EA based on decomposition (MOEA/D) proposed by Zhang and Li in 2006 [152].

In addition to proposing new techniques, further analysis on MOEAs has also been undertaken. Similarly to EAs, convergence of MOEAs fulfilling the two same conditions as EAs (*cf.* Section 2.1) has been proven [141, p. 2-22–2-25]. The elitism condition in the case of MOEAs means that the population can move only towards the Pareto-optimal front. Thus, the next population must be equal or better in the sense of Pareto-dominance compared to the previous generation. Convergence conditions have been further discussed in [127, 128]. In [98] it has been shown that convergence cannot been guaranteed with several MOEAs, such as PAES, SPEA, and NSGA-II, designed to maintain diversity of solutions since they can lose Pareto-optimal solutions during the search. Thus, their categorization as elitist MOEAs is somewhat misleading.

In [53], it has been shown that with certain problems, a population based MOEA performs significantly better than several single individual-based algorithms, including the ϵ-constraint method. This same conclusion had been derived empirically earlier in [67], where it was also noted that when the number of objectives increases, the performance of methods based on Pareto-dominance declines.

In the following, NSGA-II, SPEA2, ϵ-MOEA, IBEA, and MOEA/D are discussed in more detail. NSGA-II and SPEA2 have been selected for more detailed discussion since they have been the most popular, state-of-art MOEAs for a long time. For this reason they have been chosen for comparison in the attached publications of this thesis work. ϵ-MOEA, IBEA, and MOEA/D have been selected for more detailed discussion because they present a more recent state/generation of MOEAs.

Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

The elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) [34, 35] has been the most used multi-objective optimization method in the MOEA literature. The working principle of NSGA-II is as follows: At each generation, a GA is used to create a child population which has an equal size compared to the parent population. After each generation, the parent and child populations are combined together. If the population size is NP, then the combined population has size $2NP$. The combined population is sorted using non-dominated sorting and the best NP individuals are selected based on non-dominance rank. Thus, individuals from the best non-dominated class are selected first, then from the second best non-dominated class, and so on, until the number of selected individuals is NP. If the last non-dominated class of solutions to be selected is too big to fit completely in the set of NP individuals, then this non-dominated set is reduced based on a crowding estimation among the individuals of the class. The idea is to remove the most crowded individuals until the remaining individuals fit into the selected set of NP individuals.

Crowding estimation in NSGA-II is based on a distance metric called the *crowding distance*. The crowding distance for a member of a non-dominated set tries to approximate the perimeter of a cuboid formed by using the nearest neighbors of the member. The cuboid in the case of two objectives is illustrated in Figure 2.3. For a member of a non-dominated set, the crowding distance is calculated by finding the objective value difference between the two nearest solutions on either side of the member along each of the objectives (distances d_1^i and d_2^i in Figure 2.3). Then the differences are normalized by dividing them by the difference between the maximum and minimum values of the corresponding objectives. Finally, these normalized distances are summed, giving a crowding distance for the corresponding member. For those members which have a maximum or minimum value for any objective, the crowding distance is assigned to have an infinite value, *i.e.*, those members are considered as the least crowded. Finally, the members of the non-dominated set are sorted according to the crowding distances and a desired number of members having the smallest crowding distance values are removed. [33, p. 248]

The above described selection process of individuals for the next generation is illustrated in Figure 2.4. It should be noted that pruning based on diversity is done only among the members of the last non-dominated class of solutions that is selected for the next generation.

In [33, pp. 245–246], it is claimed that with early generations there exist several different non-dominated sets/classes and the diversity preservation has little effect on the selection process. When the population starts to converge to the Pareto-optimal front, the non-dominated sets become larger and eventually it is likely that the best non-dominated

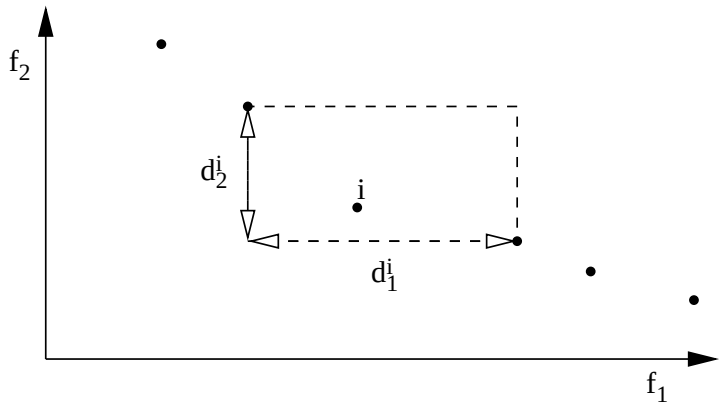


Figure 2.3: Example of the cuboid of a solution in the case of two objectives.

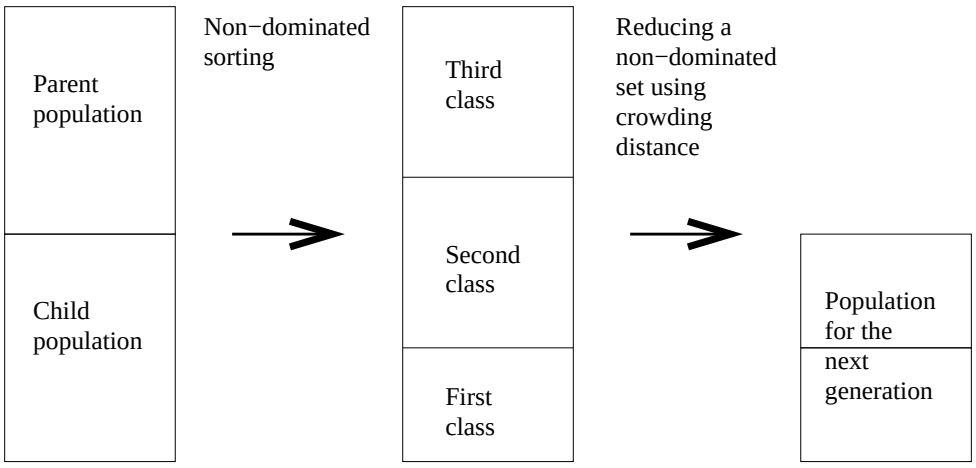


Figure 2.4: Selection of individuals for the next generation in NSGA-II.

set is larger than NP. Thus, only little diversity preservation is performed at the early generations but more during the late generations. This kind of strategy provides a way to balance between convergence and diversity, but unfortunately, it works only with two objectives, because the crowding distance metric used in NSGA-II does not estimate crowding well when the number of objectives is more than two as indicated later in this thesis work (*Publication V*). Even if there were a working diversity preservation technique, the balance between convergence and diversity changes when the number of objectives increases. When the number of objectives increases, the number of non-dominated individuals increases and diversity preservation becomes a dominating operation in the survival selection. In the light of this behavior, it becomes evident that, NSGA-II in its original form performs well only with problems having two objectives.

Improved Strength Pareto Evolutionary Algorithm (SPEA2)

The improved version of the Strength Pareto Evolutionary Algorithm (SPEA2) [161] is another commonly used MOEA. The fitness assignment of SPEA2 is based on calculating strength values for individuals. The strength value of an individual $\boldsymbol{x}$ measures how many individuals $\boldsymbol{x}$ dominates. A raw fitness value for an individual $\boldsymbol{y}$ is calculated as a sum of the strength values of those individuals that dominate $\boldsymbol{y}$. This raw fitness value is smaller for the better individuals.

Diversity preservation in SPEA2 is managed by calculating the distance of individuals to the kth nearest neighbor in the objective space. This distance value is transformed to a crowding measure; a small value means low crowding, and a large value means high crowding. The crowding measure is scaled between $[0, 1]$ and added to the fitness value of each individual. Therefore, individuals are primarily ranked using the raw fitness values and in the case of identical raw fitness values, the less crowded individual will be preferred.

In addition to the population, SPEA2 uses an extra archive for solutions. This archive has a fixed size and is mainly reserved for non-dominated solutions. However, if there are not enough non-dominated individuals, the space in the archive is filled based on the fitness value of the individuals. If the number of non-dominated individuals is larger than the archive size, then redundant individuals are removed based on the fitness values.

SPEA2 provides a well distributed set of solutions also when the number of objectives is more than two. However, when the number of objectives increases, the search will slow down because fewer solutions dominate each other.

ϵ-Dominance Multi-Objective Evolutionary Algorithm (ϵ-MOEA)

The ϵ-dominance Multi-Objective Evolutionary Algorithm (ϵ-MOEA) is based on the idea that non-dominated solutions whose objective values are close to each other (defined by ϵ) will ϵ-dominate each other [98]. A MOEA is used to find a set of solutions which do not ϵ-dominate each other. Use of ϵ-dominance will automatically maintain diversity since solutions cannot be too close to each other in the objective space.

Indicator-Based Evolutionary Algorithm (IBEA)

The Indicator-Based Evolutionary Algorithm (IBEA) [159] is based on the idea that the search is performed according to the preference of the decision-maker instead of fixing a technique for finding non-dominated solutions. Thus, IBEA searches for such a set of solutions which best satisfies the decision-maker. The preference is presented in the form of a binary indicator (which will be discussed further in Section 2.2.6). IBEA has been tested using the hypervolume and ϵ-indicator [164].

Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D)

Multiobjective EA based on decomposition (MOEA/D) [152] decomposes a MOOP into N different scalar optimization problems using the weighted Tchebycheff approach and solves these simultaneously using an EA. The EA is restricted to performing reproduction using only vectors close to each other in the weight vector space. Neighboring solutions are replaced with the new solution, if the new one has a better scalarized value.

MOEA/D represents a return to classical multi-objective optimization methods. However, MOEA/D is a recent method with good results and it might be usable also in cases when the number of objectives is large and the search based on Pareto-dominance stagnates.

2.2.6 Performance Evaluation in Multi-Objective Optimization

Since MOEAs are stochastic optimization methods, their performance is difficult to analyze theoretically. Therefore, the performance of a MOEA is usually analyzed empirically. This means performing repeated simulations with test problems and evaluating the performance in comparison with some other MOEA. However, getting conclusive results in this way is not trivial since the control parameter values used, the selected test problems, and the performance metrics affect the results, as pointed out in Section 2.1. Therefore, results based on experiments should be taken as indicative.

In the following, common test problems and performance metrics used with MOEAs are briefly discussed. A further important issue is how to visualize results for a MOOP.

Multi-Objective Test Problems

Common multi-objective test problems have been described in [25,33]. Many existing test problems have two objectives and many of them also have some simplifying characteristics (*e.g.*, several of them are separable [5]). Only recently have test problems appeared having more than two objectives and more complicated characteristics.

Some test problem sets are briefly described below. The first two test problem groups have been used in this thesis work and problem descriptions can be found in Appendix I. The last test problem group is more recent and has been mentioned here as supplementary information.

[5] An objective function is separable if its decision variables do not interact and the objective function can be presented as a sum of single-variable functions [123, p. 34]. The objective function can then be solved by optimizing each decision variable individually. A multi-objective problem is separable if all the objective functions are separable.

Zitzler-Deb-Thiele (ZDT) test problems [158] are probably the most used test problems in the MOEA literature. ZDT problems consist of six bi-objective problems, which have been designed to test the ability of a multi-objective optimization method to handle different types of difficulties with MOOPs. ZDT problems have the benefit that they are purely synthetic and their exact solutions are known. They are also easy to implement and they test several basic properties of MOEAs. However, they have several defects, which degrade their usability:

- The problems are bi-objective and not directly scalable.
- The first objective of the problems depends on a single decision variable and the second objective depends on the rest of the decision variables. The second objective is also a more complicated function. Therefore, the first objective is easier to optimize than the second, and this feature might lead to premature convergence along the first objective. (This issue is further discussed in the next chapter)
- For Pareto-optimal solutions, the first decision variable $x_1 \in [0, 1]$ and the rest of the decision variables have value 0, which is on the boundary of the decision variable value range. Thus, the boundary constraint violation correction method used has a great impact on the convergence to the Pareto-optimal front.
- For most of the problems, equally spaced values of the first decision variable correspond to equally spaced solutions in the objective space.
- New Pareto-optimal solutions can be generated using a linear combination of other Pareto-optimal solutions.

For the above-mentioned reasons, use of the original ZDT problems is becoming more rare. Some modifications of the ZDT problems have been proposed, *e.g.*, in [38].

Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems [40] have become popular in recent years. DTLZ problems are scalable to any number of objectives, although tri-objective versions are mostly used.

As with the previous problem set, DTLZ problems are synthetic and their solutions are known. Definitions for nine different DTLZ problems are given in [40]. Similar to ZDT problems, DTLZ problems address a variety of different problem characteristics. However, DTLZ problems have some limitations, *e.g.*, all the problems are separable [65].

The Walking Fish Group (WFG) test problem toolkit [65, 66] was proposed by Huband *et al.* in 2005 to increase the difficulty and variability of the test problems. As well as providing tools for test problem generation, a set of nine different test problems was proposed. These address several defects (*e.g.*, lack of non-separability) mentioned above with ZDT and DTLZ problems.

Multi-Objective Performance Measurements

Measuring the quality of a result for a MOOP is significantly more difficult than for a single-objective optimization problem. In the single-objective case, the difference between

obtained solutions can unambiguously be measured (and compared to the global optimum if known) and this measure can be used as a performance metric.

In the case of multi-objective optimization, no single, generally accepted, supreme performance measure exists. In the case of *a posteriori* multi-objective optimization, the goal is to find a set of non-dominated solutions which are as close to the Pareto-optimal front as possible and which cover the Pareto-optimal front as well as possible [33, pp. 22–23]. These two aspects are considered somewhat conflicting, and it has been argued that no single metric can measure the performance of an algorithm absolutely [33, p. 322].

The following aspects can be used to quantify the quality of a result for a MOOP:

1. Number of non-dominated solutions.
2. Closeness to the Pareto-optimal front.
3. Diversity, which includes:
   (a) Distribution of solutions.
   (b) Extent of solutions.

Measuring several different aspects gives more information about the characteristics of a solution set than a single metric value.

Several commonly used performance metrics can be found in [33, pp. 320–338] and [164]. In the following, several common performance metrics are briefly covered. Most of them have also been used in the publications included in this thesis. Metrics the for studies were selected according to prevailing insight about their suitability to measure certain characteristics.

Two commonly used convergence metrics are generational distance and error ratio [33, pp. 324–327], with the former being more often found. For both of these metrics, less is better and the optimal value is zero.

Generational distance *GD* measures the average distance of solutions to the Pareto-optimal front. Thus, from each solution, the shortest distance to the Pareto optimal front is measured and then the mean of these distances is calculated. This measure has been used in *Publication I*, *Publication II*, *Publication III*, and *Publication VI*.

Error ratio *ER* measures the fraction of solutions that are not Pareto-optimal. This measure was used in *Publication I* and *Publication III* but not later since it was found convergence can be measured better with generational distance.

The usability of these metrics is limited since the metrics can be used only when the global Pareto-optimal front is known.

Diversity is also a key consideration. The diversity of the obtained set of solutions has often been measured with spacing, spread, and maximum spread metrics [33, pp. 327–331]. For spacing and spread, less is better and the optimal value is zero. For maximum spread, the optimal value is one; if the full spread is not reached then the value is less.

Spacing S measures the standard deviation of the distances from each solution to the nearest solution in the obtained non-dominated set. This measure has been used in all the publications included in this thesis.

Spread Δ measures both uniformity of the obtained non-dominated set and the distance to the extreme values of the Pareto-optimal front. This measure was used in *Publication I* and *Publication III* but not later since it was found that the spacing metric measures distribution better.

Maximum spread, normalized version $\bar{D}$ measures the distance between extreme solutions in the obtained set relative to the distance between the extreme solutions of the Pareto-optimal front. This measure has been used in *Publication I*, *Publication III*, and *Publication VII* to measure the maximal extent of an approximation set.

As can be noted, some of the diversity metrics need knowledge about the Pareto-optimal front and thus the usability of such metrics is limited. Spacing does not require knowledge about the Pareto-optimal front and is usable in practice, although it has some weaknesses as noted in [40].

An inverted version of generalized distance [24] also exists and it can be used for measuring overall quality of the obtained set of solution, *i.e.*, how well solutions cover the actual Pareto-optimal front. Also for this metric, less is better and the optimal value is zero.

Inverted generational distance *IGD* measures the average distance of the Pareto-optimal front to the solutions. Thus, from each member of the Pareto-optimal front (approximation), the shortest distance to the set of solutions is measured and then the mean of these distances is calculated. This measure is a relatively recent approach and has been used in [84, 88].

The above mentioned performance metrics are called *unary metrics*. This means that the performance measurement needs a single set of solutions. There exist also *binary metrics*, which compare two sets of solutions. These do not need knowledge about the Pareto-optimal front and some of them are compliant with Pareto-dominance. This means that if one approximation set dominates another, it also has a better metric value [25, pp. 253]. It has been claimed that unary metrics are not Pareto-compliant and not capable of indicating whether one set of solutions is better than another set of solutions [164].

Common binary metrics are the ϵ-indicator [164], the set coverage metric [156], and the hypervolume [164] (also known as an S metric, Lebesgue measure, and $\mathcal{V}$ measure in *Publication I* and *Publication IV*). For all of these metrics, absolute optimal values cannot be given, rather the two sets of solutions have to be compared with each other in the two possible ways and then the metric values are compared.

ϵ-indicator $I_\epsilon(A, B)$ between two non-dominated sets A and B measures how much objective values of A must be changed in order to dominate B. If $I_\epsilon(A, B) < I_\epsilon(B, A)$, then A is better than B.

Set coverage metric $C(A, B)$ between two non-dominated sets A and B measures the fraction of members of B that are dominated by members of A. If $C(A, B) > C(B, A)$, then A is better than B. This measure has been used in *Publication I* and *Publication IV*.

Hypervolume $HV(A, B)$ between two non-dominated sets of solutions A and B calculates the hypervolume of the objective space which is dominated by A but not B. If $HV(A, B) > HV(B, A)$, then A is better than B. This measure has been used in *Publication I* and *Publication IV*. The unary version of the metric has been used in *Publication VII*.

The hypervolume can be calculated also for a single set of non-dominated points. Then hypervolume measures the space dominated by the set of points and limited to a reference point. It has been proven that for a MOOP, the set of solutions having maximal hypervolume value is the set of all Pareto-optimal solutions [45].

As can be noted earlier, the binary metrics above provide only a single value after comparing two solution sets. Thus, convergence and diversity are presented with a single value and it is difficult to make conclusions about these two aspects separately. Moreover, it has been reported that the hypervolume (as well as any other total order metric) is biased towards certain portions of the objective space [157]. Thus, maximizing the hypervolume value does not always lead to a uniformly distributed set of solutions with a fixed number of solutions. One drawback of binary metrics is that they are computationally expensive. The computational cost of hypervolume calculation in particular has limited its usability in practice, for which reason, several studies (*e.g.*, [13, 51]) have considered ways to accelerate the hypervolume calculation.

A further way to evaluate the performance of MOEAs is the use of attainment surfaces [48, 50]. An attainment surface for a set of obtained non-dominated solutions marks the region of the objective space that is dominated by the solution set. It should be noted that the attainment surface corresponds exactly to the surface used for hypervolume calculation. The attainment surfaces can be used to visualize the results of repeated simulation runs. The attainment surfaces show the boundary of the objective space which has been attained with a certain fraction of repetitions. For example, a 50% attainment surface shows the border of the objective space which has been attained during half of the repetitions. The attainment surface is more informative than a performance metric value but the attainment surface can be reasonably visualized only in the case of two or three objectives. In addition to analysis of a single method, different MOEAs can be compared using attainment surfaces. The comparison can be done visually and/or by calculating performance metric values from the attainment surfaces. Converting the results to numbers means once again losing some part of the information.

Visualization of Non-Dominated Solutions

Probably the best way to evaluate an obtained set of solutions is to visualize it in the objective space. In this case the evaluation will be qualitative instead of quantitative. Visualization of solutions has a more important role than merely evaluating the performance of an algorithm; visualization is important for the decision-maker who picks the final solution.

In the simplest case, visualization means plotting obtained solutions into the objective space (as done in *Publication I*, *Publication III*, *Publication IV*, and *Publication V*). Unfortunately, this approach is best suited only for cases with two objectives. Three objectives already cause problems in visualization since illustration of three dimensional objective space will be a projection to two dimensional space and, therefore, the relation between the objective values of the solutions becomes more difficult to observe. Beyond three objectives, the solution space cannot be visualized directly except as a set of projection images. However, alternative ways to visualize a set of solutions exist and the most common ways have been described in [108, pp. 239–249] and [33, pp. 316–320]. A few common visualization techniques are briefly described below:

Scatter-Plot Matrix Method plots all the pairs of objectives in different subplots, which are arranged into a matrix shape. In the matrix, the jth column of the ith row corresponds to the subplot, where the jth objective value of each solution is plotted with respect to the ith objective value.

Value Path Method presents solutions in a two-dimensional axis, where the horizontal axis corresponds to an objective identity and the vertical axis corresponds to an objective value. A single solution is presented with a cross-line, which shows values of the objectives for the solution. The value path method can be used even with a large number of objectives and solutions to get information about the value ranges and the diversity of the solutions.

Bar Chart Method is similar to the value path method except that a bar is used to illustrate an objective value of a solution. If there are N solutions with M objectives, the corresponding bar chart will have $N \times M$ bars. The usability of the bar chart method is thus limited to only a very small number of objectives and solutions.

Star Coordinate Method plots each solution with a circle, which is divided into M equal size segments. The border of each segment between the center and circumference of the circle represents a value range of a single objective. The objective values of a solution are then presented with connected points in the segment borders. The spider-web chart and the petal diagram are slightly different variations of the star coordinate method. Since a single solution is represented with a single plot, use of the method with a large number of solutions is difficult.

One recent and promising approach is to use a heatmap to visualize a solution set [120]. A heatmap is a two-dimensional array, where cells code a numerical value with their color/shade. Thus, a heatmap is a color/gray-scale picture. When heatmap visualization is used to show a result for a MOOP, one dimension of the heat-map represents individual solutions and the other represents decision variable, objective and/or constraint values. Thus, besides objective values, a heatmap can be used to visualize decision variable values and constraint values at the same time. The ability to illustrate a large amount of information (a whole set of solutions) in a quite easily interpreted form makes heatmaps an attractive choice for visualization.

2.3 Differential Evolution

The Differential Evolution (DE) algorithm [31,119] belongs to the family of EAs and the concept was introduced by Storn and Price in 1995 [134,135]. The name of the method refers to the idea of using differences between individuals to mutate an individual.

Design principles in DE are simplicity, efficiency, and the use of floating-point encoding. DE is characterized by self-adaptivity, linear scalability (*i.e.*, the computational cost of the algorithm increases linearly with the number of decision variables) and ability to perform a rotationally invariant search. Although DE uses real-coding of variables in its genetic operations, DE can be used to solve problems with different types of variables by just a simple conversion to the actual variable types prior to evaluation of an objective or constraints [95,119].

DE has been gaining in popularity in recent years because of its good performance observed in numerical optimization of practical problems. It has also performed well with a number of test problems [117, 125, 126]. The 2006 IEEE Congress on Evolutionary Computation (CEC 2006) was the first major conference to arrange a special session dedicated solely to DE, and three years later, the DE special session was the largest at the conference.

Several variations of the idea exist and these are referred to in the literature as DE strategies [119,135]. The following section describes the most used DE strategy in the literature, DE/rand/1/bin.

2.3.1 Basic Differential Evolution, DE/rand/1/bin

Basic DE is meant for unconstrained single-objective optimization and therefore notations in this section are for single-objective optimization. As in a typical EA, the idea in DE is to start with a randomly generated initial population, which is then improved using selection, mutation, and crossover operations. Several ways exist to determine a termination criterion for an EA, as mentioned in Section 2.1, but usually a predefined upper limit G_{max} for the number of generations to be computed is used. This termination condition is used also with DE in this thesis work.

Initialization of the Population

Values for the initial population in DE are typically drawn from a uniform distribution. Formally this can be presented as [118]:

$$\begin{aligned} &P_G = \{\boldsymbol{x}_{1,G}, \boldsymbol{x}_{2,G}, \ldots, \boldsymbol{x}_{NP,G}\}, \quad \boldsymbol{x}_{i,G} = (x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}) \\ &x_{j,i,0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ &i = 1, 2, \ldots, NP, \quad NP \geq 4, \quad j = 1, 2, \ldots, D\ . \end{aligned} \tag{2.16}$$

In this representation, P_G denotes a population after G generations (0 is an initial generation), $\boldsymbol{x}_{i,G}$ denotes a decision vector (or individual) of the population, and $rand_j[0,1]$ denotes a uniformly distributed random variable in the value range $[0,1]$. Terms $x_j^{(lo)}$ and $x_j^{(hi)}$ denote lower and upper parameter bounds in the initialization, respectively.

The size of the population is denoted by NP and the dimension of decision vectors is denoted by D.

It should be noted that the values of initialization bounds ($\boldsymbol{x}^{(lo)}$, $\boldsymbol{x}^{(hi)}$) can be different from the values of boundary constraints ($\boldsymbol{x}^{(LO)}$, $\boldsymbol{x}^{(HI)}$) in the problem definition (*cf.* Section 2.2). For example, some decision variables might be unbounded in the problem definition, but some lower and upper bounds are still needed to initialize these variables. It should be noted that DE is able to advance the search out of the initialization bounds of the decision variables if this is not restricted.

Other ways of initialization also exist, *e.g.*, if some knowledge exists about the position of the optimum, part of the initial population may be initialized around a possible position of the optimum using normal distribution.

Mutation and Crossover

DE goes through each decision vector $\boldsymbol{x}_{i,G}$ of the population and creates a corresponding trial vector $\boldsymbol{u}_{i,G}$ as follows [118]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}\,, \\
&\text{(randomly selected,} \\
&\quad \text{except mutually different and diffrent from } i\text{)} \\
&j_{rand} = \text{round}\,(rand_i[0,1) \cdot D) \\
&\text{for}(j = 1; j \leq D; j = j + 1) \\
&\{ \\
&\quad \text{if}(rand_j[0,1) < CR \vee j == j_{rand}) \\
&\qquad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\
&\quad \text{else} \\
&\qquad u_{j,i,G} = x_{j,i,G} \\
&\}\,.
\end{aligned}
\tag{2.17}
$$

Indices r_1, r_2, and r_3 are mutually different and drawn from the set of the population indices. Function round() rounds to the nearest integer. Functions $rand_i[0,1)$ and $rand_j[0,1)$ return a random number drawn from the uniform distribution between 0 and 1 for each i and j. Both CR and F are user defined control parameters for the DE algorithm and they remain fixed during the whole execution of the algorithm. Parameter CR, controlling the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors and not from the old decision vector $\boldsymbol{x}_{i,G}$. The condition $j == j_{rand}$ ensures that at least one element of the trial vector is different compared to the elements of the old vector. Parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$ (*i.e.*, larger than 0 and the upper limit is in practice around 1 although there is no hard upper limit). Effectively, CR controls the rotational invariance of the search [6], and smaller values (*e.g.*, 0.1) are more suitable with separable problems while larger values (*e.g.*, 0.9) are for non-separable problems [118]. Control parameter F controls the speed and robustness of the search, *i.e.*, a lower value for F increases the convergence rate but also the risk of getting

[6] The search is rotationally invariant if it is independent from the rotation of the coordinate axis of the search space. Rotationally invariant search is preferable if the problem is not separable, as is the case with most practical problems [97, 129].

stuck into a local optimum [118]. Parameters CR and NP have a similar effect on the convergence rate as F [83, 118].

The difference between two randomly chosen vectors, $\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}$, defines the magnitude and direction of the mutation. When the difference is added to a third randomly chosen vector $\boldsymbol{x}_{r_3,G}$, this change corresponds to mutation of this third vector. The basic idea of DE is that the mutation is self-adaptive to the objective function space and to the current population. It is worth noting that the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [59] is also self-adaptive and able to perform a rotationally invariant search but with the computational burden of covariance matrix calculations that scale unfavorably with the dimensionality of the problem. At the beginning of the optimization process with DE, the magnitude of mutation is large because vectors in the population are far away from each other in the search space. When the evolution proceeds and the population converges, the magnitude of mutations gets smaller. Thus, the self-adaptive mutation of DE allows a global or local search, whichever is the more appropriate.

Selection

After each mutation and crossover operation the trial vector $\boldsymbol{u}_{i,G}$ is compared to the old decision vector $\boldsymbol{x}_{i,G}$. If the trial vector has an equal [7] or lower objective value, then it replaces the old vector. This can be presented as follows [118]:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if} \quad f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} . \tag{2.18}$$

The average objective value of the population will never deteriorate, because the trial vector replaces the old vector only if it has equal or lower objective value. Therefore, DE is an elitist search method.

Overall Algorithm

The overall presentation of basic DE (sometimes also referred as "classic DE") is presented in Figure 2.5 [118]. This DE strategy is identified in the DE literature with the notation DE/rand/1/bin. In this notation, 'rand' indicates how the vector for mutation is selected. The number of vector differences used in the mutation is indicated next, and 'bin' indicates the way the old vector and the trial vector are recombined. A number of other DE strategy variants also exists [25, 118, 119, 135].

An empirical comparison study between several DE strategies with a set of single-objective problems was conducted in [107]. It was concluded that a DE/best/1/bin strategy generally performed best for the problem set but based on the result, DE/rand/1/bin also performed well. Limitations of the study in [107] were also noted by referring to the No Free Lunch (NFL) theorem [144]. In general, the performance difference between the two strategies is that DE/best/1/bin is greedier and faster but DE/rand/1/bin is more reliable and therefore performs better with harder problems [119, pp. 154–156].

[7] Preferring the trial vector in the case of equal objective values has importance if the objective landscape contains a plateau; preferring the old vector would cause the search to stagnate on the plateau.

$$
\begin{array}{l}
\text{Input : } D, G_{max}, NP \geq 4, F \in (0, 1+], CR \in [0,1], \text{and initial bounds: } \boldsymbol{x}^{(lo)}, \boldsymbol{x}^{(hi)} \\
\text{Initialize : } \left\{ \begin{array}{l} \forall i \leq NP \wedge \forall j \leq D : x_{j,i,0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)} \right), \\ i = \{1, 2, \ldots, NP\}, j = \{1, 2, \ldots, D\}, G = 0, rand_j[0,1] \in [0,1] \end{array} \right.
\end{array}
$$

$$
\left\{ \begin{array}{l}
\text{While } G < G_{max} \\
\quad \left\{ \begin{array}{l}
\text{Mutate and recombine:} \\
r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \text{randomly selected,} \\
\qquad \text{except mutually different and different from } i \\
j_{rand} \in \{1, 2, \ldots, D\}, \text{randomly selected for each } i \\
\\
\forall i \leq NP \left\{ \begin{array}{l} \forall j \leq D, u_{j,i,G} = \left\{ \begin{array}{ll} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) & \\ \qquad \text{if} \quad rand_j[0,1) < CR \vee j == j_{rand} & \\ x_{j,i,G} \quad \text{otherwise} & \end{array} \right. \\ \\ \text{Select :} \\ \boldsymbol{x}_{i,G+1} = \left\{ \begin{array}{ll} \boldsymbol{u}_{i,G} & \text{if} f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{array} \right. \end{array} \right. \\
G = G + 1
\end{array} \right.
\end{array} \right.
$$

Figure 2.5: Basic DE algorithm

The stagnation possibility of the DE/rand/1/bin strategy has been discussed in [97]. It is possible that the search stagnates or premature convergence occurs before reaching the global optimum. These two cases can be distinguished by observing the diversity of the population (diversity is lost in the case of premature convergence). The probability of stagnation or premature convergence can be reduced by increasing the size of the population and/or F. The search can be also repeated several times to increase confidence. The strategy can be simply modified to fulfill the convergence properties given in Section 2.1: Adding random values from a non-finite probability (*e.g.*, Gaussian) distribution to decision variable values when the trial vector is created is sufficient to guarantee convergence to the global optimum.

2.3.2 Differential Evolution for Multiple Objectives

Several extensions of DE for multi-objective optimization have been proposed. Early proposals converted a MOOP into a single-objective form (*e.g.*, [5,16,143]). Later proposals are mainly based on Pareto-dominance. In the following, these later proposals are first listed in chronological order and then described briefly.

The first method extending DE for multi-objective optimization using the Pareto approach was the Pareto-based DE approach [17]. Pareto Differential Evolution [11] was also mentioned around the same time, unfortunately without an explicit description of the method. After these two methods, the Pareto(-frontier) Differential Evolution (PDE) algorithm [2,3] and the first version of Generalized Differential Evolution (GDE1) [90] were introduced. Then, Self-adaptive PDE (SPDE) [1], the Pareto DE Approach (PDEA) [102], Adaptive Pareto DE (APDE) [150,151], Multi-Objective DE (MODE) [145], Vector Eval-

uated DE (VEDE) [115], the second version of GDE (GDE2) in *Publication III*, and Non-dominated Sorting DE (NSDE) [68] were proposed. Later, DE for Multiobjective Optimization (DEMO) [124], the third version of GDE (GDE3) in *Publication IV*, and ϵ-MyDE [130] were introduced. Some of the most recent proposals are DE for Multiobjective Optimization with Random Sets (DEMORS) [60], Multiobjective DE based Decomposition (MODE/D) [99], ϵ-Constraint method with Cultured DE (ϵ-CCDE) [9], the DE algorithm based on ϵ-dominance and an orthogonal design method (ϵ-ODEMO) [15], Cluster-Forming Differential Evolution (CFDE) [74], Multi-objective Differential Evolution with local dominance and scalar selection (MODE-LD+SS) [113], Adaptive Multi-objective Differential Evolution with Stochastic Coding Strategy (AS-MODE) [154], and Multi-Objective Differential Evolution Algorithm (MODEA) [4].

In addition to new algorithm proposals, other relevant studies also exist. One of these is about incorporating directional information in the selection of vectors for the mutation step of DE [69]. Comparison between GA and DE in multi-objective optimization has been done in [140] and it has been concluded that DE explores the decision variable space more efficiently than GA. A comparison between four different multi-objective DE variants is presented in [155]. The variants differ in balancing between convergence and diversity. Based on experimental results it is found that the balancing technique that is used, *e.g.*, in DEMO and the third version of GDE is better than the one used, *e.g.*, in PDEA. This same observation has been noted also in [139].

In the following, the aforementioned multi-objective DE approaches are described briefly. A more detailed review of most of the approaches is available in [25, pp. 596 – 604] and [106]. Versions of GDE are described in more detail in the next chapter.

Pareto-based DE Approach

A Pareto-based DE approach was proposed by Chang *et al.* in 1999 for multi-objective tuning of automatic train operation [17]. The approach collects non-dominated points found during the DE optimization process into an archive. The diversity of the population is maintained by using fitness sharing. Fitness sharing means that the fitness value of crowded solutions in the objective space will be degraded to avoid over-crowding [33, pp. 149–160]. When trial vectors are compared against old vectors, the comparison is done using shared fitness values. The old vector is selected only if it is better with respect to all the objectives than the trial vector; otherwise the trial vector is selected.

Pareto(-frontier) Differential Evolution (PDE) algorithm

The Pareto(-frontier) Differential Evolution (PDE) algorithm was proposed by Abbass *et al.* in 2001 [2,3]. It modifies basic DE in several ways, namely, the initial population is initialized using a Gaussian distribution $N(0.5, 0.15)$, the mutation factor F is generated from the Gaussian distribution $N(0, 1)$, and only non-dominated solutions are used for reproduction. The trial vector is selected for the next generation if it dominates the old vector; otherwise the old vector is selected. If the number of non-dominated solutions increases above an allowed maximum, the most crowded members are removed based on the distance to their nearest neighbors.

Self-adaptive PDE (SPDE)

Self-adaptive PDE (SPDE) was proposed by Abbass in 2002 [1]. SPDE is an extension of PDE to include automatic adaptation of control parameters during the optimization process. Crossover and mutation rate values are coded into individuals in the same way as decision variable values, *i.e.*, rate values are additional elements in decision vectors containing decision variable values. Control parameter values also evolve similarly to decision variable values. Since F of basic DE is replaced with a random number as in PDE, the mutation rate controls one extra mutation.

Pareto DE Approach (PDEA)

The Pareto DE Approach (PDEA) was proposed by Madavan in 2002 [102]. PDEA is otherwise the same as NSGA-II except that children are generated using DE instead of GA.

Adaptive Pareto DE (APDE)

Adaptive Pareto DE (APDE) was proposed by Zaharie in 2003 [150, 151]. APDE is the same as PDEA or NSGA-II except the evolutionary search engine is replaced with Adaptive Differential Evolution (ADE), which was proposed also by Zaharie [149]. ADE has a parameter adaptation guided by a theoretical rule about the evolution of population variance.

Multi-Objective DE (MODE)

Multi-Objective DE (MODE) was proposed by Xue *et al.* in 2003 [145]. MODE is another approach combining DE and NSGA-II. In MODE, basic DE is modified and an extra crowding parameter is introduced to prevent similar individuals entering the next generation. Convergence analysis of MODE and a theoretical convergence proof are given in [146,147]. The analysis makes an assumption that DE is able to reach any point in the search space during the search process. In general, this assumption is not valid if any of the most common DE strategies (*e.g.*, DE/rand/1/bin) are used.

Vector Evaluated DE (VEDE)

Vector Evaluated DE (VEDE) was proposed by Parsopoulos *et al.* in 2004 [115]. VEDE is basically the same as Vector Evaluated GA (VEGA) except that GA is replaced with DE. Thus, the population is divided into M subpopulations and each subpopulation optimizes only one objective out of M objectives. As with VEGA, VEDE does not have any diversity preservation technique.

Non-dominated Sorting DE (NSDE)

Non-dominated Sorting DE (NSDE) was proposed by Iorio and Li in 2004 [68]. NSDE is the same as PDEA except that a different DE strategy (DE/current-to-rand/1) is used

in NSDE. A better performance of NSDE over NSGA-II with rotated MOOPs [8] has been reported in [68].

DE for Multiobjective Optimization (DEMO)

DE for Multiobjective Optimization (DEMO) was proposed by Robič and Filipič in 2005 [124]. DEMO works in the same way as PDEA except that if a dominance relation exists between the trial and old vector, then the dominated vector is directly rejected. This lowers the number of stored solutions at the end of a generation. Therefore, the number of solutions to be sorted and reduced is smaller than with PDEA. DEMO modifies the basic DE algorithm also in such a way that selected vectors are directly included into the current population when the trial and old vectors are compared [9].

ϵ-MyDE

ϵ-MyDE was proposed by Santana-Quintero and Coello Coello in 2005 [130]. This approach uses different archives for non-dominated solutions and the actual population. Use of ϵ-dominance helps to maintain diversity among found non-dominated solutions. The approach also has several smaller design details, *e.g.*, the selection of solutions for mutation has been modified and the basic DE algorithm has been changed to include an additional mutation. The selection between the trial and old vectors selects the dominating vector if one exists; otherwise a random choice is made.

DE for Multiobjective Optimization with Random Sets (DEMORS)

DE for Multiobjective Optimization with Random Sets (DEMORS) was proposed by Hernández-Diaz *et al.* in 2006 [60]. DEMORS is based on ϵ-MyDE and rough sets theory. It has a two-phase approach combining a global search using DE with a Pareto-adaptive ϵ-dominance and a local search using rough sets. The approach uses different archives for non-dominated and dominated solutions besides the actual population.

Multiobjective DE Based on Decomposition (MODE/D)

Multiobjective DE based on decomposition (MODE/D) was proposed by Li and Zhang in 2006 [99]. MODE/D is the same as MOEA/D mentioned in Section 2.2.5 except that the EA is replaced with DE.

MODE/D differs from the other DE approaches listed above since it does not perform Pareto-optimization but multiple single-objective optimization with scalarized objectives. Improvements to MODE/D have been proposed in [101].

[8] In a rotated MOOP the decision variable space has been rotated around the origin with a rotation matrix M that is linear and orthogonal. A rotated MOOP can be obtained from a MOOP by evaluating objectives using $M \times \boldsymbol{x}$ as an argument instead of $\boldsymbol{x}$. The rotation matrix M can have different rotation angles to different directions and these angles can be obtained randomly as long as M remains constant during the search.

[9] Instant update of the population was originally considered by the developers of DE but use of the temporary population was adopted mainly because of parallelization issues [119, p. VIII]. Instant replacement might also lead to premature convergence.

ϵ-Constraint Method with Cultured DE (ϵ-CCDE)

ϵ-Constraint method with Cultured DE (ϵ-CCDE) was proposed by Becerra and Coello Coello in 2006 [9]. Similarly to MODE/D, also ϵ-CCDE applies single-objective optimization to solve multi-objective problems. In ϵ-CCDE the classical ϵ-constraint method is hybridized with a cultural algorithm called Cultured Differential Evolution.

Results are promising but the method was only tested with bi-objective test problems due to implementation difficulties with a higher number of objectives.

DE Algorithm Based on ϵ-Dominance and an Orthogonal Design Method (ϵ-ODEMO)

A DE algorithm based on ϵ-dominance and an orthogonal design method (ϵ-ODEMO) was proposed by Cai *et al.* in 2007 [15]. ϵ-ODEMO uses the orthogonal design method to generate an initial population and then the ϵ-dominance concept to maintain an archive of non-dominated solutions. The use of ϵ-dominance maintains the distribution of solutions without a separate diversity preservation technique.

Cluster-Forming Differential Evolution (CFDE)

Cluster-Forming Differential Evolution (CFDE) was proposed by Justesen and Ursem in 2009 [74]. CFDE is based on defining a desired number of final solutions and using the same number of subpopulations during the search to form distinctive clusters. The approach is based on DEMO but the secondary fitness measure based on diversity is changed with two alternating fitness measures. Reduction of the population after each generation is performed in individual subpopulations.

Multi-Objective Differential Evolution with Local Dominance and Scalar Selection (MODE-LD+SS)

Multi-objective Differential Evolution with local dominance and scalar selection (MODE-LD+SS) was proposed by Montaño, Coello Coello, and Mezura-Montes in 2010 [113]. MODE-LD+SS incorporates a concept of local dominance in order to improve convergence and a scalar selection mechanism for a combined parent and child population in order to find non-dominated solutions covering the Pareto-optimal front.

Adaptive Multi-Objective Differential Evolution with Stochastic Coding Strategy (AS-MODE)

Adaptive Multi-objective Differential Evolution with Stochastic Coding Strategy (AS-MODE) was proposed by Zhong and Zhang in 2011 [154]. A stochastic coding strategy means that each individual is presented as a stochastic region that is sampled in order to find better solutions. The control parameters are adjusted using probability-based adaption instead of fixed parameter values. The members of a new population are selected using non-dominated sorting and crowding distance.

Multi-Objective Differential Evolution Algorithm (MODEA)

Multi-Objective Differential Evolution Algorithm (MODEA) was proposed by Ali, Siarry, and Pant in 2012 [4]. It uses an Opposition-Based Learning for generating an initial population, a modified mutation (called as randomized localization), and a selection operator that can been seen as a compromise between the selection mechanism of PDEA and DEMO/GDE3. The members of the next generation are selected in the same way as in NSGA-II.

2.3.3 Differential Evolution and Constraints

Besides solving problems with multiple objectives, DE has also been modified for handling problems with constraints [18, 96, 100, 133, 142]. Many of these approaches are based on applying penalty functions, which has the problem of selecting penalty parameters as noted earlier. To overcome this problem, the following set of selection rules have recently been used extensively [105]:

1. Between two feasible solutions, the one with a better fitness value is selected.
2. If one solution is feasible and the other one is infeasible, the feasible solution is selected.
3. Between two infeasible solutions, the one with less constraint violation is selected.

The amount of constraint violation in the third rule has been measured either with the Pareto-dominance relation between constraint violations [92, 93] or as the sum of constraint violations [104]. The first approach is used in all the versions of Generalized Differential Evolution described in the next chapter.

Chapter III

Generalized Differential Evolution

"Everything should be made as simple as possible, but no simpler."
– Albert Einstein

This chapter contains a summary of the work done by the author within the scope of this thesis. The algorithms and the related studies presented in the attached publications of this thesis work, as well as several other articles by the author, are discussed below.

The main outcome of the thesis work is study and further development of the method called *Generalized Differential Evolution (GDE)*. GDE is an extension of DE for optimization with several objectives and constraints. Unlike several other DE approaches for constrained and/or multi-objective optimization, GDE requires no extra control parameters compared to the original DE.

A primary goal of GDE has been to keep changes to DE as simple as possible and to avoid unnecessary complexity. The key idea and justification for the name is that the extension falls back to basic DE in the case of an unconstrained single-objective problem. This property distinguishes GDE from the multi-objective DE approaches described in Section 2.3.2. Thus, GDE is a single- and multi-objective optimizer, and relatively simple compared to the other approaches.

GDE uses the DE/rand/1/bin strategy described in Section 2.3.1. This strategy was chosen for GDE because of its simplicity and good observed performance. The strategy is also the most commonly used DE strategy in the literature [25, p. 594]. Different DE strategies were not compared since the main focus was the multi-objective part of the method, and not the search method used to create trial solutions.

Several GDE development versions were developed and they differ in the way multi-objective optimization is performed – more precisely, how the diversity of solutions is maintained during the search. In the case of multiple objectives, all the GDE versions perform *a posteriori* optimization as other modern MOEAs.

GDE can be implemented in such a way that the number of function evaluations is reduced since the constraint-domination relation (*cf.* Section 2.2.3) is used in the selection. Even comparison between single constraint values can reveal that the trial vector does not constraint-dominate the old vector, and therefore the old vector is preserved. This reduces the number of constraint function evaluations needed compared to evaluation of

all the constraints, an approach used with most constraint handling approaches. This reduction of constraint evaluation is helpful in the case of many and/or computationally expensive constraint functions.

Different development versions of GDE are described in the following section. Their performance is demonstrated with common bi- and tri-objective test problems mentioned in Section 2.2.6 and defined in Appendix I. The results are shown in Figures 3.1–3.12 and Tables 3.1–3.2.

Problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 were selected from the set of ZDT problems. These problems are solved using 250 generations and a population size of 100 [1]. Problems DTLZ1, DTLZ2, DTLZ4, DTLZ5, and DTLZ7 were selected from the set of DTLZ problems. These problems are solved using 250 generations and a population size of 200. It is justified to use a larger population size than with the ZDT problems to approximate the Pareto-optimal front since the objective space is three dimensional instead of two. The control parameter values $CR = 0.2$ and $F = 0.2$ are used with the problems with the exception that values $CR = 0.0$ and $F = 0.5$ are used with ZDT4 [2]. These values were selected based on experiments and observations with these problems during this thesis work. Small CR values are used for the problems since the problems are separable (*cf.* Section 2.3.1). Another reason for small CR values is that the objectives of the ZDT problems, especially ZDT4, exhibit different degree of difficulty. This means that solving different individual objectives needs different computational effort. Using a large CR value would lead to a premature convergence along the first objective far before the second objective converges, as noted in [81], *Publication II*, and *Publication VI*.

In all the experiments described in this chapter, possible boundary constraint violations have been handled by reflecting violating decision variable values back from the violated boundary using the technique described in Section 2.2.2.

Since the DE/rand/1/bin strategy has been used in the GDE versions, in theory, convergence to the Pareto-optimal front cannot be guaranteed since not all the points in the search space are necessarily attainable (*cf.* Sections 2.1 and 2.2.5). However, empirically GDE has been noted to converge well, and the latest version, GDE3, has been seen to provide good approximations of the Pareto-optimal fronts, as can be noted from the results in the following sections and related publications (*e.g.*, [84]). Theoretical convergence issues of GDE will be discussed further in Chapter 4.

3.1 First Version, GDE1

The first version, GDE1, was proposed by Lampinen [90] as a further development of the constraint handling approach based on dominance relation [92], and the name Generalized Differential Evolution appeared for the first time in [81]. GDE1 extends the basic DE algorithm for constrained multi-objective optimization by simply modifying the selection

[1] The number of generations and the population size come from [158], where ZDT problems were used to compare several MOEAs. Afterwards, these values have been used almost as a standard with the ZDT problems

[2] ZDT4 has multiple equally spaced local Pareto-optimal fronts and $F = 0.5$ advances moving from one local front to another (*Publication II* and *Publication VI*).

operation of DE. In GDE1, the selection operation is based on constraint-domination defined in Section 2.2.3 and can be defined as:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if} \quad \boldsymbol{u}_{i,G} \preceq_c \boldsymbol{x}_{i,G} \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} . \tag{3.1}$$

The weak constraint-domination relation is used to maintain congruity with the selection operation of DE. Thus, in the case of equality, the trial vector is preferred. One should note that the selection is fully elitist in the sense of Pareto-dominance, *i.e.*, the best solutions cannot be lost during the search.

As mentioned earlier, one benefit of using the dominance relation in the selection is that it can be implemented in such a way that the number of function evaluations is reduced because all the constraints and objectives do not always need to be evaluated. Inspecting constraint violations (even one constraint) is often enough to determine which vector to select for the next generation [92,119]. Depending on the problem, the reduction can be truly remarkable as noted in [83,91]. In practice, it is wise to evaluate computationally expensive functions last, since the later a function is in the evaluation order, the fewer times it gets evaluated. The order of the functions also has an effect on the search process since the search is directed at the beginning according to the first constraints and objectives. For example, evaluation of the first constraints can already determine the comparison between the target and trial vectors and the rest of the constraints and objectives then have no effect on the comparison.

GDE1 does not have any kind of diversity preservation, which is rare for modern MOEAs. Nevertheless, GDE1 has been able to provide good results with some problems in *Publication I* and [82]. It has, however, been found to be rather sensitive to the selection of the control parameter values, as noted in *Publication II*.

The final populations for the ZDT and DTLZ problems are shown in Figures 3.1 and 3.2 with the Pareto-optimal fronts. The members of the final population are shown in the figures in addition to the non-dominated members to give a better idea about the behavior of the search method. The difficulty in finding the concave Pareto-optimal front of ZDT2 is observable in Figure 3.1. Furthermore, only the edges of the Pareto-optimal front of DTLZ4 are found in Figure 3.2. DTLZ4 is similar to DTLZ2 except that preservation of the diversity of solutions is more difficult in the case of DTLZ4.

3.2 Second Version, GDE2

The second version, GDE2, introduced a diversity preservation operation to GDE in *Publication III*. Again, only the selection operation of basic DE was modified. The selection is done based on crowding in the objective space when the trial and old vector are feasible and incomparable based on Pareto-dominance [3]. More formally, the selection

[3] Solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are incomparable if neither $\boldsymbol{x}_1$ weakly dominates $\boldsymbol{x}_2$ nor $\boldsymbol{x}_2$ weakly dominates $\boldsymbol{x}_1$ [164]

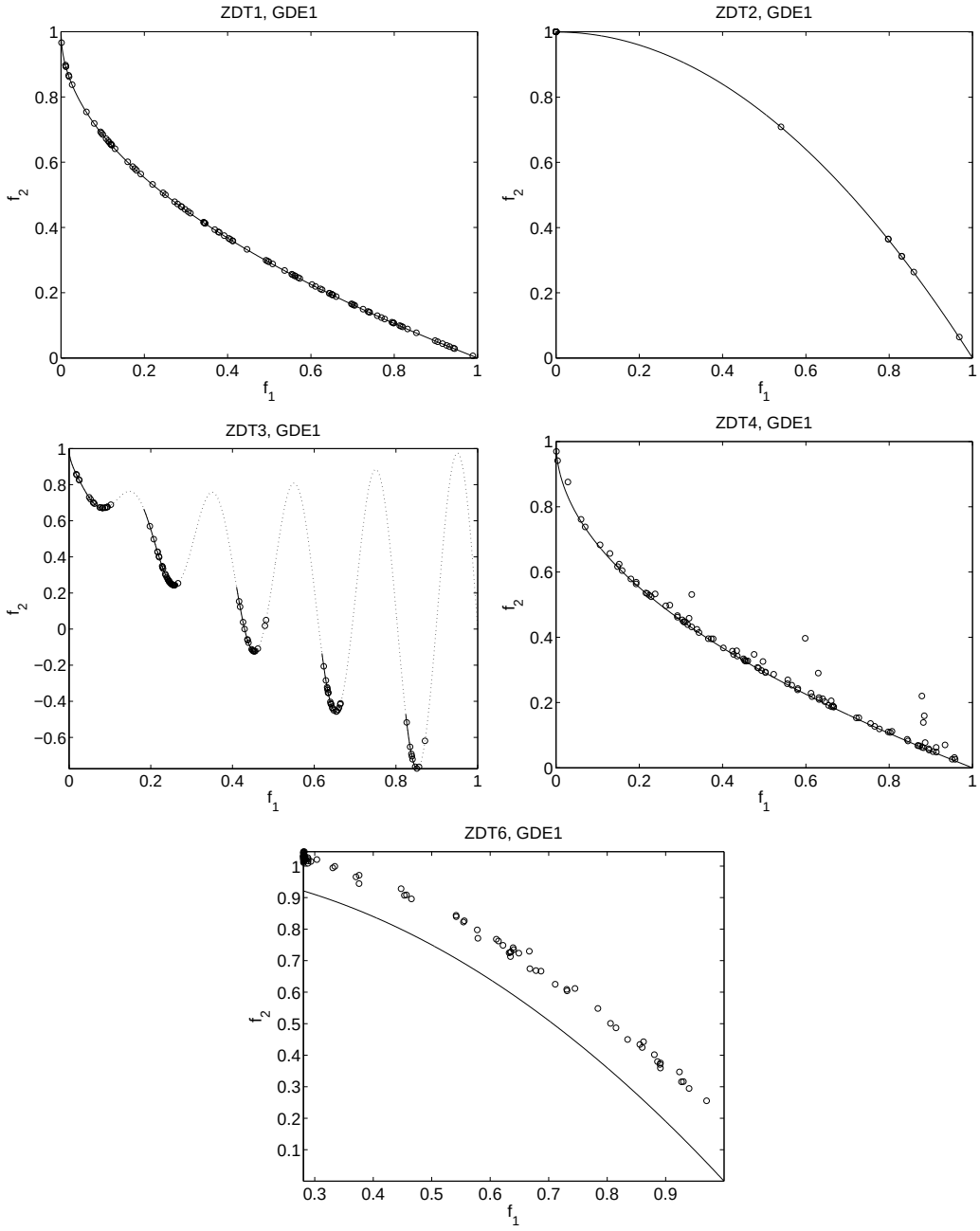


Figure 3.1: Results for the ZDT problems using GDE1.

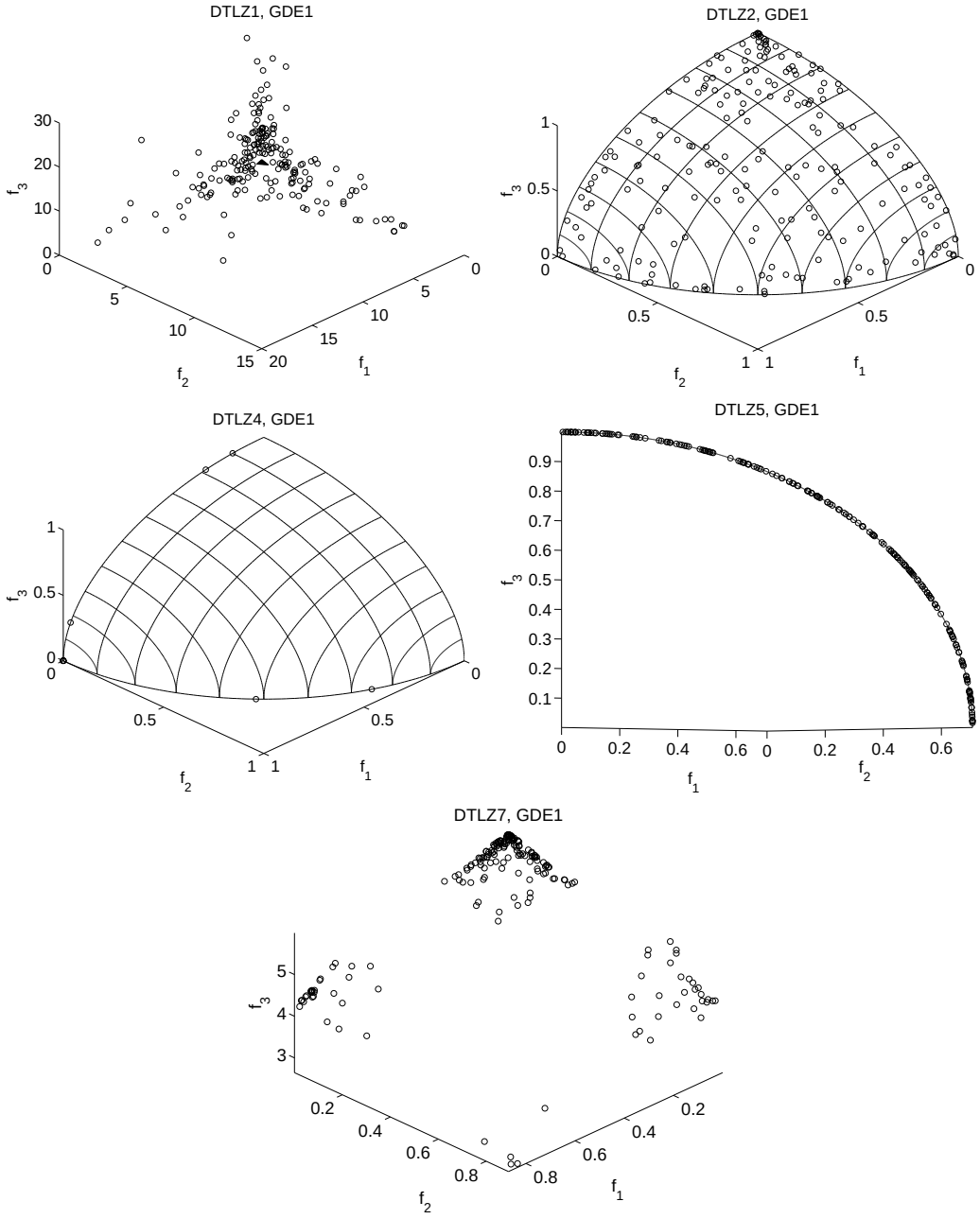


Figure 3.2: Results for the DTLZ problems using GDE1.

operation is now:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if} \begin{cases} \boldsymbol{u}_{i,G} \preceq_c \boldsymbol{x}_{i,G} \\ \vee \\ \begin{cases} \forall j \in \{1,\ldots,K\} : g_j(\boldsymbol{u}_{i,G}) \leq 0 \\ \wedge \\ \boldsymbol{x}_{i,G} \not\prec \boldsymbol{u}_{i,G} \\ \wedge \\ d_{\boldsymbol{u}_{i,G}} \geq d_{\boldsymbol{x}_{i,G}} \end{cases} \end{cases} \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases}, \tag{3.2}$$

where d_i is a distance measure for measuring the distance of a particular solution i to its neighbor solutions. Implementation was done using the crowding distance of NSGA-II. However, as noted in *Publication III*, any other distance measure could be used instead of the crowding distance. The use of another distance measure is advisable if the number of objectives is more than two, since the crowding distance no longer estimates true crowding in such cases, as noted in *Publication V*.

The selection operation is illustrated as a flowchart in Figure 3.3. Since, the Pareto-dominance relation in not the only criterion in the selection, loss of Pareto-optimal solutions is possible during the search.

As non-dominated sorting is not used, crowding is measured among the whole population. This improves the extent and distribution of the obtained set of solutions but slows down the convergence of the overall population because it favors isolated solutions far from the Pareto-optimal front until all the solutions are converged near the Pareto-optimal front. GDE2, similar to GDE1, has been noted to be rather sensitive to the selection of the control parameters values.

The final populations for the ZDT and DTLZ problems are shown in Figures 3.4 and 3.5. The better diversity obtained for ZDT2 and DTLZ4 compared to GDE1 can be observed by comparing Figures 3.1 and 3.2 to Figures 3.4 and 3.5. A slower convergence compared to GDE1 can be observed with ZDT3 and DTLZ1.

3.3 Third Version, GDE3

The third version is GDE3 (*Publication IV* and *Publication V*). In addition to the selection operation, another part of basic DE has also been modified: Now, both vectors are saved, when comparing feasible and incomparable solutions (the selection operation is illustrated as a flowchart in Figure 3.6). Therefore, at the end of a generation, the size of the population may be larger than it originally was. If this is the case, the population is then decreased back to the original size based on a similar selection approach as used in NSGA-II and shown in Figure 2.4. Population members are sorted based on goals for *a posteriori* optimization given in Section 2.2.1. The worst population members according to non-dominance and crowding are removed to decrease the size of the population to the original size. Non-dominance is the primary sorting criterion and crowding is the secondary sorting criterion as in NSGA-II. From a non-empty set of solutions, it is always possible to find the last non-dominated set and from this set it is possible to find the

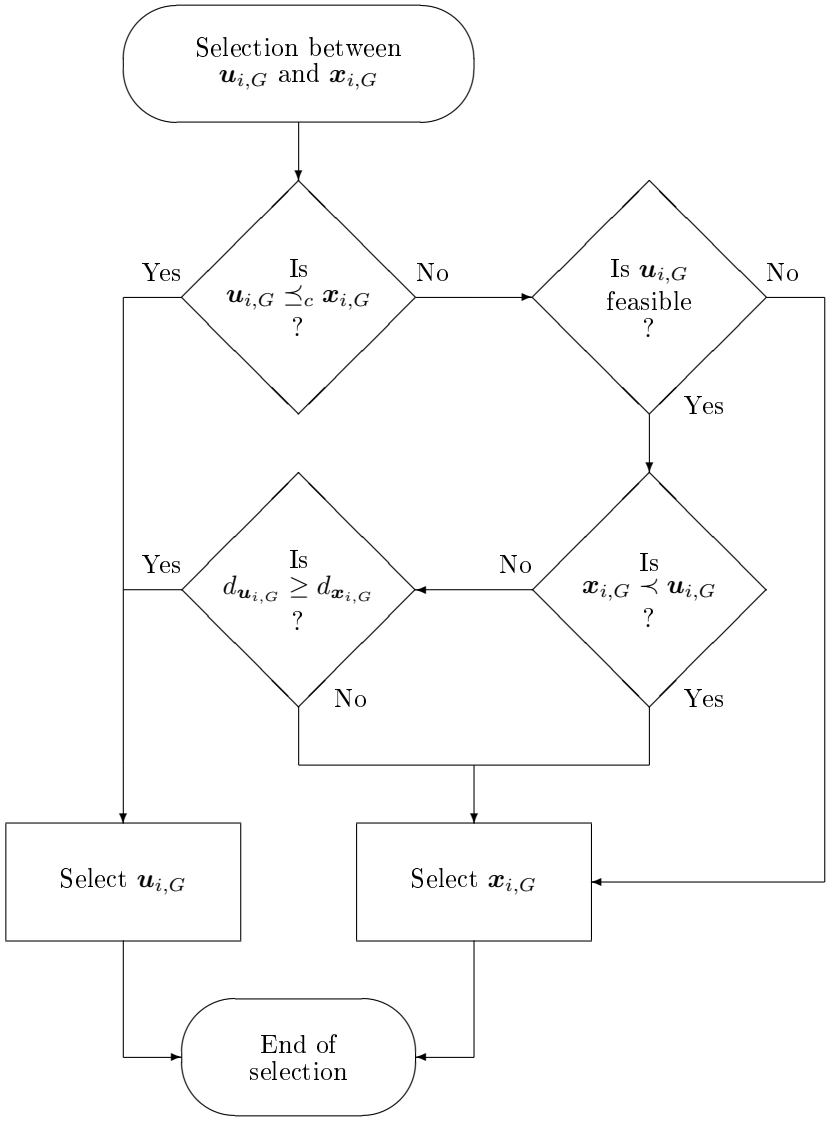


Figure 3.3: Operation for selection between the old vector $\boldsymbol{x}_{i,G}$ and trial vector $\boldsymbol{u}_{i,G}$ in GDE2.

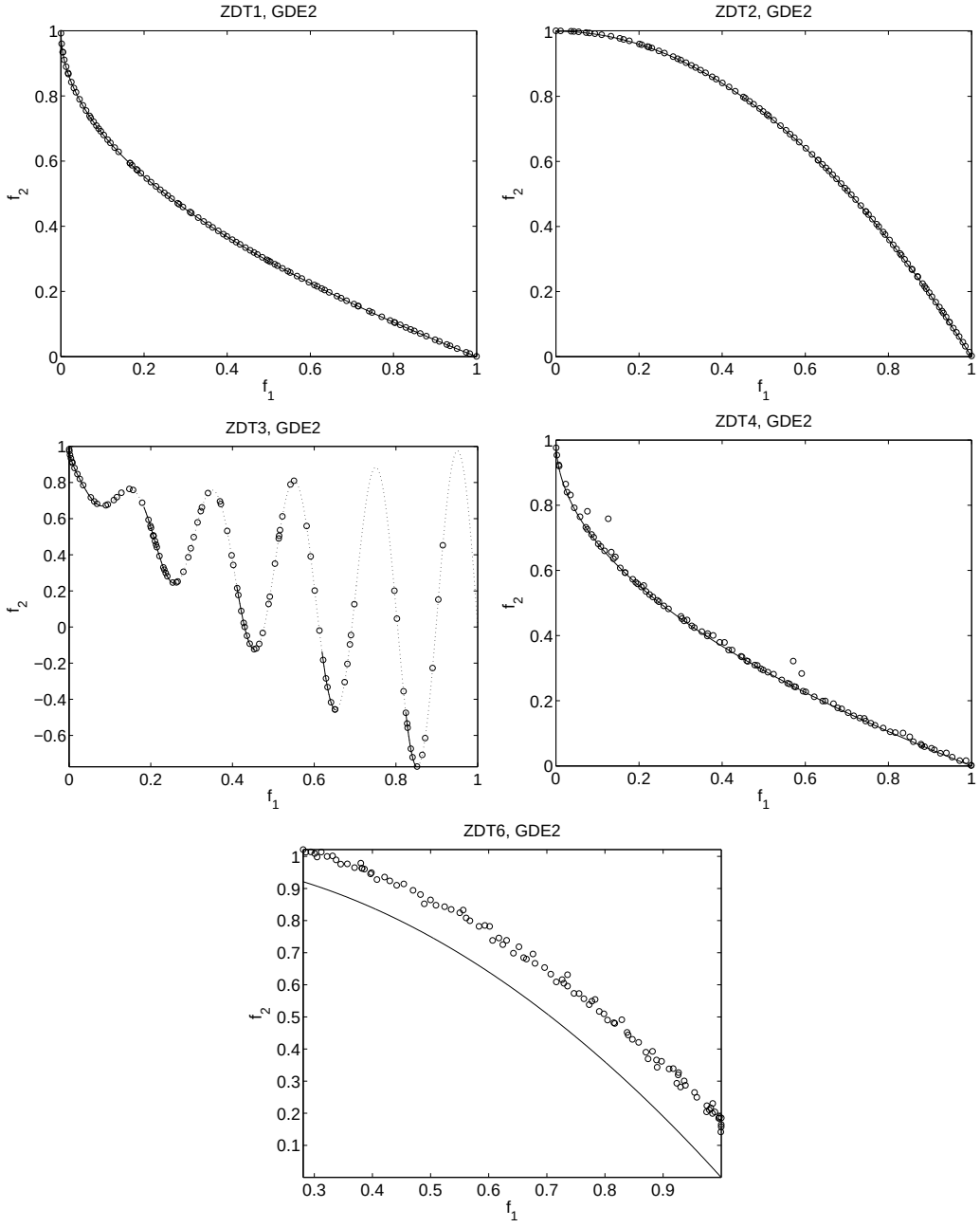


Figure 3.4: Results for the ZDT problems using GDE2.

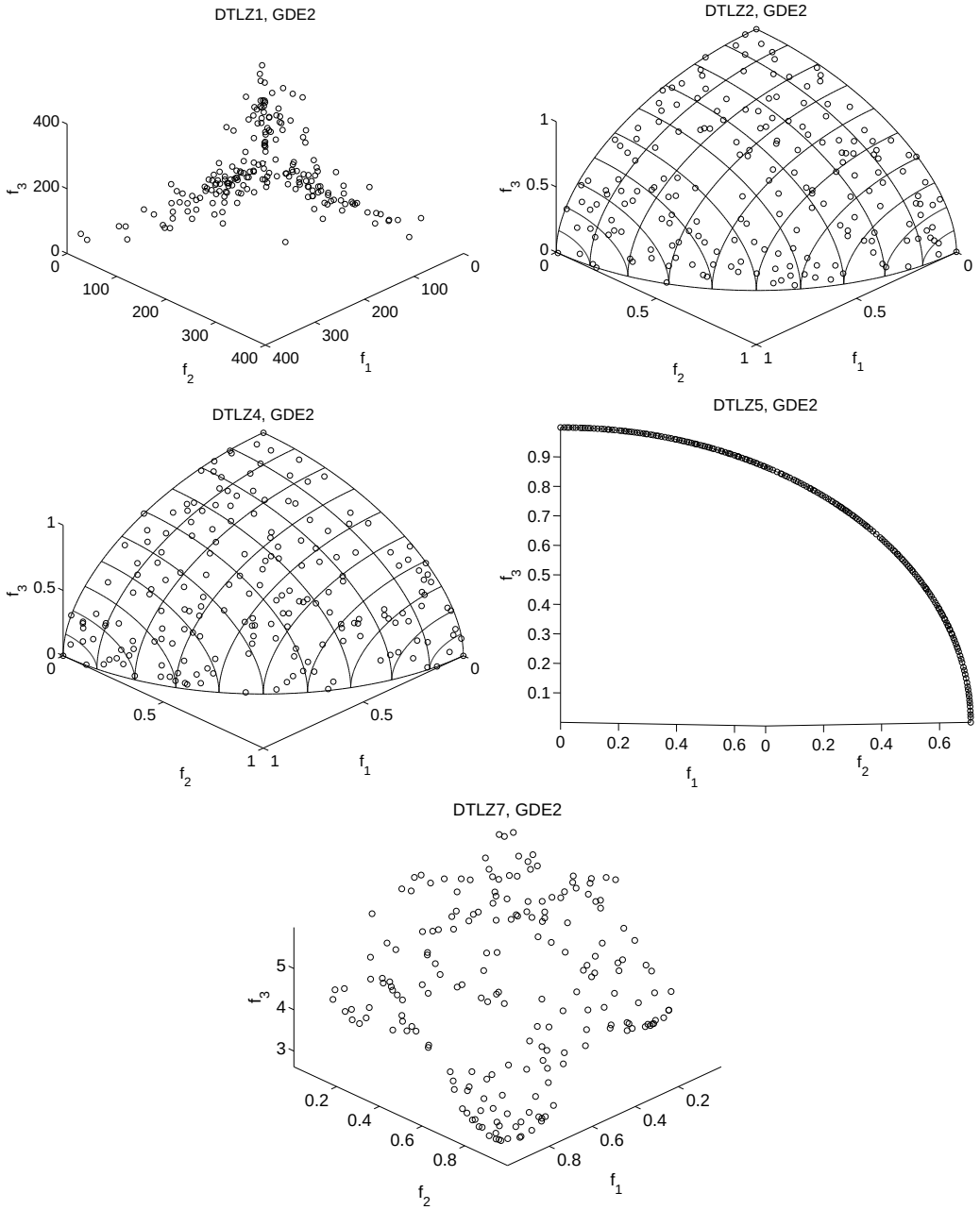


Figure 3.5: Results for the DTLZ problems using GDE2.

most crowded solution (if two solutions have the same crowding measure, one could be selected randomly for removal). Therefore, pruning of solutions can be aways performed.

Non-dominated sorting is modified to also take constraints into consideration. The selection based on the crowding distance is improved over the original method of NSGA-II to provide a better distributed set of vectors. This improvement is described in the following section and *Publication V*.

The whole GDE3 is presented in Figure 3.7. Parts that are new compared to previous GDE versions are framed in Figure 3.7. Without these parts, the algorithm is identical to GDE1. GDE3 can be seen as a combination of GDE2 and PDEA. GDE3 is similar to DEMO [124] except DEMO does not contain constraint handling nor fall back to basic DE in the case of a single objective because DEMO modifies the basic DE (*cf.* Section 2.3.2) and does not consider weak dominance in the selection. Moreover, GDE3 has an improved diversity maintenance compared to DEMO.

When $M = 1$ and $K = 0$, there are no constraints to be evaluated and the selection is simply

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if} \quad f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} , \tag{3.3}$$

which is the same as for the basic DE algorithm. The population size does not increase because this requires that $\boldsymbol{u}_{i,G}$ and $\boldsymbol{x}_{i,G}$ do not dominate each other even weakly, which cannot be true in the case of a single objective. Since the population size does not increase, there is no need to remove elements. Therefore, GDE3 is identical to basic DE in this case.

In NSGA-II and PDEA, the size of the population after a generation is $2NP$, which is then decreased to NP. In GDE3 and DEMO, the size of the population after a generation is between NP and $2NP$ because the size of the population is increased only if the trial vector and the old vector are feasible and incomparable. This will reduce the computational cost of the whole algorithm. DEMO and GDE3 emphasize convergence over diversity more than NSGA-II and PDEA, as noted in [155].

Decreasing the size of the population at the end of a generation is the most complex operation in the algorithm. The reduction is done by using non-dominated sorting and pruning based on crowding. The run-time complexity of non-dominated sorting is $O\left(NP \log^{M-1} NP\right)$ [70]. Pruning of population members based on crowding is also a complex operation. In this case the operation is performed using the crowding distance and can be performed in time $O\left(MNP \log NP\right)$ (*Publication V*). Therefore overall run-time complexity of GDE3 is $O\left(G_{max} NP \log^{M-1} NP\right)$.

Compared to the earlier GDE versions, GDE3 improves the ability to handle MOOPs by giving a better distributed set of solutions and being less sensitive to the selection of control parameter values. GDE3 has been compared to NSGA-II and has been found to be at least comparable based on experimental results (*Publication IV*). As with GDE2 (and several MOEAs), loss of Pareto-optimal solutions is possible during the search.

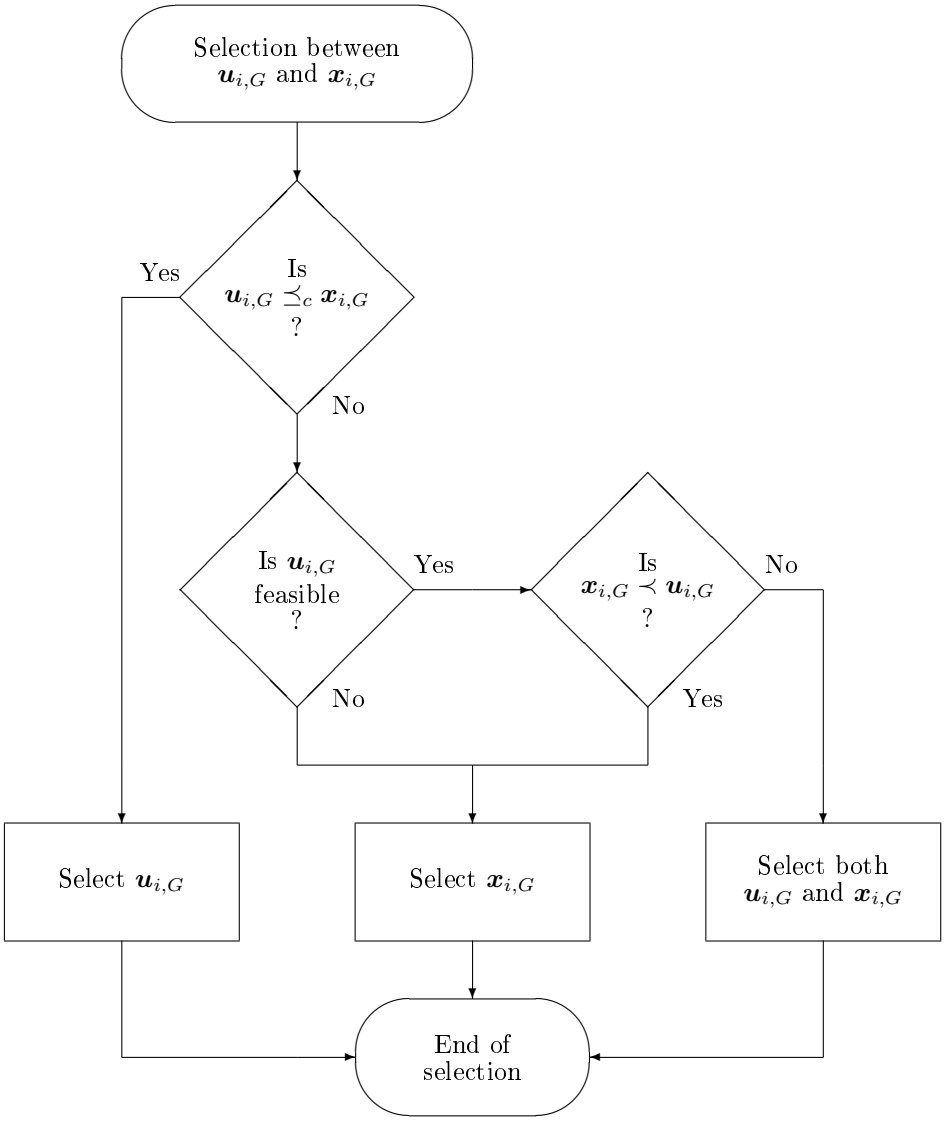


Figure 3.6: Operation for selection between the old vector $\boldsymbol{x}_{i,G}$ and trial vector $\boldsymbol{u}_{i,G}$ in GDE3.

$$
\begin{cases}
\text{Input : } D, G_{max}, NP \geq 4, F \in (0, 1+], CR \in [0, 1], \text{and initial bounds: } \boldsymbol{x}^{(lo)}, \boldsymbol{x}^{(hi)} \\
\text{Initialize : } \begin{cases} \forall i \leq NP \wedge \forall j \leq D : x_{j,i,0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)} \right) \\ i = \{1, 2, \ldots, NP\}, j = \{1, 2, \ldots, D\}, G = 0, n = 0, rand_j[0,1] \in [0,1], \end{cases} \\
\begin{cases}
\text{While } G < G_{max} \\
\quad \forall i \leq NP \begin{cases}
\text{Mutate and recombine:} \\
r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \text{randomly selected,} \\
\qquad \text{except mutually different and different from } i \\
j_{rand} \in \{1, 2, \ldots, D\}, \text{randomly selected for each } i \\
\forall j \leq D, u_{j,i,G} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \qquad \text{if} \quad rand_j[0,1) < CR \vee j == j_{rand} \\ x_{j,i,G} \quad \text{otherwise} \end{cases} \\
\text{Select :} \\
\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if} \quad \boldsymbol{u}_{i,G} \preceq_c \boldsymbol{x}_{i,G} \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} \\
\boxed{\begin{array}{l} \text{Set :} \\ \quad n = n + 1 \\ \quad \boldsymbol{x}_{NP+n,G+1} = \boldsymbol{u}_{i,G} \end{array} \quad \text{if} \begin{cases} \forall j : g_j(\boldsymbol{u}_{i,G}) \leq 0 \\ \wedge \\ \boldsymbol{x}_{i,G+1} == \boldsymbol{x}_{i,G} \\ \wedge \\ \boldsymbol{x}_{i,G} \not\prec \boldsymbol{u}_{i,G} \end{cases}}
\end{cases} \\
\quad \boxed{\begin{cases}
\text{While } n > 0 \\
\qquad \text{Select } \boldsymbol{x} \in \mathcal{P} = \{\boldsymbol{x}_{1,G+1}, \boldsymbol{x}_{2,G+1}, \ldots, \boldsymbol{x}_{NP+n,G+1}\} : \\
\qquad\quad \begin{cases} \boldsymbol{x} \text{ belongs to the last non-dominated set of } \mathcal{P} \\ \wedge \\ \boldsymbol{x} \text{ is the most crowded in the last non-dominated set} \end{cases} \\
\qquad \text{Remove } \boldsymbol{x} \text{ from } \mathcal{P} \\
\qquad n = n - 1
\end{cases}} \\
\quad G = G + 1
\end{cases}
\end{cases}
$$

Figure 3.7: The GDE3 algorithm

3.3.1 Diversity Preservation for Bi-Objective Problems

The diversity preservation technique used in GDE3 is an improved version of the approach used in NSGA-II. In NSGA-II, the crowding distance values are calculated once for all the members of a non-dominated set. Members having the smallest crowding distance values are removed without taking into account how removal of a member will affect the crowding distance value of its neighbors. The outcome is that the diversity of the remaining members after removal is non-optimal.

The diversity preservation operation in GDE3 removes the most crowded members from a non-dominated set one by one and updates the crowding distance value of the remaining members after each removal (*Publication V*). A straightforward approach would have time complexity class $O\left(MNP^2\right)$. A more sophisticated algorithm having the same time complexity class as NSGA-II, $O\left(MNP \log NP\right)$, has been proposed in *Publication V*. This approach was implemented when GDE3 was originally introduced but published later in *Publication V* because detailed description of the diversity preservation technique was too extensive for *Publication IV*.

The final populations for the ZDT and DTLZ problems are shown in Figures 3.8 and 3.9. The distribution obtained for the ZDT problems is good, *i.e.*, points are uniformly distributed. The distribution of points for the DTLZ problems is still far from uniform (except for DTLZ5, which has a curve shaped Pareto-optimal front).

Publication V shows that the crowding distance does not estimate crowding properly when the number of objectives is more than two. This is a significant observation since NSGA-II is the most popular MOEA and crowding distance has been used in many studies, some of which are mentioned in *Publication V*. The crowding distance has also been used in cases with more than two objectives, *e.g*, in [4, 116]. It should be noted that several multi-objective DE approaches mentioned in Section 2.3.2 use the crowding distance and therefore do not provide good diversity when the number of objectives is more than two.

3.3.2 Diversity Preservation for Many-Objective Problems

As noted in the previous section, the diversity preservation technique in GDE3 does not generally provide good diversity when the number of objectives is more than two. Therefore, a new efficient diversity preservation technique was needed for many-objective problems. The term many-objective is used in the EMO literature when the number of objectives is more than two or three (no consensus exists regarding the number of objectives required for a problem to be deemed many-objective). In this work, many-objective refers to a situation when the number of objectives is three or more.

A pruning method intended to be both effective and relatively fast was proposed in *Publication VII*. The basic idea of the method is to eliminate the most crowded members of a non-dominated set one by one, and update the crowding information of the remaining members after each removal. One can notice that the main idea in the approach in *Publication VII* is similar to the approach in *Publication V* but the diversity metric calculation has been modified as an improvement.

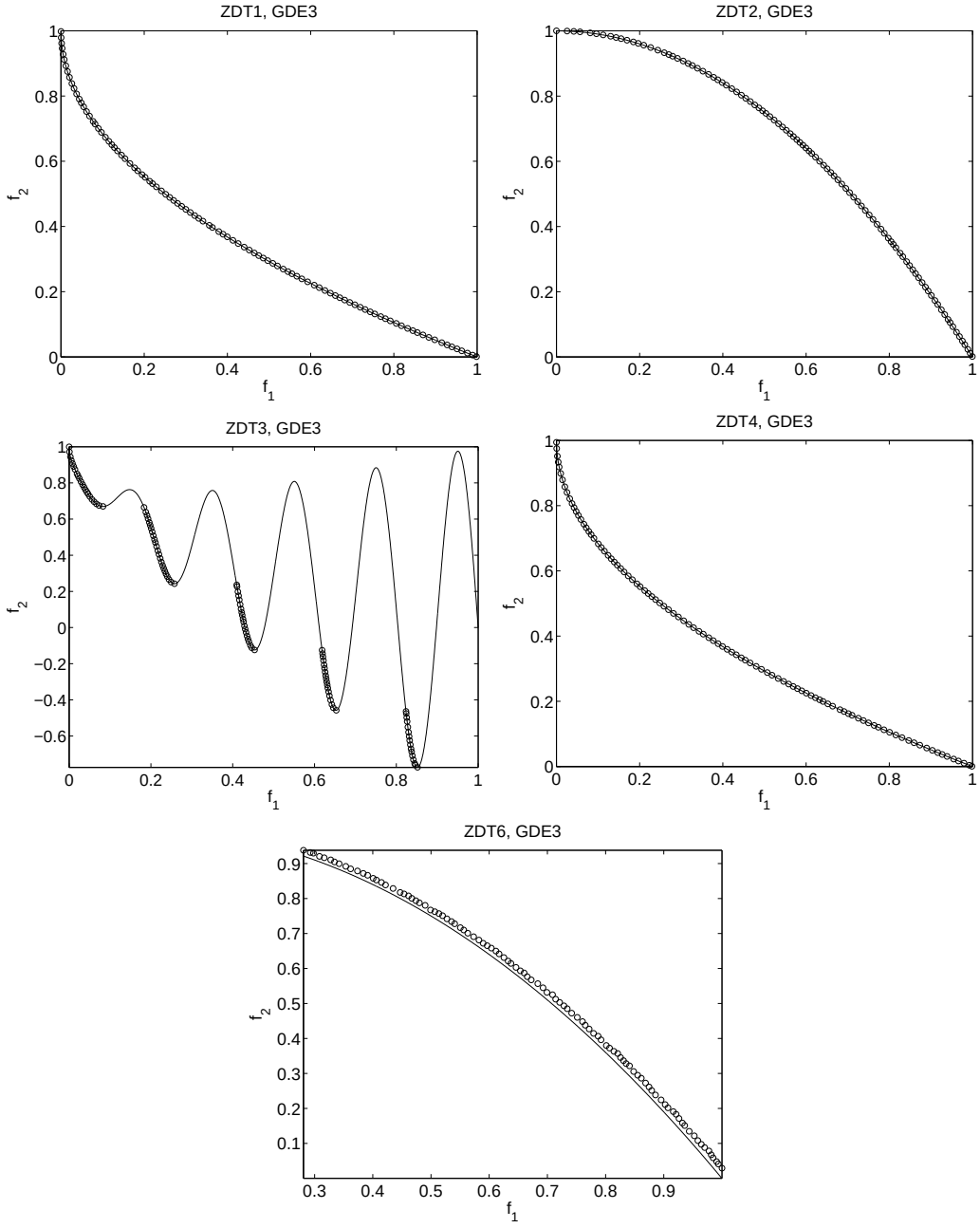


Figure 3.8: Results for the ZDT problems using GDE3.

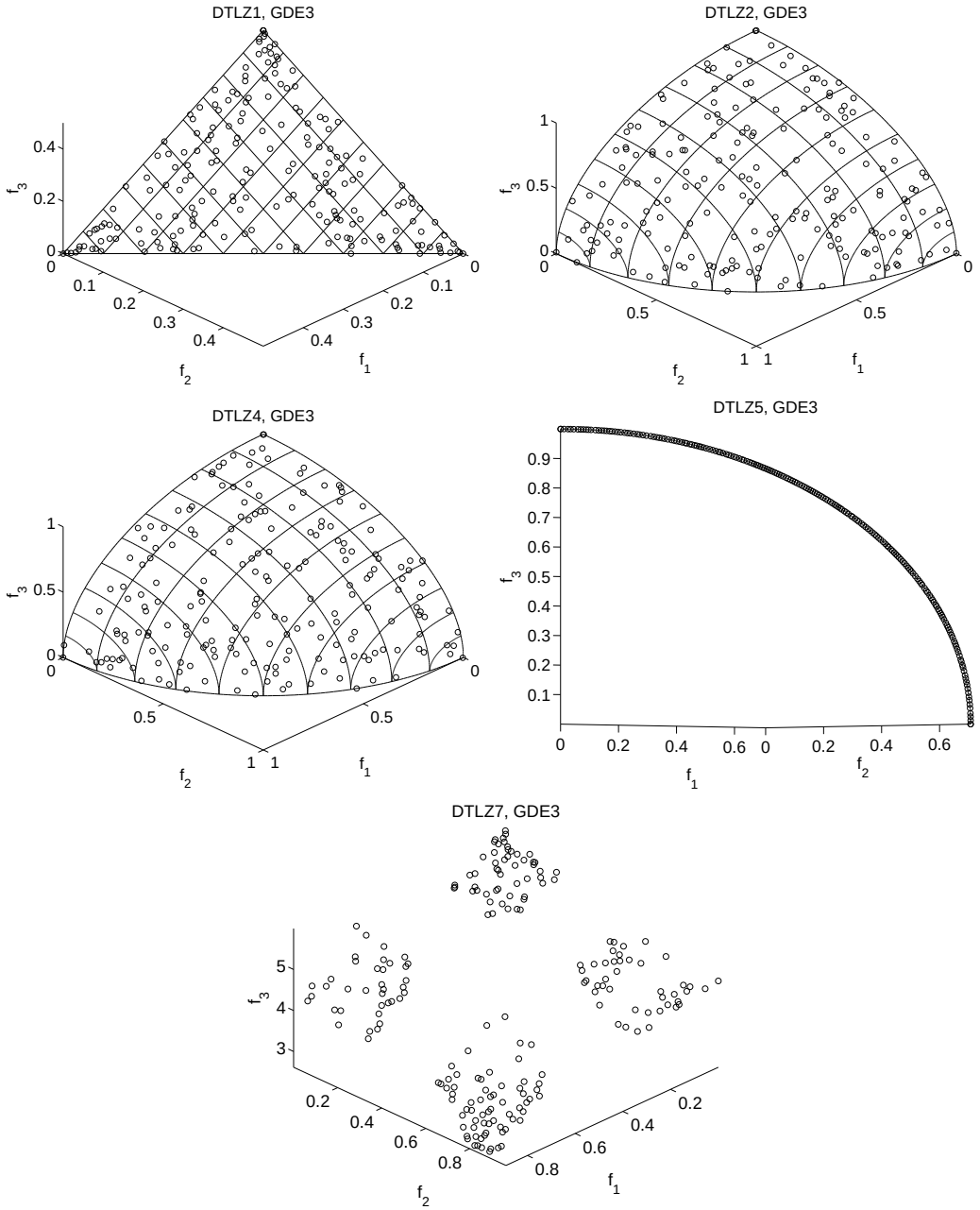


Figure 3.9: Results for the DTLZ problems using GDE3.

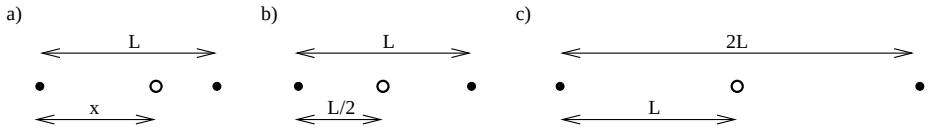


Figure 3.10: Three examples of a solution (circle) between the two nearest neighbors (dots) demonstrating the vicinity distance.

The crowding estimation is based on the nearest neighbors of solution candidates in the objective space and an efficient search method to find the k nearest neighbors (k-NNs). Two crowding estimation techniques were proposed: one that always uses two nearest neighbors for the crowding estimation (2-NN) and one that uses as many neighbors as there are objectives (M-NN). Distances are measured in a normalized objective space using the Euclidean distance measure.

In the first crowding estimation approach, the solution candidates are sorted using distance to the nearest neighbor as a primary sorting key and distance to the second nearest neighbor as the secondary sorting key. The solution candidate that has the smallest distance to the neighbors is considered the most crowded.

In the second crowding estimation approach, distances to M nearest neighbors are multiplied and the solution candidate having the smallest product is considered the most crowded one. The product of distances, called here *vicinity distance*, is formally:

$$d_v = \prod_{i=1}^{M} L_p^{NN_i} \ , \tag{3.4}$$

where $L_p^{NN_i}$ is distance to ith nearest neighbor according to the L_p distance metric [4]. The vicinity distance is a very simple measure but it already contains information about vicinity and location. This is illustrated in Figure 3.10 with two nearest neighbors in three different cases. In all the cases, a solution (circle) is located on a line between two nearest neighbors (filled dots). In the first case a) the solution is closer to the right neighbor than the left neighbor and the vicinity distance will be $d_v = x\,(L - x) = xL - x^2$. When the solution moves between the two nearest neighbors, its vicinity distance value will change along a paraboloid curve and have the maximum value $d_v = L^2/4$ when the solution is in the middle of the two nearest neighbors as shown in the b). If the solution remains in the middle of the two nearest neighbors but the distance to the nearest neighbors doubles as in case c) compared to b), then the vicinity distance will increase to value $d_v = L^2$. Therefore, the value of the vicinity distance is dependent on both vicinity and location. It will be large for a solution having a large distance to all neighboring solutions.

An efficient exact k-NN search technique used is the *equal-average nearest neighbor search (ENNS) algorithm* [58, 121]. ENNS projects data to a projection axis and uses the inequality between the projected values and the corresponding Euclidean distance to reduce the number of actual distance calculations. The speed of the proposed method is further increased by using a priority queue to store elements. Details of the algorithm

[4] The L_p metric calculates the distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ with the formula $L_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{M} |x_i - y_i|^p\right)^{1/p}$.

can be found in *Publication VII*, and a scalability study in [79]. The algorithm has been observed to provide good diversity and be relatively fast. When the number of objectives increases, the method gets slower but still appears to be faster than the other effective distance-based approaches used in MOEAs [70]. As mentioned in [79], several potential ways to speed up the new pruning method exist. For example, the exact nearest neighbor search itself could be enhanced. Currently, one projection axis is used in the ENNS method. It would be possible to select several projection axes to different directions and project data to them. This would increase the number of pre-calculations but presumably decrease the number of search steps in finding the nearest neighbors in ENNS. Besides projection values, variances of vectors could also be calculated to obtain a smaller searching area for the nearest neighbors [7]. Other improvements to ENNS have been proposed in [10,114].

A clustering method similar to the pruning method in *Publication VII*, has been proposed in [52]. The two methods were developed independently and the major difference between them is that the method in [52] uses an approximate nearest neighbor search whereas the pruning method in *Publication VII* performs an exact nearest neighbor search. The findings in [52] show that use of an approximate nearest neighbor search does not significantly reduce the obtained clustering result but gives notable speedup.

The diversity preservation technique used in GDE3 can be directly replaced with the diversity preservation technique intended for a large number of objectives presented in *Publication VII*. It has been noted in *Publication VII* that both 2-NN and M-NN crowding estimation techniques can be used in many-objective optimization and they provide similar results but the 2-NN distance measure is faster to calculate when M increases. The final populations for the ZDT and DTLZ problems solved using GDE3 with the 2-NN crowding estimation technique are shown in Figures 3.11 and 3.12. The distribution can be observed to be good for all the problems.

The numerical results in *Publication V* and *Publication VII* are directly comparable since the same control parameter values and computers have been used to produce the results. By looking at the spacing metric values with the ZDT problems in both papers, comparable performance can be noticed with the proposed methods ("Proposed" in *Publication V*; "GDE3, 2-NN" and "GDE3, M-NN" in *Publication VII*) [5]. When spacing metric values with the DTLZ problems are compared, a clear improvement can be observed, as expected based on Figures 3.9 and 3.12.

GDE3 with the 2-NN crowding estimation technique was one of the participants in a multi-objective optimization competition arranged at the 2007 IEEE Congress on Evolutionary Computation (CEC 2007). The task was to solve a set of multi-objective problems having from two to five objectives defined in [63] and based on the results reported in [84] GDE3 with the 2-NN crowding estimation technique received a winning entry nomination in the competition.

[5] Method names "Original" in *Publication V* and "GDE3, CD" in *Publication VII* refer to the same method.

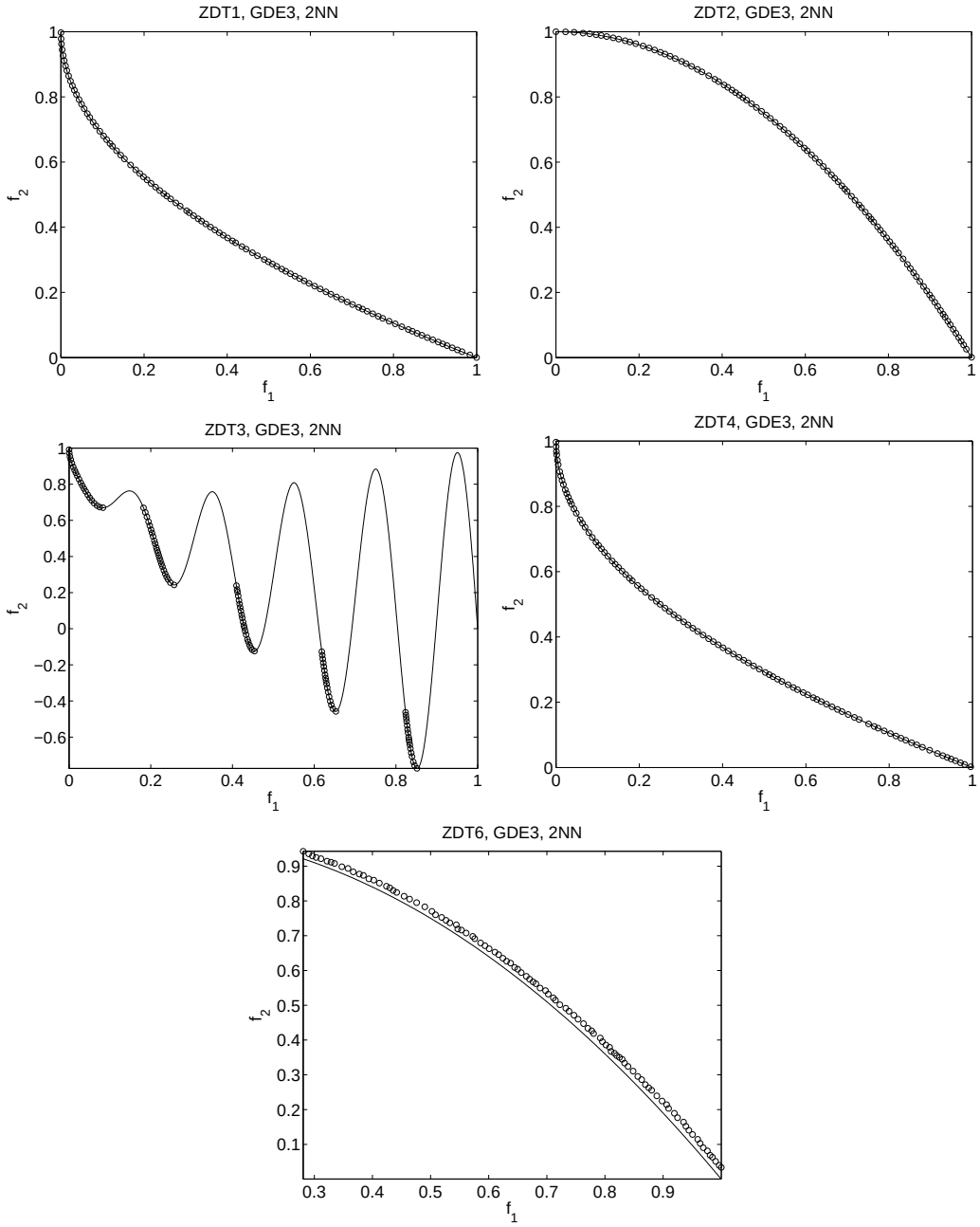


Figure 3.11: Results for the ZDT problems using GDE3 with the 2-NN crowding estimation technique.

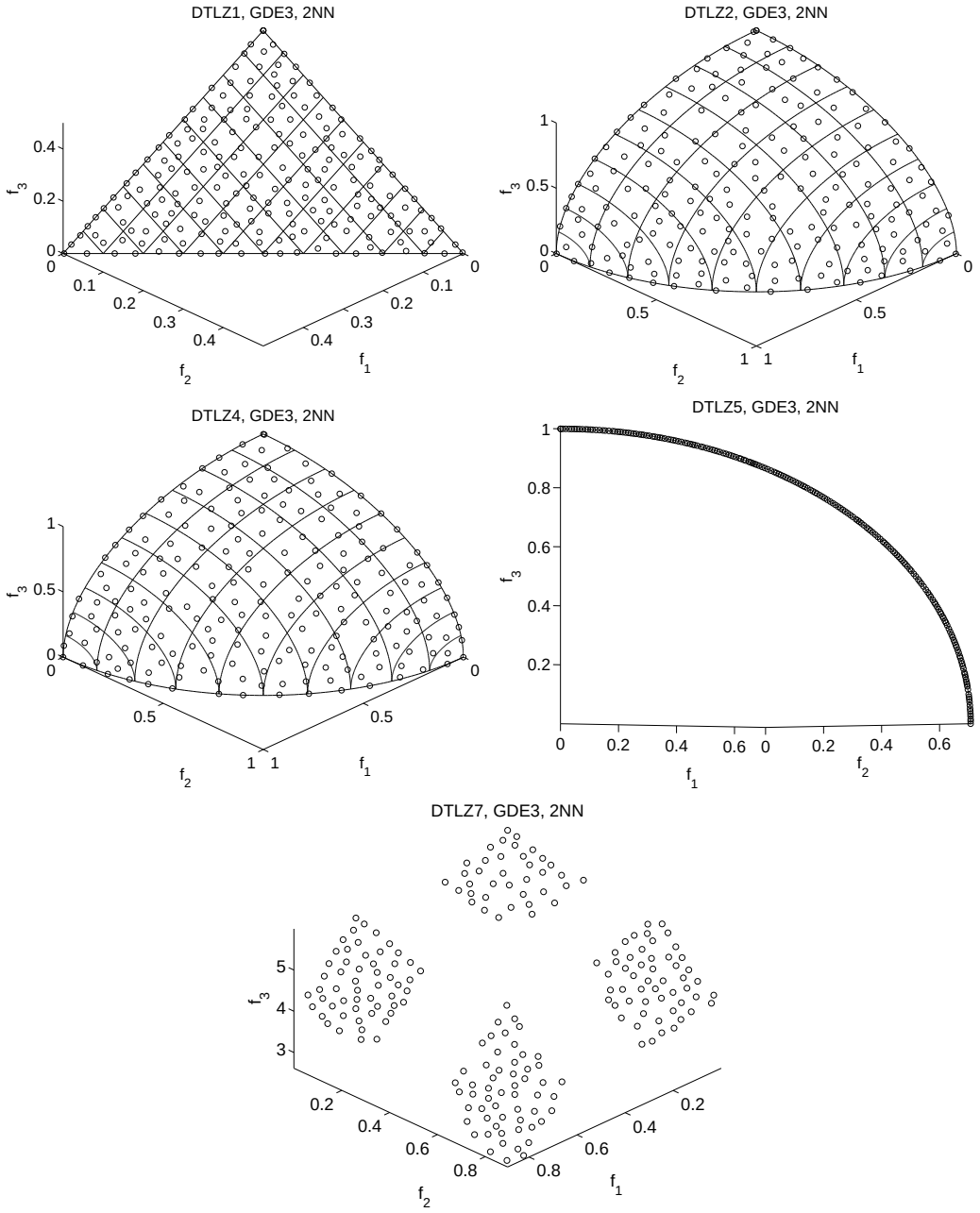


Figure 3.12: Results for the DTLZ problems using GDE3 with the 2-NN crowding estimation technique.

3.4 Numerical Comparison of the GDE Versions

Tests with different GDE development versions and test problems were repeated 100 times. The results are given in Tables 3.1 and 3.2. Unique non-dominated solutions are extracted from the final generation and the cardinality ($\aleph$) of this set presented. Closeness to the Pareto-optimal front is measured with the generational distance (GD). The diversity of the obtained solution is measured using the spacing (S) metric. Overall quality of the obtained non-dominated set is illustrated with the inverted generational distance (IGD).

Tables 3.1 and 3.2 show that performance has improved during GDE development versions. In particular, the obtained cardinality, diversity, and overall performance have improved.

3.5 Study of Control Parameter Values for GDE

The effect of control parameters CR and F has been studied with GDE1 and GDE3 in *Publication II* and *Publication VI*, respectively. It has not been clear how the control parameter values affect the results when there is no diversity preservation (*Publication II*) and when diversity preservation is used (*Publication VI*). Different control parameter values have been tested using bi-objective test problems and performance metrics described in [33, pp. 326–328, 338–360]. The problem set in *Publication VI* has been appended with a set of non-separable problems defined in [38]. The experiments were restricted to the two objective problems due to space limitation in the articles.

According to the diversity and cardinality measures in *Publication II* and *Publication VI*, GDE3 generally provides a better distribution with a larger number of non-dominated solutions than GDE1. The better diversity is also well visible when Figures 3.1 and 3.8 are compared. In *Publication VI*, it has been noted that GDE3 is able to produce a good Pareto front approximation with a larger number of non-dominated solutions with a wider range of different control parameter value combinations. The results of GDE3 (especially the number of non-dominated solutions) are also less sensitive to variation of the control parameter values compared to GDE1, *i.e.*, GDE3 appears to be more robust in terms of the control parameter value selection.

A preliminary parameter study with GDE2 has also been performed but the results have not been published. In the preliminary study, GDE2 was found to be sensitive to the control parameter values similarly to GDE1.

Different mutation factor F values were tried up to 3.0 [6] but the results suggest that a larger F value than usually used in single-objective optimization does not give any extra benefit in multi-objective optimization. As noted in *Publication II*, if the complexity of objectives differs greatly (*i.e.*, the objectives demand considerably different computational effort if solved individually), it is better to use a smaller CR value to prevent the population from converging to a single point on the Pareto-optimal front.

[6] F values larger than 2.0 have not been used in single-objective optimization [94]. A larger value was chosen for the experiments since the effect of F in multi-objective optimization was unclear. In preliminary F values up to 5.0 were also tried but noted as unnecessarily large.

Table 3.1: Mean and standard deviation values of cardinality ($\aleph$), generational distance (GD), spacing (S), and inverted generational distance (IGD) from 100 independent runs with the GDE versions and the ZDT test problems.

Problem	Method	$\aleph$	GD	S	IGD
ZDT1	GDE1	$9.8707e{+}01\pm 9.9234e-01$	$7.9401e{-}05\pm 8.9500e-06$	$1.1611e{-}02\pm 2.0445e-03$	$3.6562e{-}05\pm 6.9741e-06$
	GDE2	$9.9827e{+}01\pm 3.8060e-01$	$8.2341e{-}05\pm 1.0647e-05$	$6.7852e{-}03\pm 6.5228e-04$	$1.6252e{-}05\pm 5.4849e-07$
	GDE3	$1.0000e{+}02\pm 0.0000e+00$	$2.0780e{-}05\pm 5.5679e-06$	$6.3803e{-}03\pm 5.0843e-04$	$1.7026e{-}05\pm 6.7607e-07$
	GDE3, 2NN	$1.0000e{+}02\pm 0.0000e+00$	$1.8424e{-}05\pm 7.6065e-06$	$2.8465e{-}03\pm 2.7662e-04$	$1.4088e{-}05\pm 1.5334e-07$
	GDE3, MNN	$1.0000e{+}02\pm 0.0000e+00$	$1.7886e{-}05\pm 5.8128e-06$	$2.7015e{-}03\pm 2.6961e-04$	$1.4016e{-}05\pm 1.2230e-07$
ZDT2	GDE1	$1.7969e{+}01\pm 4.3753e+00$	$1.8470e{-}04\pm 2.5585e-05$	$7.7930e{-}02\pm 2.8647e-02$	$2.5535e{-}04\pm 8.8487e-05$
	GDE2	$9.8887e{+}01\pm 8.6454e-01$	$1.2907e{-}04\pm 1.4953e-05$	$7.0592e{-}03\pm 6.4203e-04$	$1.7612e{-}05\pm 7.5518e-07$
	GDE3	$1.0000e{+}02\pm 0.0000e+00$	$3.7610e{-}05\pm 2.2664e-05$	$7.4705e{-}03\pm 1.1947e-02$	$1.7718e{-}05\pm 1.2176e-06$
	GDE3, 2NN	$1.0000e{+}02\pm 0.0000e+00$	$3.3107e{-}05\pm 2.1824e-05$	$4.5017e{-}03\pm 1.5798e-02$	$1.4469e{-}05\pm 1.0701e-06$
	GDE3, MNN	$1.0000e{+}02\pm 0.0000e+00$	$3.2125e{-}05\pm 2.1209e-05$	$4.1592e{-}03\pm 1.4134e-02$	$1.4272e{-}05\pm 9.4162e-07$
ZDT3	GDE1	$7.7350e{+}01\pm 4.6458e+00$	$7.5621e{-}05\pm 2.4994e-05$	$1.5244e{-}02\pm 3.9025e-03$	$6.0599e{-}05\pm 1.6036e-05$
	GDE2	$4.3102e{+}01\pm 3.5450e+00$	$1.4858e{-}04\pm 8.1965e-05$	$1.6692e{-}02\pm 3.2548e-03$	$4.2632e{-}05\pm 6.6759e-06$
	GDE3	$1.0000e{+}02\pm 0.0000e+00$	$1.7702e{-}05\pm 4.0392e-06$	$4.2699e{-}03\pm 3.8531e-04$	$1.1928e{-}05\pm 3.6297e-07$
	GDE3, 2NN	$1.0000e{+}02\pm 0.0000e+00$	$1.6210e{-}05\pm 3.5637e-06$	$2.2963e{-}03\pm 2.9270e-04$	$1.0393e{-}05\pm 1.4318e-07$
	GDE3, MNN	$1.0000e{+}02\pm 0.0000e+00$	$1.6612e{-}05\pm 4.1601e-06$	$2.1646e{-}03\pm 2.2485e-04$	$1.0343e{-}05\pm 1.1664e-07$
ZDT4	GDE1	$7.8798e{+}01\pm 7.3290e+00$	$3.7243e{-}04\pm 1.7616e-04$	$1.3469e{-}02\pm 2.6249e-03$	$4.0105e{-}05\pm 7.8157e-06$
	GDE2	$8.4081e{+}01\pm 7.1965e+00$	$3.6964e{-}04\pm 1.6712e-04$	$9.6220e{-}03\pm 1.5388e-03$	$2.3989e{-}05\pm 3.3808e-06$
	GDE3	$1.0000e{+}02\pm 0.0000e+00$	$3.9580e{-}06\pm 3.0481e-06$	$6.1654e{-}03\pm 6.0282e-04$	$2.7287e{-}05\pm 6.5904e-05$
	GDE3, 2NN	$1.0000e{+}02\pm 0.0000e+00$	$5.1378e{-}06\pm 4.0741e-06$	$3.3962e{-}03\pm 1.5001e-03$	$2.6092e{-}05\pm 6.9919e-05$
	GDE3, MNN	$1.0000e{+}02\pm 0.0000e+00$	$4.2760e{-}06\pm 3.6195e-06$	$3.0126e{-}03\pm 8.6901e-04$	$2.6295e{-}05\pm 8.5056e-05$
ZDT6	GDE1	$3.5390e{+}01\pm 9.6775e+00$	$1.4157e{-}02\pm 2.1238e-03$	$3.3855e{-}02\pm 1.5365e-02$	$3.5261e{-}04\pm 3.5452e-05$
	GDE2	$6.2889e{+}01\pm 4.2448e+00$	$1.0651e{-}02\pm 9.0784e-04$	$1.1567e{-}02\pm 1.4900e-03$	$3.1265e{-}04\pm 2.2723e-05$
	GDE3	$1.0000e{+}02\pm 0.0000e+00$	$1.4555e{-}03\pm 1.4046e-04$	$5.9064e{-}03\pm 5.1987e-04$	$6.6336e{-}05\pm 7.0549e-05$
	GDE3, 2NN	$1.0000e{+}02\pm 0.0000e+00$	$1.4894e{-}03\pm 1.7214e-04$	$3.9464e{-}03\pm 3.7292e-04$	$5.7084e{-}05\pm 6.3232e-06$
	GDE3, MNN	$1.0000e{+}02\pm 0.0000e+00$	$1.5390e{-}03\pm 6.0437e-04$	$3.7544e{-}03\pm 3.7457e-04$	$5.6755e{-}05\pm 5.6651e-06$

Table 3.2: Mean and standard deviation values of cardinality ($\aleph$), generational distance (GD), spacing (S), and inverted generational distance (IGD) from 100 independent runs with the GDE versions and the DTLZ test problems.

Problem	Method	$\aleph$	GD	S	IGD
DTLZ1	GDE1	$2.7414e{+}01\pm 5.5310e+00$	$9.5081e{-}01\pm 2.4820e-01$	$1.4143e{-}01\pm 3.8488e-02$	$9.9790e{-}04\pm 3.1580e-04$
	GDE2	$1.3240e{+}01\pm 3.3639e+00$	$2.5983e{+}01\pm 6.5663e+00$	$2.0326e{-}01\pm 7.2324e-02$	$1.0725e{-}02\pm 4.2351e-03$
	GDE3	$2.0000e{+}02\pm 0.0000e+00$	$6.0050e{-}03\pm 1.0818e-02$	$2.9482e{-}02\pm 1.6161e-03$	$8.5743e{-}05\pm 1.2220e-04$
	GDE3, 2NN	$2.0000e{+}02\pm 0.0000e+00$	$6.2010e{-}03\pm 1.0177e-02$	$1.4671e{-}02\pm 8.8322e-04$	$8.7355e{-}05\pm 1.2045e-04$
	GDE3, MNN	$2.0000e{+}02\pm 0.0000e+00$	$4.4849e{-}03\pm 9.2179e-03$	$1.5581e{-}02\pm 8.3106e-04$	$6.6809e{-}05\pm 1.0856e-04$
DTLZ2	GDE1	$2.0000e{+}02\pm 0.0000e+00$	$4.9471e{-}05\pm 2.0035e-06$	$4.3724e{-}02\pm 2.6125e-03$	$5.3107e{-}05\pm 2.0176e-06$
	GDE2	$2.0000e{+}02\pm 0.0000e+00$	$5.5430e{-}05\pm 1.9509e-06$	$3.4031e{-}02\pm 1.7304e-03$	$4.8581e{-}05\pm 1.4472e-06$
	GDE3	$2.0000e{+}02\pm 0.0000e+00$	$5.5634e{-}05\pm 1.8877e-06$	$3.5472e{-}02\pm 1.9498e-03$	$4.9266e{-}05\pm 1.5254e-06$
	GDE3, 2NN	$2.0000e{+}02\pm 0.0000e+00$	$5.7255e{-}05\pm 2.4149e-06$	$1.9975e{-}02\pm 1.1090e-03$	$3.9907e{-}05\pm 4.3232e-07$
	GDE3, MNN	$2.0000e{+}02\pm 0.0000e+00$	$5.8156e{-}05\pm 2.7023e-06$	$2.0868e{-}02\pm 1.0968e-03$	$3.9451e{-}05\pm 2.8485e-07$
DTLZ4	GDE1	$6.2755e{+}00\pm 1.9519e+00$	$5.0198e{-}04\pm 1.0986e-04$	$3.5275e{-}01\pm 1.7676e-01$	$5.1008e{-}04\pm 1.4430e-04$
	GDE2	$2.0000e{+}02\pm 0.0000e+00$	$5.6168e{-}05\pm 2.0555e-06$	$3.4457e{-}02\pm 1.9792e-03$	$4.8845e{-}05\pm 1.3942e-06$
	GDE3	$2.0000e{+}02\pm 0.0000e+00$	$5.6301e{-}05\pm 2.0287e-06$	$3.5208e{-}02\pm 1.9135e-03$	$5.0081e{-}05\pm 7.1383e-06$
	GDE3, 2NN	$2.0000e{+}02\pm 0.0000e+00$	$5.9618e{-}05\pm 2.1462e-06$	$1.8732e{-}02\pm 2.0320e-03$	$3.9958e{-}05\pm 4.4046e-07$
	GDE3, MNN	$2.0000e{+}02\pm 0.0000e+00$	$5.9713e{-}05\pm 2.2476e-06$	$1.9779e{-}02\pm 1.4856e-03$	$3.9543e{-}05\pm 3.1801e-07$
DTLZ5	GDE1	$2.0000e{+}02\pm 0.0000e+00$	$7.7555e{-}07\pm 5.3369e-08$	$8.0103e{-}03\pm 7.6584e-04$	$1.7771e{-}05\pm 1.7985e-06$
	GDE2	$2.0000e{+}02\pm 0.0000e+00$	$7.7721e{-}07\pm 5.2233e-08$	$4.9175e{-}03\pm 3.3362e-04$	$9.0884e{-}06\pm 2.5734e-07$
	GDE3	$2.0000e{+}02\pm 0.0000e+00$	$7.8519e{-}07\pm 6.5487e-08$	$6.4111e{-}03\pm 3.8961e-04$	$1.0805e{-}05\pm 4.3664e-07$
	GDE3, 2NN	$2.0000e{+}02\pm 0.0000e+00$	$7.7613e{-}07\pm 5.1540e-08$	$3.1877e{-}03\pm 2.0126e-04$	$7.7837e{-}06\pm 6.6070e-08$
	GDE3, MNN	$2.0000e{+}02\pm 0.0000e+00$	$7.9339e{-}07\pm 6.0149e-08$	$3.0482e{-}03\pm 2.0087e-04$	$7.7370e{-}06\pm 5.8958e-08$
DTLZ7	GDE1	$1.9690e{+}02\pm 1.8599e+00$	$2.9472e{-}04\pm 1.1981e-04$	$3.9305e{-}02\pm 6.0968e-03$	$3.3187e{-}04\pm 5.6861e-05$
	GDE2	$1.1076e{+}02\pm 4.5928e+00$	$3.6347e{-}03\pm 1.0600e-03$	$4.7638e{-}02\pm 6.1964e-03$	$1.9114e{-}04\pm 1.7906e-05$
	GDE3	$2.0000e{+}02\pm 0.0000e+00$	$1.2080e{-}03\pm 3.8379e-04$	$3.0603e{-}02\pm 3.3204e-03$	$1.2521e{-}04\pm 1.0607e-05$
	GDE3, 2NN	$2.0000e{+}02\pm 0.0000e+00$	$8.5639e{-}04\pm 1.6450e-04$	$1.2088e{-}02\pm 1.3598e-03$	$8.8440e{-}05\pm 1.0817e-06$
	GDE3, MNN	$2.0000e{+}02\pm 0.0000e+00$	$8.9192e{-}04\pm 1.0657e-04$	$1.2710e{-}02\pm 9.9847e-04$	$8.9277e{-}05\pm 1.0234e-06$

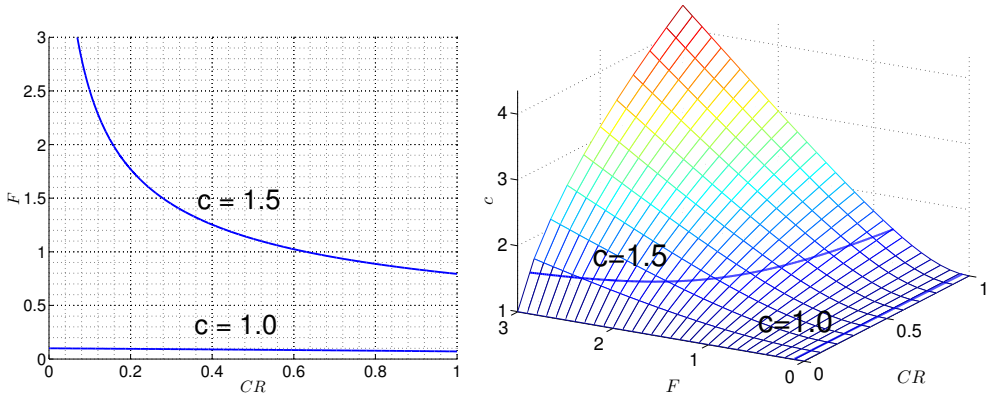


Figure 3.13: Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$ in the $CR - F$ control parameter space.

An inverse relationship between the CR and F values was observed from the results in *Publication II* and *Publication VI*, *i.e.*, a larger F value can be used with a small CR value than with a large CR value, and this relationship is non-linear. An explanation for this was found from theoretical analysis of the single-objective DE algorithm. A formula for the relationship between the control parameters of DE and the evolution/development of the population variance/standard deviation has been presented in [148].

The change of the population standard deviation between successive generations due to the crossover and mutation operations is denoted with c and its value is calculated as:

$$c = \sqrt{2F^2CR - 2CR/NP + CR^2/NP + 1}\ . \tag{3.5}$$

When $c < 1$, the crossover and mutation operations decrease the population standard deviation. When $c = 1$, the standard deviation does not change, and when $c > 1$, the standard deviation increases. Since the selection operation of an EA usually decreases the population standard deviation, $c > 1$ is recommended in order to prevent premature convergence. On the other hand, if c is too large, the search process proceeds reliably, but too slowly. In *Publication II* and *Publication VI* it has been observed that $c = 1.5$ is a suitable but not strict upper limit. This limit has been noticed also with single-objective problems [94]. When the size of the population is relatively large (*e.g.*, $NP > 50$), the value of c depends mainly on the values of CR and F. Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$ are shown in Figure 3.13.

Since experiments in *Publication II* and *Publication VI* are limited to bi-objective test problems, similar experiments are repeated here with the DTLZ test problems varying the number of objectives (M) from two to five and the population size according to formula $NP = (M - 1) * 100$. Results are shown in Figures 3.14–3.18 as surfaces in the control parameter space. A similar non-linear relation between the CR and F values can also be noticed here with most of the problems. One should note that the formula in 3.5 does not take into account the effect of the problem characteristics on the performance metric values. Thus, performance metric curves cannot be expected to follow formula 3.5 exactly.

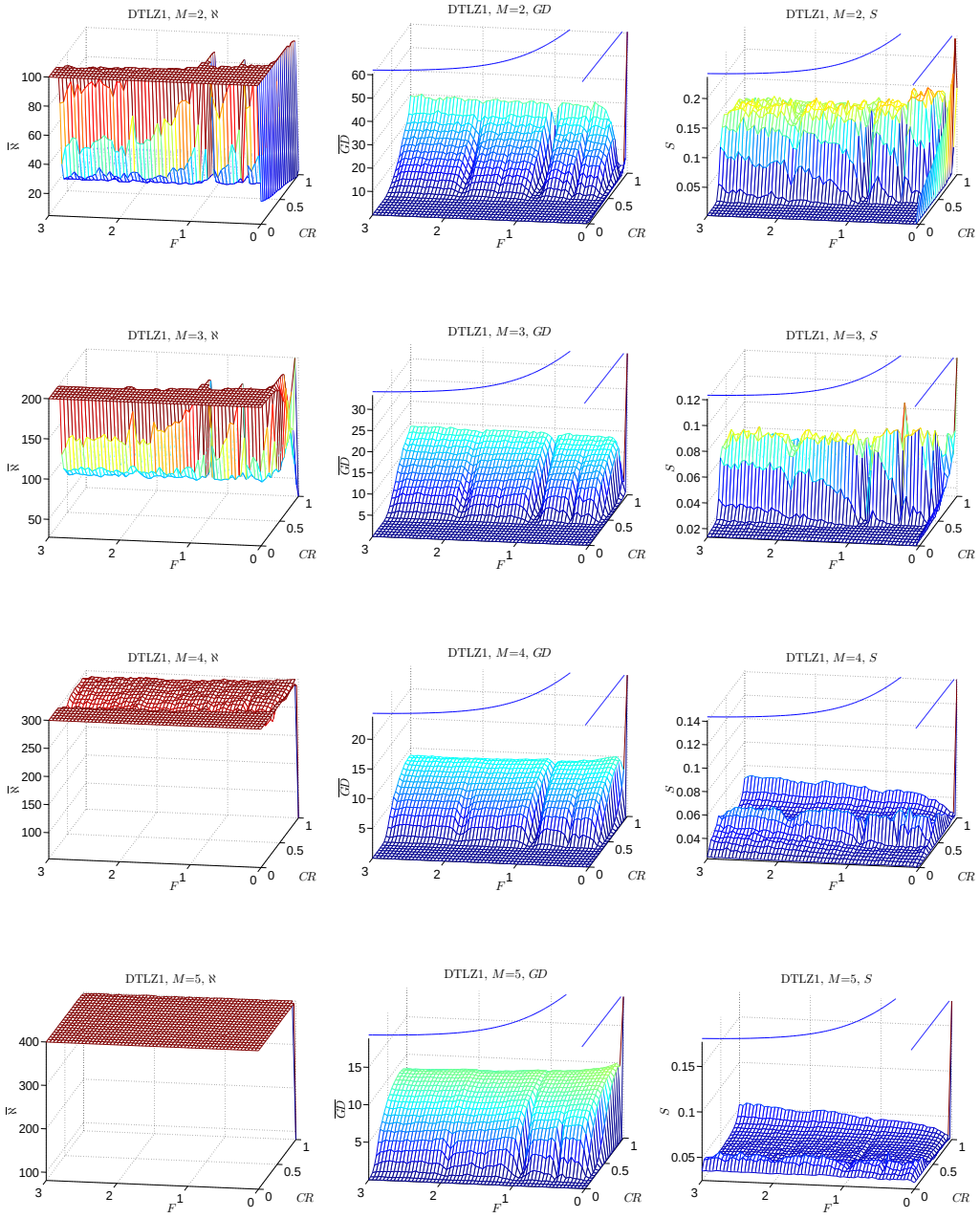


Figure 3.14: Mean values of the number of non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for DTLZ1 with 2–5 objectives shown as surfaces in the control parameter space.

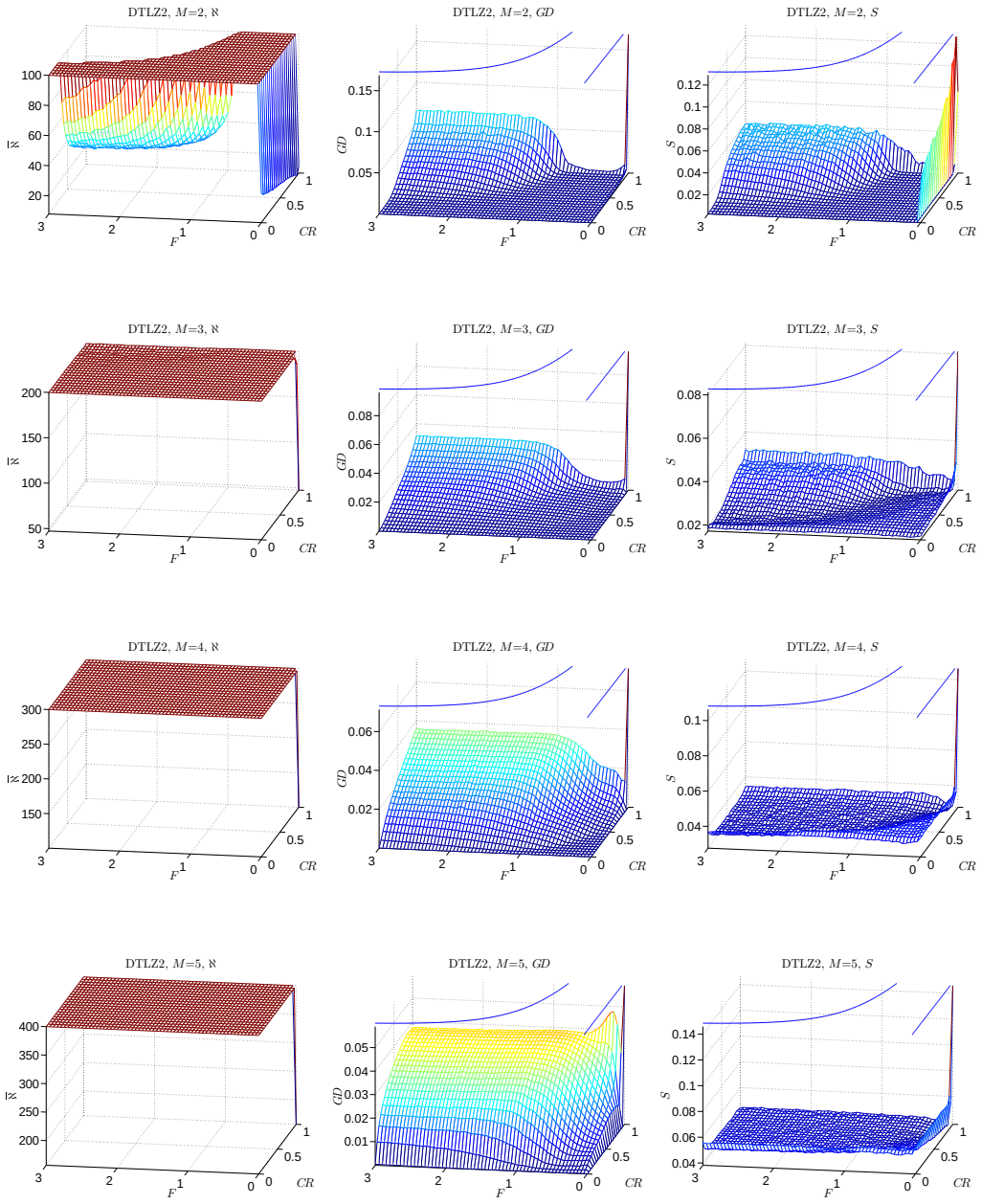


Figure 3.15: Mean values of the number of non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for DTLZ2 with 2–5 objectives shown as surfaces in the control parameter space.

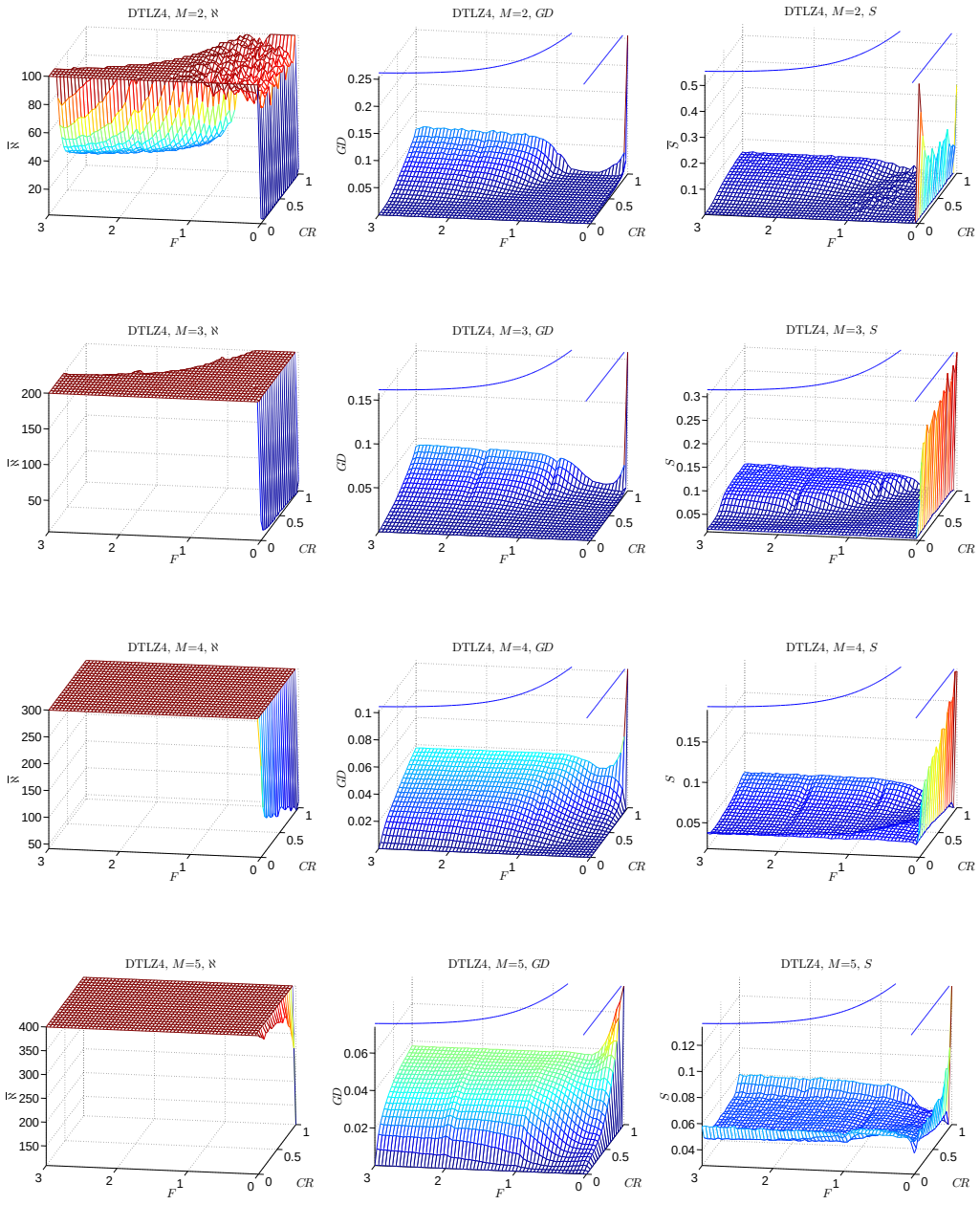


Figure 3.16: Mean values of the number of non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for DTLZ4 with 2–5 objectives shown as surfaces in the control parameter space.

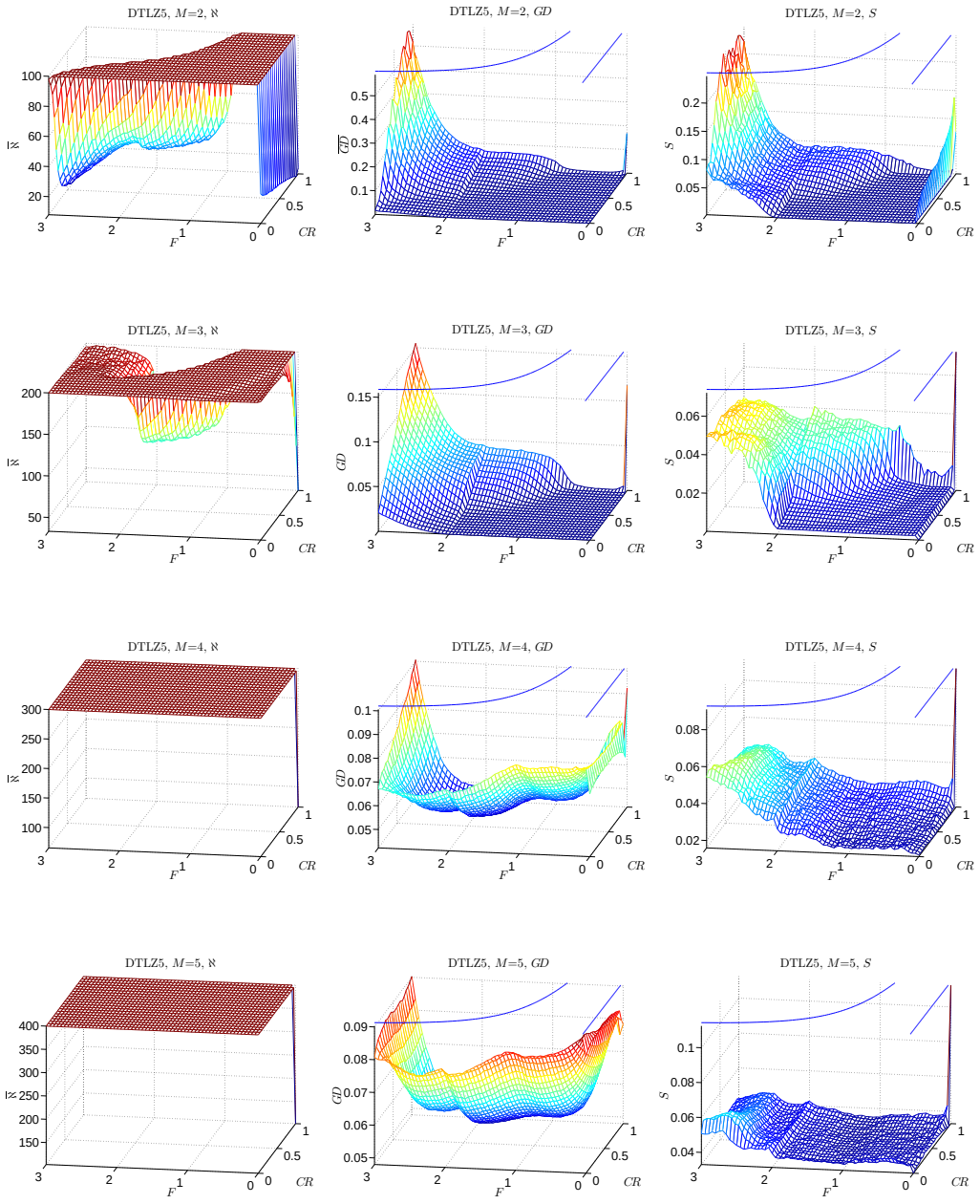


Figure 3.17: Mean values of the number of non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for DTLZ5 with 2–5 objectives shown as surfaces in the control parameter space.

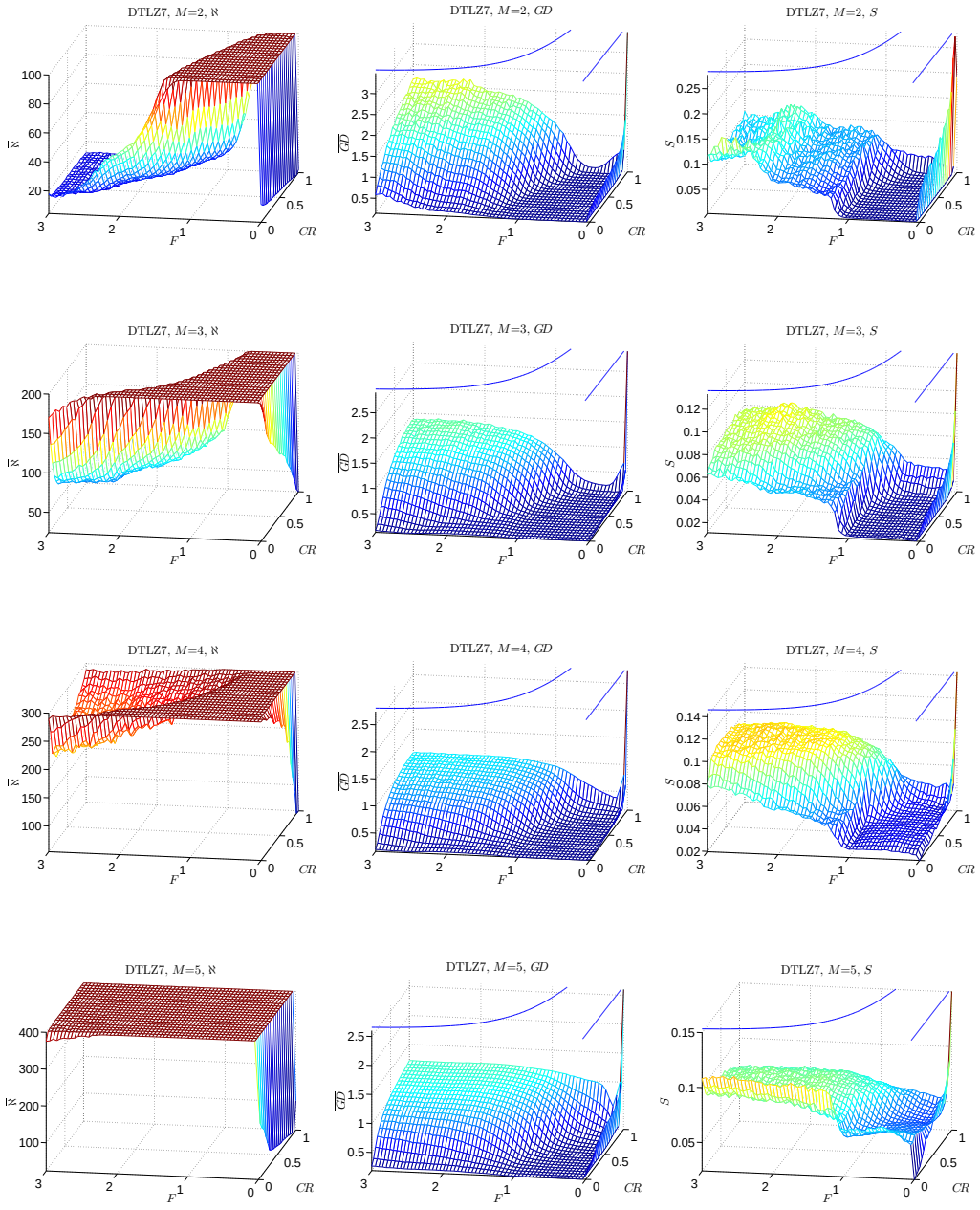


Figure 3.18: Mean values of the number of non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for DTLZ7 with 2–5 objectives shown as surfaces in the control parameter space.

Observations in *Publication II* and *Publication VI* and Figures 3.14–3.18 suggest that the theory for the population standard deviation between successive generations in Equation 3.5 is applicable also in the case of multi-objective DE and provides a way to select good control parameter value combinations. This is natural, since single-objective optimization can be seen as a special case of multi-objective optimization, and a large c means a slow but reliable search, while a small c means the opposite. Based on the results in *Publication II* and *Publication VI* it is advisable to select values for CR and F satisfying the condition $1.0 < c < 1.5+$ (*i.e.*, the upper limit is not strict). Values of CR and F can be fixed according to problem characteristics (as advised in Section 2.3.1) and to satisfy the condition above.

The observed results were good with small CR and F values that also mean a low c value (close to 1.0). One reason for the good performance with small control parameter values is that the test problems had conflicting objectives, which reduced overall selection pressure and prevented premature convergence. Another reason is that most of the problems had relatively easy objective functions to solve, leading to a faster convergence with a small F, while a bigger F might be needed for more difficult functions with several local optima and/or if NP is small.

3.6 Constrained Optimization with GDE

Above, the performance of GDE versions has been described only with unconstrained problems. However, the GDE versions include in their definition also a constraint handling approach which is the same in all the versions. This constraint handling approach was first introduced and evaluated for single-objective optimization with DE in [91] and later extended into multi-objective optimization with GDE.

In [82], a small set of mechanical design problems including several constraints was solved using GDE1. Good estimations of the Pareto-optimal front were obtained. The extent of the approximation sets was measured and found to be comparable to the results with NSGA-II. GDE1 was also used to solve a given set of constrained single-objective optimization problems in the CEC 2006 Special Session on Constrained Real-Parameter Optimization [83]. GDE1 was able to solve almost all the problems in a given maximum number of solution candidate evaluations. A better solution than previously known was found for some problems. It was also demonstrated that GDE needs a lower number of function evaluations than required if all the constraints are to be evaluated (as the case of several other constraint handling techniques).

In *Publication IV*, the ability of GDE versions to handle several constraints and different types of decision variables is demonstrated using a bi-objective spring design problem [36, 75]. GDE versions use real-coded variables, which are converted into corresponding actual variable types before evaluation of the objective and constraint functions. In the spring design problem, the problem is to design a helical compression spring which has a minimum volume and minimal stress. The objective functions are non-linear and the problem has three decision variables: the number of spring coils x_1 (integer), the wire diameter x_2 (discrete having 42 non-equispaced values), and the mean coil diameter x_3 (real). In addition to the boundary constraints, the problem has eight inequality constraints most of which are non-linear. A formal description of the problem is given in Appendix I.

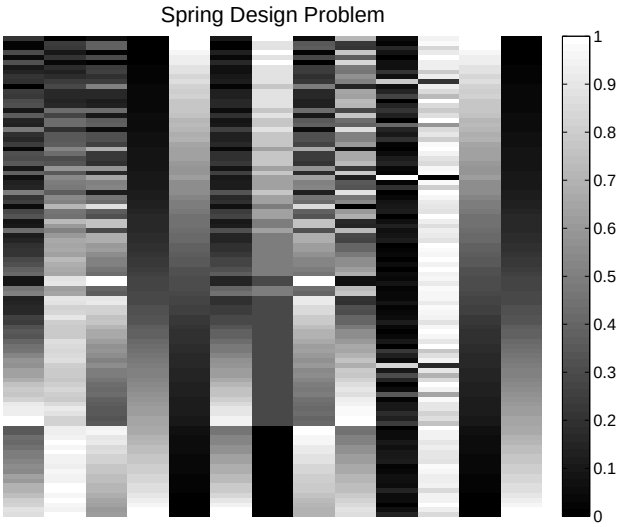


Figure 3.19: Heatmap of the spring design problem solved with GDE3. Columns 1–3 correspond to decision variables, columns 4–5 correspond to objectives, and columns 6–13 correspond to constraint functions. Rows correspond to individuals.

Non-dominated points extracted from the final population of the different GDE versions and NSGA-II are shown in Figure 2 of *Publication IV*. The size of the population and the number of generations were both 100. The number of function evaluations needed for the GDE versions are reported in Table 2 of *Publication IV*. It can be observed that the constraint handling approach used in the GDE versions reduce the actual number of function evaluations. It can also be noted that GDE2 and GDE3 evaluate objectives equally many times but GDE1 evaluates the first objective more often than the second objective. The reason for this is that possible incomparability between feasible target and trial vectors needs to be inspected in GDE2 and GDE3.

The spring design problem solved with GDE3 is shown as a heatmap (*cf.* Section 2.2.6) in Figure 3.19. The value ranges of decision variables, objectives, and constraint functions have been normalized to ease observation.

The performance of GDE3 is compared with NSGA-II in Figures 3.20–3.23 using a set of common bi-objective test problems given in [33, pp. 362–367] and formally defined in Appendix I.

The first problem is known as BNH and it has two decision variables and constraints. Results for this problem are shown in Figure 3.20. The population size and number of generations were 100 and 150, respectively. The control parameter values were $CR = 0.4$ and $F = 0.3$ for GDE3 [7], and $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 10$, and $\eta_m = 20$ for NSGA-II [8].

[7] The control parameter values for GDE3 were selected experimentally by trying out a few parameter value combinations and choosing the most promising, which were kept the same for all the constrained test problems.

[8] The control parameter values, the size of the population, and the number of generations of NSGA-II for the problems were obtained with the program code from [76].

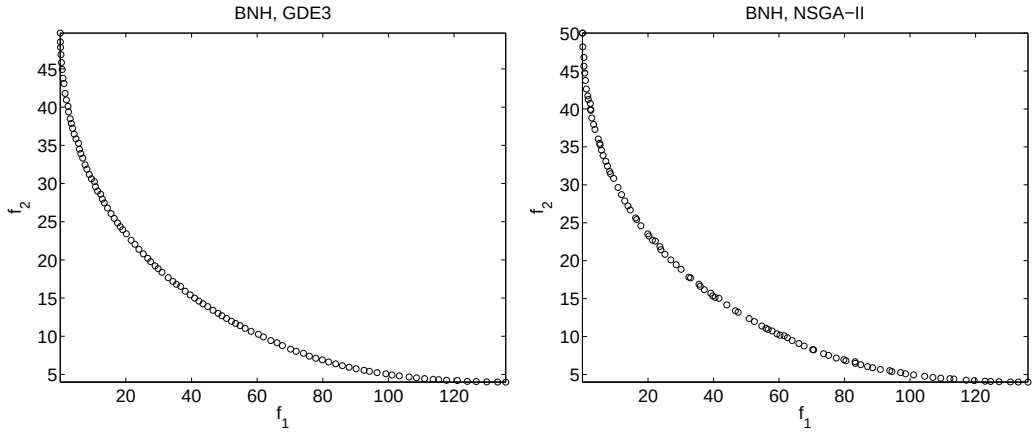


Figure 3.20: Solutions for BNH using GDE3 and NSGA-II.

The second problem, OSY, has six decision variables and constraints. Results for this problem are shown in Figure 3.21. The population size and number of generations were 200 and 250, respectively. The control parameter values for GDE3 were $CR = 0.4$ and $F = 0.3$, and for NSGA-II $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 5$, and $\eta_m = 5$.

The third problem, SRN, has two decision variables and constraints. Results for this problem are shown in Figure 3.22. Both the population size and number of generations was 100. The control parameter values for GDE3 were $CR = 0.4$ and $F = 0.3$, and for NSGA-II $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 5$, and $\eta_m = 5$.

The last problem, TNK, also has two decision variables and constraints. Results for this problem are shown in Figure 3.23. The population size and number of generations were 200 and 300, respectively. The control parameter values for GDE3 were $CR = 0.4$ and $F = 0.3$, and for NSGA-II $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 5$, and $\eta_m = 5$.

From the results, similar performance between the methods can be observed. GDE3 in general provides better diversity but NSGA-II provides a better result in the case of OSY.

The GDE versions have been successfully applied also for more challenging constrained multi-objective optimization problems such as scaling filter design [89], multi-objective scheduling for NASA's deep space network array and space science missions [72, 73], balanced surface acoustic wave and microwave filters design [56,138], Yagi-Uda antenna design [57], a software project scheduling problem [20], and a molecular sequence alignment problem [80]. The last problem is non-linear with thousands of integer decision variables. Such large problems have rarely been successfully solved with an EA. GDE3 has been developed further [19, 122] and implemented into publicly available object-oriented framework for multi-objective optimization [43]. GDE3 has also been used for comparison purposes in several studies, *e.g.*, in [38,153,154].

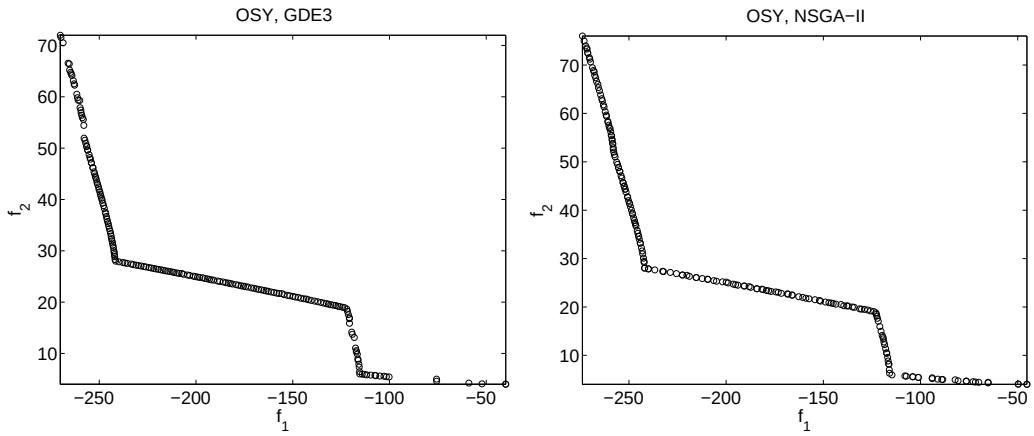


Figure 3.21: Solutions for OSY using GDE3 and NSGA-II.

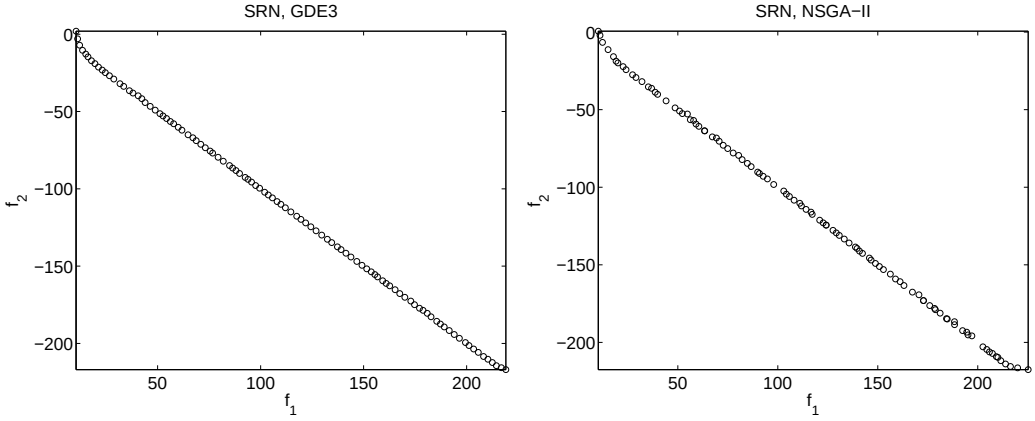


Figure 3.22: Solutions for SRN using GDE3 and NSGA-II.

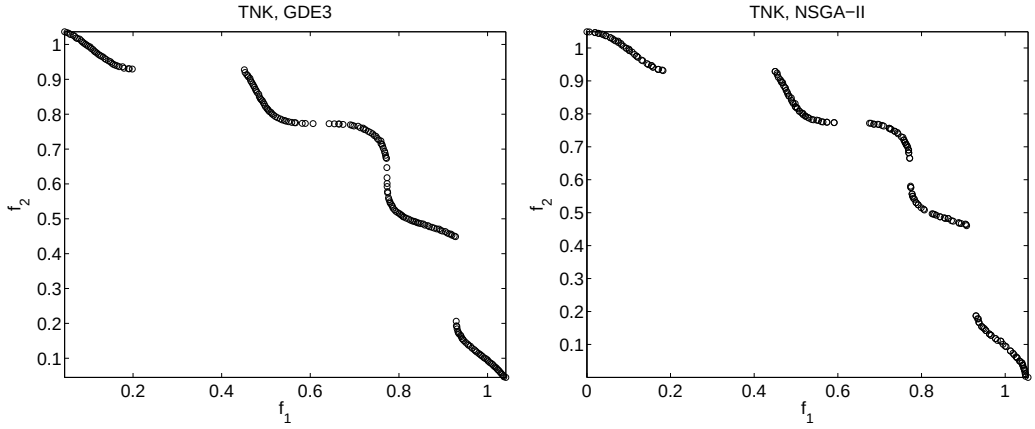


Figure 3.23: Solutions for TNK using GDE3 and NSGA-II.

Chapter IV

Conclusions and Discussion

"Essentially, all models are wrong, but some are useful."
– George E. P. Box

This chapter summarizes the main conclusions from this thesis work. Different versions of GDE and their properties are briefly reviewed. The chapter also contains discussion of the prospects and limitations of GDE, and MOEAs in general. Potential future directions are considered.

The thesis contains the work done by the author on the development of Generalized Differential Evolution (GDE) since the initial idea in [90]. GDE is a real-coded general purpose EA extended from the basic DE algorithm [134] to handle multiple objectives and constraints. Each GDE version falls back to the basic DE algorithm in the case of an unconstrained single-objective problem. GDE does not contain any extra control parameter compared to basic DE. DE was chosen as a basic search "engine" because it is an effective and widely applicable evolutionary algorithm characterized by simplicity, linear scalability, and ability to perform a rotationally invariant search [119]. The DE/rand/1/bin strategy has been used in all the GDE versions as a search method, therefore results apply mainly to this strategy. Different strategies have different search properties that would presumably affect convergence properties when used in GDE.

The first version, GDE1, was originally proposed by Lampinen [90] and then studied by the author of this thesis work. GDE1 extends DE for constrained multi-objective optimization by modifying the selection rule of basic DE. The key idea in the selection rule is that the trial vector is selected to replace the old vector in the next generation if the trial vector weakly constraint-dominates the old vector. There is neither explicit non-dominated sorting during the optimization process, nor an extra repository for non-dominated vectors, nor any specific mechanism for preserving diversity. Nevertheless, GDE1 has been observed to perform well, as can be noted from the results in *Publication I*. However, the diversity of the obtained solutions could have been better, and GDE1 has been found rather sensitive to the selection of the control parameter values (*e.g.*, the empirical results in *Publication II*).

The second version, GDE2 (*Publication III*), was proposed to improve the diversity of solutions. GDE2 makes a selection between the old and trial vector based on crowding in the objective function space when the vectors are feasible and do not dominate each

other in the objective function space. This improves the extent and distribution of an obtained set of solutions but slows down the convergence of the population because it favors isolated solutions far from the Pareto-optimal front until all the solutions have converged near the Pareto-optimal front. Thus, the additional requirement of good diversity increases the difficulty of the search and slows down the convergence. This GDE version, too, has been observed to be rather sensitive to the selection of the control parameter values (*Publication III*).

The third and latest enumerated version is GDE3 (*Publication IV* and *Publication V*). In addition to the selection operation change, a further modification to basic DE is population reduction at the end of each generation, if the size of the population has grown during the generation. For GDE3, in the case of feasible and incomparable solutions, both the old and the trial vectors are saved for the population of the next generation. At the end of each generation, the size of the population is reduced using non-dominated sorting and pruning based on crowding estimation. GDE3 provides better distribution of solutions than the earlier GDE versions and is also more robust in terms of the selection of the control parameter values, as concluded in *Publication IV* and *Publication VI*.

The diversity preservation technique of GDE3 is an improved version of the technique in NSGA-II based on the crowding distance. The technique has been noticed to provide good diversity in the case of two objectives but the diversity deteriorates with a larger number of objectives because the crowding distance metric does not estimate crowding well when the number of objectives is more than two (*Publication V*). This observation is noteworthy because NSGA-II is the most used MOEA, several multi-objective DE variants apply crowding distance, and the crowding distance metric has subsequently been used in several studies with more than two objectives.

In the light of the defect in the crowding distance metric, GDE3 has been further developed with the diversity preservation technique designed for many-objective problems (*Publication VII*). This technique provides good diversity also in cases with more than two objectives. Furthermore, it is fast, especially when solving problems having only a few objectives (*cf.* Figure 3.12 and results in *Publication VII*). The time needed by the pruning technique increases when the number of objectives and number of non-dominated solutions to be pruned increases, but it is substantially less compared to similar approaches in other MOEAs [70].

The influence of the control parameters has been studied and discussed in *Publication II* and *Publication VI* with respect to GDE1 and GDE3. Multi-objective optimization is fundamentally different from single-objective optimization since the population is not expected to converge to a single point. The study finds that GDE3 is more robust with respect to control parameter values and provides a better diversity than GDE1.

The non-linear relationship between CR and F was observed following the theory of basic single-objective DE [148] concerning the relationship between the control parameters and the development of the population standard deviation. Based on this observation, it is advisable to select values for CR and F satisfying the condition $1.0 < c < 1.5+$, where c denotes the change of the population standard deviation between successive generations due to the variation (crossover and mutation) operations. When $c < 1$, the standard deviation decreases, and when $c > 1$, the standard deviation increases. If the difficulty of the objectives differ, it is better to use a small CR value to prevent convergence of one

objective before the other of others.

The various GDE versions have been used in a number of problems having different numbers of objectives and constraints [54, 56, 57, 72, 80, 82, 89, 138]. The experimental results in Chapter 3 demonstrate how the performance of the method has improved throughout the development steps. Results in Table 2 of *Publication IV* and in [83, 91] demonstrate how the constraint handling technique used in GDE reduces the number of function evaluations needed. GDE3 with the improved diversity preservation has been able to solve successfully (*i.e.*, find a good set of non-dominated solutions) some difficult problems involving up to five objectives [84] and performs well compared to several other MOEAs [136].

The main contribution of this thesis work is the study of GDE1 and the development of the later GDE versions. This includes programming all the versions and performing the tests. The thesis work has contributed new information about:

- Diversity preservation in multi-objective optimization.
- The influence of control parameters in a multi-objective DE.
- Usability of DE for constrained multi-objective optimization.

GDE itself is an *a posteriori* method, *i.e.*, a set of non-dominated points is first generated and after that the decision-maker picks a suitable compromise solution. GDE can be converted to *a priori* and interactive optimization without great difficulty. In *a priori* optimization the original problem is converted to a single-objective form that can be solved with GDE since it then operates exactly as the original DE algorithm. In the interactive form, the search process could be interrupted occasionally (*e.g.*, after a selected number of generations or based on the development of the solutions) for the decision-maker to conduct refinement of objectives and constraints. The search could then be continued with the old and/or reinitialized population and the refined problem definition. Refinement of the objectives and constraints does not need to mean merely adding or removing them; refinement could be extended to modification of the objective and constraint functions. It might be necessary to update the objective and constraint functions since they are often models of reality and therefore contain inaccuracy. This inaccuracy should be kept in mind during numerical optimization because the global optimum for a model might be different than for the actual optimization problem. Therefore, searching for a better solution for a model is not worthwhile beyond a certain precision.

Currently, GDE is a potential general purpose optimizer for global optimization with constraints and objectives. The different GDE versions have their own target domains. For example, GDE1 is suitable for single-objective optimization and if diversity of solutions is not of primary importance. The original version of GDE3 is suitable when the number of objectives is two and GDE3 with the diversity preservation technique proposed in *Publication VII* is suitable with a higher number of objectives. GDE2 can be useful if the nature of possible discontinuities in the Pareto-optimal front needs to be investigated. For example, ZDT3 has disconnected Pareto-optimal fronts but based on a set of non-dominated points it is not certain if the actual front has discontinuity or some parts are just not covered. Observation of the whole final set of solutions (not just

non-dominated solutions) indicates that the Pareto-optimal front is truly disconnected (*cf.* ZDT3 in Figure 3.4).

The convergence property of GDE has been briefly discussed at the beginning of Chapter 3. As was stated, GDE with the DE/rand/1/bin strategy cannot be proven theoretically to always converge. However, if the DE strategy is modified slightly, as discussed in Section 2.3.1, GDE1 can be proven to converge since both necessary conditions mentioned in Section 2.2.5 are then fulfilled. This does not hold with the other GDE versions since diversity maintenance is a part of selection and deterioration of solutions is possible, as noticed in [98]. If the selection is changed to consider only Pareto-dominance (and not diversity) after some number of generations (*e.g.*, when a certain number of non-dominated solutions has been obtained), also the later GDE versions with the modified DE strategy would then converge in theory.

Some limitations in GDE exist. When the number of objectives increases, the speed of the search will slow down. The reason for this phenomenon is that GDE (as many other MOEAs) applies the Pareto-dominance relation in its selection. When the number of objectives increases, the proportion of non-dominated members in the population will also increase rapidly and the selection based on Pareto-dominance is not able to sort the members. This has been illustrated in [33, p. 417] and [85].

In the case of diversity preservation, diversity will become the ruling selection criterion between solutions when the number of objectives increases, and the emphasis on extreme solutions may even advance the search away from the Pareto-optimal front, as noted in [85,116]. Thus, diversity preservation will slow down convergence further. It is important to note that the cause of these difficulties lies in the definition of Pareto-optimality. A new term (*e.g.*, hyper-, super-, or ultra-objective) would be needed for cases when the number of objectives is so large that selection based on Pareto-dominance is not able to sort the members of a population.

A large number of objectives also causes other problems, *i.e.*, visualization of the obtained optimization result becomes more difficult when the number of objectives is more than three. With the increase in the number of objectives, the number of points to be used should increase exponentially in order to cover the Pareto-optimal front with the same density. This demand will increase the computational expense, and it is also questionable, whether the decision-maker will be able to cope with the huge number of alternative solutions generated. These issues of visualization, choice, and computational expense are probably the reasons why most MOEA research has focused on problems with only two or three objectives [25, p. 305].

Current research is attempting to solve the difficulties arising with a large number of objectives (*e.g.*, [85]). However, it appears that a large number of objectives causes so many difficulties that it might be better to use *a posteriori* MOEAs only for optimization problems with a couple of objectives. If a problem contains too large a number of objectives to be solved using *a posteriori* MOEA, then some *a priori* technique (*cf.* Section 2.2.4) could be used instead. It might also be possible to decrease the number of objectives (*e.g.*, using techniques in [14,37]) and solve the resulting problem using *a posteriori* MOEA.

Finally, it can be concluded that GDE3 with the diversity preservation technique for many-objective problems is a good choice for global optimization with different types

of decision variables, constraints, and a few (*i.e.*, 1–5) objectives. This version has performed very well when compared with other modern MOEAs [136].

Since the results with the GDE versions are mainly based on a limited number of experiments, the full applicability range in terms of the number of constraints and objectives is not precisely known and therefore exact limits cannot be given.

Future Directions

Many real world problems have computationally expensive objective and constraint functions. These have been problematic for EAs since EAs generally require a large number of function evaluations. One resolution might be parallelization of the algorithm. GDE, like EAs in general, can be easily parallelized [119, pp. 267–276, 401–404]. Another approach for computationally expensive functions is to use approximations of functions, meta-models [71], during most of the search and evaluate the actual functions only when really required. These modifications to GDE are probably useful when GDE is applied to practical problems.

Further investigation of the effect of control parameters is necessary in order to increase the usability of GDE. The effect of control parameters on the compromise between speed and robustness would need further research. There already exists some research on an automatic control parameter adaptation for DE, *e.g.*, in [1,64,149]. This is an interesting and important research topic. Automatic control parameter adaption is needed in order to increase the usability of DE and GDE.

The NFL theorem has been confirmed to hold also in multi-objective optimization [29]. However, it might be a case that practical problems form a subset of all the possible problems. Then, it would be possible to study the properties of this subset and find an optimization algorithm which is best suited for this subset.

As noted in *Publication VII* and [79], the diversity preservation technique presented for many-objective problems could be improved further. This issue as well as the concept of optimal diversity form potential subjects for future study.

CHAPTER V

Errata and Additional Information to Publications

"*Errare humanum est perseverare diabolicum.*"
– Lucius A. Seneca (Seneca the Younger)

Publication I

In Section 2 (p. 3), the reference after the term "Pareto-optimization" should be [77].

In Section 3 (pp. 4–5), the text should read "decision variable vector" instead of "objective vector".

The normalized version of maximal spread has been utilized and symbol $\bar{D}$ should be used instead of D.

The reference 13 should be in the form: "Kenneth V. Price, *New Ideas in Optimization*, chapter An Introduction to Differential Evolution, pp. 79–108, McGraw-Hill, London, 1999."

In the results, the non-dominated solutions are presented for all the methods (not just GDE).

Publication II

In Figures 1–10, the label of the vertical axis should contain *GD* instead of *CD* for generational distance.

Symbol c in Section 4 denotes the change of the population "standard deviation", not "variance".

The number of generations were kept fixed for individual test problems, *i.e.*, not the same value was used for all the problems but suitable values were experimentally found for each test problem.

Publication III

In Section 2 (p. 753), the reference after the term "Pareto-optimization" should be [77].

The normalized version of maximal spread has been utilized and symbol $\bar{D}$ should be used instead of D.

The results of SPEA were taken from [160]. The results of NSGA-II were generated using a program code available in [76]. The control parameter values of NSGA-II for the ZDT problems were obtained with the program code.

Publication IV

At the end of Section 2 (p. 444), in the second condition of constraint-domination, the text should read "constraint function violation space" instead of "constraint function space".

The population reduction term in Equation (3) (p. 445) should read as in Chapter 3, Figure 3.7:

$$
\left\{
\begin{array}{l}
\text{While } n > 0 \\
\qquad \text{Select } \vec{x} \in \mathcal{P} = \{\vec{x}_{1,G+1}, \vec{x}_{2,G+1}, \ldots, \vec{x}_{NP+n,G+1}\} : \\
\qquad\qquad \left\{
\begin{array}{l}
\vec{x} \text{ belongs to the last non-dominated set of } \mathcal{P} \\
\wedge \\
\vec{x} \text{ is the most crowded in the last non-dominated set}
\end{array}
\right. \\
\qquad \text{Remove } \vec{x} \text{ from } \mathcal{P} \\
\qquad n = n - 1
\end{array}
\right.
$$

The term "constrained-non-domination" in the first footnote means a situation in which two solutions do not dominate each other in the space of the constraint violations.

Omni-Optimizer [41] was selected for comparison in the case of single-objective optimization since Omni-Optimizer is a multipurpose optimizer in the same way as GDE3. Omni-Optimizer was novel at the time, and the reported results were promising.

Publication V

A heap is a tree structure in which the root node has a smaller value than the nodes of the subtrees. Creation of the heap as well as insertion and removal operations of nodes can be done with a relatively low computational cost [27, pp. 140–152].

At the end of Section V, instead of "random values for *CD*s were used instead of calculated values" it would have been more accurate to write "random values were used instead of calculated *CD* values". Thus, the method was otherwise the same but randomly generated values were used instead of calculated *CD* values to test if the pruning result using *CD* values differs from a random pruning.

Publication VI

Symbol c denotes the change of the population "standard deviation", not "variance".

After the sentence "In practice it has been observed that $c < 1.5$ is suitable upper limitation for most of the cases" should be reference [94] instead of [119].

The large average *GD* values for the SCH1 test problem in Table 2 can be explained by looking at the middle of the sub-figure of Figure 1 in *Publication II* and *Publication VI*. Generational distance values are large when F is close to 0 or greater than 1 and therefore also the average values over the control parameter value combinations are large.

Publication VII

Equation (1) provides "a lower bound" for the Euclidean distance between two vectors, not "an upper bound" as written after the equation.

At the end of Section 2 (p. 557), the complexity analysis is erroneous: the time complexity $O(MN \log N)$ for finding all M nearest neighbors holds only with a fixed dimensional space. The actual time complexity class of the nearest neighbor method in *Publication VII* cannot be given since it is problem-dependent. It is, however, between $O(MN \log N)$ and $O(MN^2)$. Therefore, the complexity class estimate at the end of page 557 might not hold, although empirical evidence would seem to support it.

The normalized version of maximal spread has been utilized and symbol $\bar{D}$ should be used instead of D.

A term "nadir" should be removed from Section 3 (p. 558).

At the end of Section 3 (p. 561), there should be a term "log-linear" instead of "logarithmic".

For any remaining errors, *cf.* the quotation at the beginning of this chapter.

[1] Abbass, H. A. The self-adaptive Pareto Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI, USA, May 2002), X. Yao, Ed., IEEE Service Center, Piscataway, NJ, USA, pp. 831–836.

[2] Abbass, H. A., and Sarker, R. The Pareto Differential Evolution algorithm. *International Journal on Artificial Intelligence Tools 11*, 4 (2002), 531–552.

[3] Abbass, H. A., Sarker, R., and Newton, C. PDE: a Pareto-frontier Differential Evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)* (Seoul, South Korea, 2001), IEEE Service Center, Piscataway, NJ, USA, pp. 971–978.

[4] Ali, M., Siarry, P., and Pant, M. An efficient Differential Evolution based algorithm for solving multi-objective optimization problems. *European Journal of Operational Research 217*, 2 (2012), 404–416.

[5] Babu, B. V., and Jehan, M. M. L. Differential Evolution for multi-objective optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)* (Canberra, Australia, Dec 2003), R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, Eds., IEEE Press, Piscataway, NJ, USA, pp. 2696–2703.

[6] Bäck, T. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, NY, USA, New York, 1996.

[7] Baek, S., Jeon, B., and Sung, K.-M. A fast encoding algorithm for vector quantization. *IEEE Signal Processing Letters 4*, 12 (Dec 1997), 325–327.

[8] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

[9] Becerra, R. L., and Coello Coello, C. A. Solving hard multiobjective optimization problems using ϵ-constraint with cultured Differential Evolution. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)* (Reykjavik, Iceland, Sept 2006), T. P. Runarsson, H.-G. Beyer,

E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 543–552.

[10] Beldianu, S. F. A fast nearest neighbor search algorithm for image vector quantization. In *Proceedings of the International Symposium on Signals, Circuits & Systems (ISSCS 2005)* (Iasi, Romania, July 2005), IEEE, pp. 485–488.

[11] Bergey, P. K. An agent enhanced intelligent spreadsheet solver for multi-criteria decision making. In *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 1999)* (Milwaukee, Wisconsin, USA, Aug 1999), W. D. Haseman and D. L. Nazareth, Eds., Association for Information Systems, Atlanta, GA, USA, pp. 966–968.

[12] Beyer, H.-G. *The Theory of Evolution Strategies*. Springer-Verlag, Berlin, Heidelberg, Germany, 2001.

[13] Bradstreet, L., While, L., and Barone, L. Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (Singapore, Sept 2007), D. Srinivasan and L. Wang, Eds., IEEE, pp. 3203–3210.

[14] Brockhoff, D., and Zitzler, E. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)* (Reykjavik, Iceland, Sept 2006), T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 533–542.

[15] Cai, Z., Gong, W., and Huang, Y. A novel Differential Evolution algorithm based on ϵ-domination and orthogonal design method for multiobjective optimization. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)* (Matsushima, Japan, March 2007), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 286–301.

[16] Chang, C. S., and Xu, D. Y. Differential Evolution based tuning of fuzzy automatic train operation for mass rapid transit system. *IEE Proceedings on Electric Power Applications 147*, 3 (May 2000), 206–212.

[17] Chang, C. S., Xu, D. Y., and Quek, H. B. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. *IEE Proceedings on Electric Power Applications 146*, 5 (Sept 1999), 577–583.

[18] Chang, T.-T., and Chang, H.-C. Application of Differential Evolution to passive shunt harmonic filter planning. In *8th International Conference on Harmonics and Quality of Power* (Athens, Greece, Oct 1998), IEEE Press, pp. 149–153.

[19] Chen, W., Shi, Y.-j., and Teng, H.-f. A Generalized Differential Evolution combined with EDA for multi-objective optimization problems. In *Proceedings of*

the 4th international conference on Intelligent Computing (ICIC 2008): Advanced Intelligent Computing Theories and Applications - with Aspects of Artificial Intelligence (Shanghai, China, September 2008), D.-S. Huang, D. C. Wunsch II, D. S. Levine, and K.-H. Jo, Eds., vol. 5226 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 140–147.

[20] Chicano, F., Luna, F., Nebro, A. J., and Alba, E. Using multi-objective metaheuristics to solve the software project scheduling problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO 2011)* (Dublin, Ireland, July 2011), N. Krasnogor and P. L. Lanzi, Eds., ACM, pp. 1915–1922.

[21] Clerc, M. *Particle Swarm Optimization*. ISTE, 2006.

[22] Coello Coello, C. A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering 191*, 11–12 (2002), 1245–1287.

[23] Coello Coello, C. A. EMOO web page, 2012. [online] http://www.lania.mx/~ccoello/EMOO/, 15.4.2012.

[24] Coello Coello, C. A., and Cruz Cortés, N. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines 6*, 2 (2005), 163–190. Inverted generational distance is a proposal of an anonymous reviewer.

[25] Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. Springer Science+Business Media, 2007.

[26] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York, United States of America, 2002.

[27] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms*. Prentice-Hall, 1990.

[28] Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (San Francisco, CA, USA, July 2001), L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., Morgan Kaufmann, pp. 283–290.

[29] Corne, D. W., and Knowles, J. D. No Free Lunch and free leftovers theorems for multiobjective optimisation problems. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)* (Faro, Portugal, April 2003), C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds., vol. 2632 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 327–341.

[30] Corne, D. W., Knowles, J. D., and Oates, M. J. The Pareto envelope-based selection algorithm for multiobjective optimization. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)* (Paris, France, Sept 2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 839–848.

[31] Das, S., and Suganthan, P. N. Differential Evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation 15*, 1 (February 2011), 4–31.

[32] De Jong, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Dissertation Abstracts International 36(10), 5140B (University Microfilms No. 76-9381), 1975.

[33] Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, England, 2001.

[34] Deb, K., Agarwal, S., Pratab, A., and Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI (PPSN VI)* (Paris, France, 2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 849–858.

[35] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (April 2002), 182–197.

[36] Deb, K., Pratap, A., and Moitra, S. Mechanical component design for multiple objectives using elitist non-dominated sorting GA. In *Proceedings of the Parallel Problem Solving from Nature VI (PPSN VI)* (Paris, France, 2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 859–868.

[37] Deb, K., and Saxena, D. K. Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)* (Vancouver, BC, Canada, July 2006), G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., IEEE Press, pp. 3353–3360.

[38] Deb, K., Sinha, A., and Kukkonen, S. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (Seattle, WA, USA, July 2006), M. Keijzer et al., Eds., ACM Press, pp. 1141–1148.

[39] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI, USA, May 2002), X. Yao, Ed., IEEE Service Center, Piscataway, NJ, USA, pp. 825–830.

[40] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, A. Abraham, L. Jain, and R. Goldberg, Eds. Springer-Verlag, London, 2005, pp. 105–145.

[41] Deb, K., and Tiwari, S. Omni-optimizer: A procedure for single and multi-objective optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)* (Guanajuato, Mexico, March 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 47–61.

[42] Dorigo, M., and Stützle, T. *Ant Colony Optimization*. The MIT Press, 2004.

[43] Durillo, J. J., and Nebro, A. J. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software 42*, 10 (2011), 760–771.

[44] Eiben, A. E., and Smith, J. E. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2003.

[45] Fleicher, M. The measure of Pareto optima: Applications to multi-objective metaheuristics. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)* (Faro, Portugal, April 2003), C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds., vol. 2632 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 519–533.

[46] Fonseca, C. M. *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. PhD thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, United Kingdom, 1995.

[47] Fonseca, C. M., and Fleming, P. J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (Urbana-Champaign, IL, USA, July 1993), S. Forrest, Ed., Morgan Kaufmann Publishers Inc., pp. 416–423.

[48] Fonseca, C. M., and Fleming, P. J. On the performance assessment and comparison of stochastic multiobjective optimizers. In *Proceedings of the Parallel Problem Solving from Nature IV (PPSN IV)* (Berlin, Germany, Sept 1996), H.-M. Voigt, W. Ebeling, I. Rechenberger, and H.-P. Schwefel, Eds., vol. 1141 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 584–593.

[49] Fonseca, C. M., and Fleming, P. J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms–part I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans 28*, 1 (Jan 1998), 26–37.

[50] Fonseca, C. M., Grunert da Fonseca, V., and Paquete, L. Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)* (Guanajuato, Mexico, March 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 250–264.

[51] Fonseca, C. M., Paquete, L., and López-Ibáñez, M. An improved dimension-sweep algorithm for the hypervolume indicator. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)* (Vancouver, BC, Canada, July 2006), G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., IEEE Press, pp. 3973–3979.

[52] Fränti, P., Virmajoki, O., and Hautamäki, V. Fast agglomerative clustering using k-nearest neighbor graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (Nov 2006), 1875–1881.

[53] Giel, O., and Lehre, P. K. On the effect of populations in evolutionary multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (Seattle, WA, USA, July 2006), M. Keijzer et al., Eds., ACM Press, pp. 651–658.

[54] Giuliano, M. E., and Johnston, M. D. Multi-objective evolutionary algorithms for scheduling the James Webb space telescope. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)* (Sydney, Australia, September 2008), J. Rintanen, B. Nebel, J. C. Beck, and E. Hansen, Eds., AAAI Press, pp. 107–115.

[55] Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[56] Goudos, S. K., and Sahalos, J. N. Pareto optimal microwave filter design using multiobjective Differential Evolution. *IEEE Transactions on Antennas and Propagation 58*, 1 (2010), 132–144.

[57] Goudos, S. K., Siakavara, K., Vafiadis, E., and Sahalos, J. N. Pareto optimal Yagi-Uda antenna design using multi-objective Differential Evolution. *Progress In Electromagnetics Research 10* (2010), 231–251.

[58] Guan, L., and Kamel, M. Equal-average hyperplane partitioning method for vector quantization of image data. *Pattern Recognition Letters 13*, 10 (Oct 1992), 693–699.

[59] Hansen, N., and Ostermeier, A. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC 1996)* (Nayoya, Japan, May 1996), T. Fukuda and T. Furuhashi, Eds., IEEE Press, pp. 312–317.

[60] Hernández-Diaz, A. G., Santana-Quintero, L. V., Coello Coello, C., Caballero, R., and Molina, J. A new proposal for multi-objective optimization using Differential Evolution and rough sets theory. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (Seattle, WA, USA, July 2006), M. Keijzer et al., Eds., ACM Press, pp. 675–682.

[61] Holland, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[62] Horn, J., Nafpliotis, N., and Goldberg, D. E. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation* (Orlando, FL, USA, June 1994), Z. Michalewicz, H. Kitano, D. Schaffer, H.-P. Schwefel, and D. Fogel, Eds., IEEE Press, pp. 82–87.

[63] Huang, V. L., Qin, A. K., Deb, K., Zitzler, E., Suganthan, P. N., Liang, J. J., Preuss, M., and Huband, S. Problem definitions for performance assessment on multi-objective optimization algorithms. Technical report, School of EEE, Nanyang Technological University, Singapore, 639798, January 2007. [online] http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2007/CEC-07-TR-13-Feb.pdf 15.4.2012.

[64] Huang, V. L., Qin, A. K., and Suganthan, P. N. Self-adaptive Differential Evolution algorithm for constrained real-parameter optimization. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)* (Vancouver, BC, Canada, July 2006), G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., IEEE Press, pp. 324–331.

[65] Huband, S., Barone, L., While, L., and Hingston, P. A scalable multi-objective test problem toolkit. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)* (Guanajuato, Mexico, March 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 280–295.

[66] Huband, S., Barone, L., While, L., and Hingston, P. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation 10*, 5 (2006), 477–506.

[67] Hughes, E. J. Evolutionary multi-objective optimization: Many once or one many? In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)* (Edinburgh, Scotland, Sept 2005), B. McKay et al., Eds., IEEE Press, pp. 222–227.

[68] Iorio, A., and Li, X. Solving rotated multi-objective optimization problems using Differential Evolution. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004)* (Cairns, Australia, Dec 2004), G. I. Webb and X. Yu, Eds., vol. 3339 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 861–872.

[69] Iorio, A. W., and Li, X. Incorporating directional information within a Differential Evolution for multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (Seattle, WA, USA, July 2006), M. Keijzer et al., Eds., ACM Press, pp. 691–697.

[70] Jensen, M. T. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation 7*, 5 (Oct 2003), 503–515.

[71] Jin, Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing 9*, 1 (Jan 2005), 3–12.

[72] Johnston, M. D. Multi-objective scheduling for NASA's future deep space network array. In *Proceedings of the 5th International Workshop on Planning and Scheduling for Space (IWPSS 2006)* (Baltimore, MD, USA, Sept 2006), Space Telescope Science Institute (STScI), pp. 27–35.

[73] Johnston, M. D., and Giuliano, M. E. Multi-objective scheduling for space science missions. *Journal of Advanced Computational Intelligence and Intelligent Informatics 15*, 8 (2011), 1140–1148.

[74] Justesen, P. D., and Ursem, R. K. Multiobjective distinct candidates optimization (MODCO): A cluster-forming Differential Evolution algorithm. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)* (Nantes, France, April 2009), M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds., vol. 5467 of *Lecture Notes in Computer Science (LNCS)*, pp. 525–539.

[75] Kannan, B. K., and Kramer, S. N. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design 116*, 2 (1994), 405–411.

[76] Kanpur Genetic Algorithms Laboratory. Software developed at KanGAL, 2007. [online] http://www.iitk.ac.in/kangal/codes.shtml, 14.11.2007.

[77] Knowles, J. D. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Reading, UK, January 2002.

[78] Knowles, J. D., and Corne, D. W. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation 8*, 2 (April 2000), 149–172.

[79] Kukkonen, S., and Deb, K. An empirical scalability study of a non-dominated solutions pruning algorithm. In *Late Breaking Papers Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)* (Matsushima, Japan, March 2007), pp. 69–74.

[80] Kukkonen, S., Jangam, S. R., and Chakraborti, N. Solving the molecular sequence alignment problem with Generalized Differential Evolution 3 (GDE3). In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007)* (Honolulu, HI, USA, April 2007), P. Bonissone, C. Coello Coello, and Y. Jin, Eds., IEEE, pp. 302–309.

[81] Kukkonen, S., and Lampinen, J. A Differential Evolution algorithm for constrained multi-objective optimization: Initial assessment. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)* (Innsbruck, Austria, Feb 2004), M. H. Hamza, Ed., ACTA Press, pp. 96–102.

[82] Kukkonen, S., and Lampinen, J. Mechanical component design for multiple objectives using Generalized Differential Evolution. In *Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM 2004)* (Bristol, United Kingdom, April 2004), I. C. Parmee, Ed., Springer, pp. 261–272.

[83] Kukkonen, S., and Lampinen, J. Constrained real-parameter optimization with Generalized Differential Evolution. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)* (Vancouver, BC, Canada, July 2006), G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., IEEE Press, pp. 911–918.

[84] Kukkonen, S., and Lampinen, J. Performance assessment of Generalized Differential Evolution 3 (GDE3) with a given set of problems. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (Singapore, Sept 2007), D. Srinivasan and L. Wang, Eds., IEEE, pp. 3593–3600.

[85] Kukkonen, S., and Lampinen, J. Ranking-dominance and many-objective optimization. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (Singapore, Sept 2007), D. Srinivasan and L. Wang, Eds., IEEE, pp. 3983–3990.

[86] Kukkonen, S., and Lampinen, J. Generalized Differential Evolution for constrained multi-objective optimization. In *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, L. T. Bui and S. Alam, Eds. Information Science Reference, 2008, pp. 43–75.

[87] Kukkonen, S., and Lampinen, J. Generalized Differential Evolution for general non-linear optimization. In *Proceedings in Computational Statistics (COMPSTAT 2008)* (Porto, Portugal, August 2008), P. Brito, Ed., Physica-Verlag Heidelberg Germany, pp. 459–471.

[88] Kukkonen, S., and Lampinen, J. Performance assessment of Generalized Differential Evolution 3 with a given set of constrained multi-objective test problems. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009)* (Trondheim, Norway, May 2009), P. Haddow et al., Eds., pp. 1943–1950.

[89] Kukkonen, S., Sampo, J., and Lampinen, J. Applying Generalized Differential Evolution for scaling filter design. In *Proceedings of Mendel 2004, 10th International Conference on Soft Computing* (Brno, Czech Republic, June 2004), R. Matoušek and P. Ošmera, Eds., Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, pp. 28–33.

[90] Lampinen, J. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001. [online] http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 15.4.2012.

[91] Lampinen, J. Multi-constrained nonlinear optimization by the Differential Evolution algorithm. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001. [online] http://www.it.lut.fi/kurssit/03-04/010778000/DECONSTR.PDF, 15.4.2012.

[92] Lampinen, J. A constraint handling approach for the Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI, USA, May 2002), X. Yao, Ed., IEEE Service Center, Piscataway, NJ, USA, pp. 1468–1473.

[93] Lampinen, J. Multi-constrained nonlinear optimization by the Differential Evolution algorithm. In *Soft Computing and Industry: Recent Advances*, R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, Eds. Springer-Verlag, 2002, pp. 305–318.

[94] Lampinen, J. Personal communication, 2005.

[95] Lampinen, J., and Storn, R. Differential Evolution. In *New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing*, G. C. Onwubolu and B. V. Babu, Eds., vol. 141. Springer-Verlag, 2004, pp. 123–166.

[96] Lampinen, J., and Zelinka, I. Mechanical engineering design optimization by Differential Evolution. In *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw-Hill, London, 1999, pp. 128–146.

[97] Lampinen, J., and Zelinka, I. On stagnation of the Differential Evolution algorithm. In *Proceedings of Mendel 2000, 6th International Conference on Soft Computing* (Brno, Czech Republic, June 2000), Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, pp. 76–83.

[98] Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation 10*, 3 (Fall 2002), 263–282.

[99] Li, H., and Zhang, Q. A multiobjective Differential Evolution based on decomposition for multiobjective optimization with variable linkages. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)* (Reykjavik, Iceland, Sept 2006), T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 583–592.

[100] Lin, Y.-C., Hwang, K.-S., and Wang, F.-S. Hybrid Differential Evolution with multiplier updating method for nonlinear constrained optimization problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI, USA, May 2002), X. Yao, Ed., IEEE Service Center, Piscataway, NJ, USA, pp. 872–877.

[101] Liu, B., Fernandez, F., Zhang, Q., Pak, M., Sipahi, S., and Gielen, G. An enhanced MOEA/D-DE and its application to multiobjective analog cell sizing. In *Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010)* (Barcelona, Spain, July 2010), G. Fogel et al., Eds., IEEE Press, pp. 1–7.

[102] Madavan, N. K. Multiobjective optimization using a Pareto Differential Evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI, USA, May 2002), X. Yao, Ed., IEEE Service Center, Piscataway, NJ, USA, pp. 1145–1150.

[103] Mezura-Montes, E., and Coello Coello, C. A. Constrained optimization via multiobjective evolutionary algorithms. In *Multiobjective Problems Solving from Nature: From Concepts to Applications*, J. Knowles, D. Corne, and K. Deb, Eds. Springer-Verlag, Natural Computing Series, 2008, pp. 53–76.

[104] Mezura-Montes, E., Coello Coello, C. A., and Tun-Morales, E. I. Simple feasibility rules and Differential Evolution for constrained optimization. In *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI 2004)* (Mexico City, Mexico, April 2004), R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and J. H. S. Azuela, Eds., vol. 2972 of *Lecture Notes in Computer Science*, Springer, pp. 707–716.

[105] Mezura-Montes, E., Miranda-Varela, M. E., and del Carmen Gómez-Ramón, R. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences 180*, 22 (2010), 4223–4262.

[106] Mezura-Montes, E., Reyes-Sierra, M., and Coello Coello, C. A. Multi-objective optimization using Differential Evolution: A survey of the state-of-the-art. In *Advances in Differential Evolution*, U. K. Chakraborty, Ed., vol. 143. Springer-Verlag, Studies in Computational Intelligence Series, 2008, pp. 173–196.

[107] Mezura-Montes, E., Velázquez-Reyes, J., and Coello Coello, C. A. A comparative study of Differential Evolution variants for global optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (Seattle, WA, USA, July 2006), M. Keijzer et al., Eds., ACM Press, pp. 485–492.

[108] Miettinen, K. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1998.

[109] Miettinen, K., and Mäkelä, M. M. Interactive multiobjective optimization system WWW-NIMBUS on the internet. *Computers & Operations Research 27*, 7–8 (2000), 709–723.

[110] Miettinen, K., and Mäkelä, M. M. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research 170*, 3 (2006), 909–922.

[111] Miettinen, K., and Mäkelä, M. M. WWW-NIMBUS, 2007. [online] http://nimbus.mit.jyu.fi/, 14.11.2007.

[112] Miettinen, K., Mäkelä, M. M., and Toivanen, J. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization 27*, 4 (2003), 427–446.

[113] Montaño, A. A., Coello Coello, C. A., and Mezura-Montes, E. MODE-LD+SS: A novel Differential Evolution algorithm incorporating local dominance and scalar selection mechanisms for multi-objective optimization. In *Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010)* (Barcelona, Spain, July 2010), G. Fogel et al., Eds., IEEE Press, pp. 1–8.

[114] Pan, J.-S., Lu, Z.-M., and Sun, S.-H. An efficient encoding algorithm for vector quantization based on subvector technique. *IEEE Transactions on Image Processing 12*, 3 (March 2003), 265–270.

[115] Parsopoulos, K. E., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., and Vrahatis, M. N. Vector evaluated Differential Evolution for multiobjective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)* (Portland, Oregon, USA, June 2004), G. W. Greenwood, Ed., IEEE, pp. 204–211.

[116] Praditwong, K., and Yao, X. How well do multi-objective evolutionary algorithms scale to large problems. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (Singapore, Sept 2007), D. Srinivasan and L. Wang, Eds., IEEE, pp. 3959–3966.

[117] Price, K., and Storn, R. Minimizing the real functions of the ICEC'96 contest by Differential Evolution. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC 1996)* (Nagoya, Japan, May 1996), T. Fukuda and T. Furuhashi, Eds., IEEE Press, pp. 842–844.

[118] Price, K. V. An introduction to Differential Evolution. In *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw-Hill, London, 1999, pp. 79–108.

[119] Price, K. V., Storn, R. M., and Lampinen, J. A. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, 2005.

[120] Pryke, A., Mostaghim, S., and Nasemi, A. Heatmap visualization of population based multi objective algorithms. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)* (Matsushima, Japan, March 2007), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 361–375.

[121] Ra, S.-W., and Kim, J.-K. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Transactions on Circuits and Systems-II 40*, 9 (Sept 1993), 576–579.

[122] Ramesh, S., Kannan, S., and Baskar, S. An improved generalized differential evolution algorithm for multi-objective reactive power dispatch. *Engineering Optimization 44*, 4 (2012), 391–405.

[123] Rao, S. S. *Engineering Optimization: Theory and Practice*, 3rd ed. John Wiley & Sons, Inc., 1996.

[124] Robič, T., and Filipič, B. DEMO: Differential Evolution for multiobjective optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)* (Guanajuato, Mexico, March 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 520–533.

[125] Rönkkönen, J., Kukkonen, S., and Lampinen, J. A comparison of Differential Evolution and Generalized Generation Gap model. *Journal of Advanced Computational Intelligence and Intelligent Informatics 9*, 5 (September 2005), 549–555.

[126] Rönkkönen, J., Kukkonen, S., and Price, K. V. Real-parameter optimization with Differential Evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)* (Edinburgh, Scotland, Sept 2005), B. McKay et al., Eds., IEEE Press, pp. 506–513.

[127] Rudolph, G. Evolutionary search under partially ordered fitness sets. In *In Proceedings of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ISI 2001)* (American University in Dubai, March 2001), ICSC Academic Press, pp. 818–822.

[128] Rudolph, G., and Agapie, A. Convergence properties of some multi-objective evolutionary algorithms. In *Proceedings of the Congress on Evolutionary Computation (CEC 2000)* (La Jolla, CA, USA, July 2000), IEEE Press, pp. 1010–1016.

[129] Salomon, R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark function. a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems 39*, 3 (1996), 263–278.

[130] Santana-Quintero, L. V., and Coello Coello, C. A. An algorithm based on Differential Evolution for multi-objective optimization. *International Journal of Computational Intelligence Research 1*, 2 (2005), 151–169.

[131] Schaffer, J. D. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, USA, 1984.

[132] Srinivas, N., and Deb, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation 2*, 3 (1994), 221–248.

[133] Storn, R. System design by constraint adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation 3*, 1 (April 1999), 22–34.

[134] Storn, R., and Price, K. Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI, University of California, Berkeley, March 1995. [online] www.icsi.berkeley.edu/ftp/global/pub/techreports/1995/tr-95-012.pdf, 15.4.2012.

[135] Storn, R., and Price, K. Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (Dec 1997), 341–359.

[136] Suganthan, P. N. Performance assessment on multi-objective optimization algorithms. Slide presentation, September 2007. [online] http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2007/CEC-07-Comparison.pdf, 15.4.2012.

[137] Systems Optimization Group, ETH Zürich. PISA, a platform and programming language independent interface for search algorithms, 2007. [online] http://www.tik.ee.ethz.ch/sop/pisa/, 16.11.2007.

[138] Tagawa, K. Multi-objective optimum design of balanced SAW filters using generalized Differential Evolution. *WSEAS Transactions on Systems 8*, 8 (2009), 923–932.

[139] Tagawa, K., Sasaki, Y., and Nakamura, H. Optimum design of balanced SAW filters using multi-objective Differential Evolution. In *Proceedings of the 8th international conference on Simulated Evolution and Learning (SEAL)* (Kanpur, India, December 2010), K. Deb, A. Bhattacharya, N. Chakraborti, P. Chakroborty, S. Das, J. Dutta, S. Gupta, A. Jain, V. Aggarwal, J. Branke, S. Louis, and K. Tan, Eds., vol. 6457 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 466–475.

[140] Tušar, T., and Filipič, B. Differential Evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)* (Matsushima, Japan, March 2007), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 257–271.

[141] Van Veldhuizen, D. A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1999.

[142] Wang, F.-S., and Chiou, J.-P. Differential Evolution for dynamic optimization of differential-algebraic systems. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC 1997)* (Indianapolis, IN, USA, 1997), T. Bäck, Z. Michalewicz, and X. Yao, Eds., IEEE Press, pp. 531–536.

[143] Wang, F.-S., and Sheu, J.-W. Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. *Chemical Engineering Science 55*, 18 (Sept 2000), 3685–3695.

[144] Wolpert, D. H., and Macready, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*, 1 (April 1997), 67–82.

[145] Xue, F., Sanderson, A. C., and Graves, R. J. Pareto-based multi-objective Differential Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)* (Canberra, Australia, Dec 2003), R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, Eds., IEEE Press, Piscataway, NJ, USA, pp. 862–869.

[146] Xue, F., Sanderson, A. C., and Graves, R. J. Modelling and convergence analysis of a continuous multi-objective differential evolution algorithm. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)* (Edinburgh, Scotland, Sept 2005), B. McKay et al., Eds., IEEE Press, pp. 228–235.

[147] Xue, F., Sanderson, A. C., and Graves, R. J. Multi-objective differential evolution – algorithm, convergence analysis, and applications. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)* (Edinburgh, Scotland, Sept 2005), B. McKay et al., Eds., IEEE Press, pp. 743–750.

[148] Zaharie, D. Critical values for the control parameters of Differential Evolution algorithms. In *Proceedings of Mendel 2002, 8th International Conference on Soft Computing* (Brno, Czech Republic, June 2002), R. Matoušek and P. Ošmera, Eds.,

Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, pp. 62–67.

[149] Zaharie, D. Control of population diversity and adaptation in Differential Evolution algorithms. In *Proceedings of Mendel 2003, 9th International Conference on Soft Computing* (Brno, Czech Republic, June 2003), R. Matoušek and P. Ošmera, Eds., Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, pp. 41–46.

[150] Zaharie, D. Multi-objective optimization with adaptive Pareto Differential Evolution. In *Proceedings of Symposium on Intelligent Systems and Applications (SIA 2003)* (Iasi, Romania, Sept 2003), H.-N. Teodorescu, C. Gaindric, and E. Sofron, Eds., Performantica Press.

[151] Zaharie, D., and Petcu, D. Adaptive Pareto Differential Evolution and its parallelization. In *Proceedings of the 5th International Conference on Parallel Processing and Applied Mathematics (PPAM 2003)* (Czestochowa, Poland, Sept 2003), R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Wasniewski, Eds., vol. 3019 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 261–268.

[152] Zhang, Q., and Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 11*, 6 (December 2007), 712–731.

[153] Zhang, Q., Zhou, A., and Jin, Y. RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. *Evolutionary Computation, IEEE Transactions on 12*, 1 (February 2008), 41–63.

[154] Zhong, J.-h., and Zhang, J. Adaptive multi-objective Differential Evolution with stochastic coding strategy. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO 2011)* (Dublin, Ireland, July 2011), N. Krasnogor and P. L. Lanzi, Eds., ACM, pp. 666–672.

[155] Zielinski, K., and Laur, R. Variants of Differential Evolution for multi-objective optimization. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007)* (Honolulu, HI, USA, April 2007), P. Bonissone, C. Coello Coello, and Y. Jin, Eds., IEEE, pp. 91–98.

[156] Zitzler, E. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany, December 1999.

[157] Zitzler, E., Brockhoff, D., and Thiele, L. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)* (Matsushima, Japan, March 2007), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg, pp. 862–876.

[158] Zitzler, E., Deb, K., and Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation 8*, 2 (2000), 173–195.

[159] Zitzler, E., and Künzli, S. Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)* (Birmingham, England, Sept 2004), X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds., vol. 3242 of *Lecture Notes in Computer Science (LNCS)*, Springer, pp. 832–842.

[160] Zitzler, E., and Laumanns, M. Test problem suite: Test problems and test data for multiobjective optimizers, 2003. [online] http://www.tik.ee.ethz.ch/~zitzler/testdata.html, 3.5.2004.

[161] Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of the Third Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2001)* (Athens, Greece, Sept 2002), K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papaliouy, and T. Fogarty, Eds., International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100.

[162] Zitzler, E., and Thiele, L. An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1998.

[163] Zitzler, E., and Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. *IEEE Transactions on Evolutionary Computation 4*, 3 (1999), 257–271.

[164] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation 7*, 2 (April 2003), 117–132.

APPENDIX I

Test Problems

Detailed descriptions of the test problems used in Chapter 3 are given below. All the problems are scalable in respect of the number of decision variables D, however the same values as in the original problem descriptions are used.

Zitzler-Deb-Thiele (ZDT) Test Problems

Zitzler-Deb-Thiele (ZDT) test problems have been described in [158] and [156, pp. 57–59]. The problems used in this thesis are ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [33, pp. 356–360]. They are designed to test the ability of a multi-objective optimization method to handle convexity (ZDT1), non-convexity (ZDT2), discontinuity (ZDT3), multi-modality (ZDT4), and non-uniformity (ZDT6) of the Pareto-optimal front. These problems are formally described below [33, pp. 356–360]:

ZDT1

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = x_1, \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = g\left(1 - \sqrt{f_1/g}\right), \\ \text{subject to} \quad & g = 1 + \tfrac{9}{D-1}\textstyle\sum_{i=2}^{D} x_i, \\ & x_i \in [0,1], D = 30\ . \end{aligned} \tag{I.1}$$

ZDT2

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = x_1, \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = g\left(1 - (f_1/g)^2\right), \\ \text{subject to} \quad & g = 1 + \tfrac{9}{D-1}\textstyle\sum_{i=2}^{D} x_i, \\ & x_i \in [0,1], D = 30\ . \end{aligned} \tag{I.2}$$

ZDT3

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = x_1, \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = g\left(1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)\right), \\ \text{subject to} \quad & g = 1 + \tfrac{9}{D-1}\textstyle\sum_{i=2}^{D} x_i, \\ & x_i \in [0,1], D = 30\ . \end{aligned} \tag{I.3}$$

ZDT4

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = x_1, \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = g\left(1 - \sqrt{f_1/g}\right), \\ \text{subject to} \quad & g = 1 + 10(D-1) + \textstyle\sum_{i=2}^{D}\left({x_i}^2 - 10\cos(4\pi x_i)\right), \\ & x_1 \in [0,1], x_i \in [-5,5], D = 10\ . \end{aligned} \tag{I.4}$$

ZDT6

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1), \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = g\left(1 - (f_1/g)^2\right), \\ \text{subject to} \quad & g = 1 + 9\left(\left(\textstyle\sum_{i=2}^{D} x_i\right)/9\right)^{0.25}, \\ & x_i \in [0,1], D = 10\ . \end{aligned} \tag{I.5}$$

The Pareto-optimal fronts for the problems can be seen in Chapter 3, Figure 3.11. Problems ZDT4 and ZDT6 are also illustrated in Figure 1 of *Publication I*. ZDT4 has many local Pareto-optimal fronts. Solutions shown for ZDT6 are uniformly distributed in the decision variable space. One can notice that the density of solutions across the Pareto-optimal front is non-uniform.

Deb-Thiele-Laumanns-Zitzler (DTLZ) Test Problems

Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems have been described in [39,40]. The problem set consists of nine different problems, which are scalable with respect to the number of objectives but the original definition for the problems is for three objectives [39, 40]. From the total set of problems, the following five problems have been used in this thesis: DTLZ1, DTLZ2, DTLZ4, DTLZ5, and DTLZ7. They differ in the shape of the Pareto-optimal front and diversity preservation. Formal definitions of the problems in a tri-objective form are given below:

DTLZ1

$$\begin{aligned} \text{Minimize} \quad & f_1(\boldsymbol{x}) = 0.5x_1x_2(1+g), \\ \text{Minimize} \quad & f_2(\boldsymbol{x}) = 0.5x_1(1-x_2)(1+g), \\ \text{Minimize} \quad & f_3(\boldsymbol{x}) = 0.5(1-x_1)(1+g), \\ \text{subject to} \quad & g = 100\left(5 + \textstyle\sum_{i=3}^{D}(x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right)\right), \\ & x_i \in [0,1], D = 7\ . \end{aligned} \tag{I.6}$$

DTLZ2

$$\begin{array}{rl} \text{Minimize} & f_1(\boldsymbol{x}) = (1+g)\cos(x_1\pi/2)\cos(x_2\pi/2), \\ \text{Minimize} & f_2(\boldsymbol{x}) = (1+g)\cos(x_1\pi/2)\sin(x_2\pi/2), \\ \text{Minimize} & f_3(\boldsymbol{x}) = (1+g)\sin(x_1\pi/2), \\ \text{subject to} & g = \sum_{i=3}^{D}(x_i - 0.5)^2, \\ & x_i \in [0,1], D = 12\ . \end{array} \tag{I.7}$$

DTLZ4

$$\begin{array}{rl} \text{Minimize} & f_1(\boldsymbol{x}) = (1+g)\cos({x_1}^{\alpha}\pi/2)\cos({x_2}^{\alpha}\pi/2), \\ \text{Minimize} & f_2(\boldsymbol{x}) = (1+g)\cos({x_1}^{\alpha}\pi/2)\sin({x_2}^{\alpha}\pi/2), \\ \text{Minimize} & f_3(\boldsymbol{x}) = (1+g)\sin({x_1}^{\alpha}\pi/2), \\ \text{subject to} & g = \sum_{i=3}^{D}(x_i - 0.5)^2, \\ & \alpha = 100, \\ & x_i \in [0,1], D = 12\ . \end{array} \tag{I.8}$$

DTLZ5

$$\begin{array}{rl} \text{Minimize} & f_1(\boldsymbol{x}) = (1+g)\cos(x_1\pi/2)\cos(\theta), \\ \text{Minimize} & f_2(\boldsymbol{x}) = (1+g)\cos(x_1\pi/2)\sin(\theta), \\ \text{Minimize} & f_3(\boldsymbol{x}) = (1+g)\sin(x_1\pi/2), \\ \text{subject to} & g = \sum_{i=3}^{D}(x_i - 0.5)^2, \\ & \theta = \frac{\pi(1+2gx_2)}{4(1+g)}, \\ & x_i \in [0,1], D = 12\ . \end{array} \tag{I.9}$$

DTLZ7

$$\begin{array}{rl} \text{Minimize} & f_1(\boldsymbol{x}) = x_1, \\ \text{Minimize} & f_2(\boldsymbol{x}) = x_2, \\ \text{Minimize} & f_3(\boldsymbol{x}) = (1+g)\left(3 - \sum_{i=1}^{2}\left[\frac{x_i}{1+g}(1+\sin(3\pi x_i))\right]\right), \\ \text{subject to} & g = 1 + 9/20\sum_{i=3}^{D} x_i, \\ & x_i \in [0,1], D = 22\ . \end{array} \tag{I.10}$$

Pareto-optimal fronts for these problems can be seen in Chapter 3, Figure 3.12.

Constrained Multi-Objective Test Problems

Constrained bi-objective test problems used in Section 3.6 have been taken from [33, pp. 362–367, 453–455].

Spring Design Problem

In the spring design problem, the problem is to design a helical compression spring which has a minimum volume and minimal stress. Objective functions are non-linear and the problem has three decision variables: the number of spring coils x_1 (integer), the wire diameter x_2 (discrete having 42 non-equispaced values), and the mean coil diameter x_3 (real). In addition to the boundary constraints, the problem has eight inequality constraint functions, most of which are non-linear. A formal description of the problem is [36, 75]:

$$
\begin{array}{ll}
\text{Minimize} & f_1(\boldsymbol{x}) = 0.25\pi^2 {x_2}^2 x_3(x_1+2), \\
\text{Minimize} & f_2(\boldsymbol{x}) = \frac{8KP_{max}x_3}{\pi {x_2}^3}, \\
\text{subject to} & g_1(\boldsymbol{x}) = l_{max} - \frac{P_{max}}{k} - 1.05(x_1+2)x_2 \geq 0, \\
& g_2(\boldsymbol{x}) = x_2 - d_{min} \geq 0, \\
& g_3(\boldsymbol{x}) = D_{max} - (x_2 + x_3) \geq 0, \\
& g_4(\boldsymbol{x}) = C - 3 \geq 0, \\
& g_5(\boldsymbol{x}) = \delta_{pm} - \delta_p \geq 0, \\
& g_6(\boldsymbol{x}) = \frac{P_{max}-P}{k} - \delta_w \geq 0, \\
& g_7(\boldsymbol{x}) = S - \frac{8KP_{max}x_3}{\pi {x_2}^3} \geq 0, \\
& g_8(\boldsymbol{x}) = V_{max} - 0.25\pi^2 {x_2}^2 x_3(x_1+2) \geq 0, \\
& x_1 \text{ is integer, } x_3 \text{ is continuous, } x_2 \in \{0.009, 0.0095, 0.0104, 0.0118, \\
& \quad 0.0128, 0.0132, 0.014, 0.015, 0.0162, 0.0173, 0.018, 0.020, \\
& \quad 0.023, 0.025, 0.028, 0.032, 0.035, 0.041, 0.047, 0.054, 0.063, \\
& \quad 0.072, 0.080, 0.092, 0.105, 0.120, 0.135, 0.148, 0.162, 0.177, \\
& \quad 0.192, 0.207, 0.225, 0.244, 0.263, 0.283, 0.307, 0.331, 0.362, \\
& \quad 0.394, 0.4375, 0.5\} \ .
\end{array}
\tag{I.11}
$$

The parameters used are as follows:

$$
\begin{array}{llll}
K = \frac{4C-1}{4C-4} + \frac{0.615x_2}{x_3}, & P = 300 \text{ lb}, & D_{max} = 3 \text{ in}, & k = \frac{G{x_2}^4}{8x_1{x_3}^3}, \\
P_{max} = 1000 \text{ lb}, & \delta_w = 1.25 \text{ in}, & \delta_p = \frac{P}{k}, & l_{max} = 14 \text{ in}, \\
S = 189000 \text{ psi}, & \delta_{pm} = 6 \text{ in}, & d_{min} = 0.2 \text{ in}, & C = D/d, \\
G = 11500000, & V_{max} = 30 \text{ in}^3 \ . & &
\end{array}
\tag{I.12}
$$

Pareto-optimal front approximations for the spring design problem are shown in Figure 2 of *Publication IV*.

BNH

$$
\begin{array}{ll}
\text{Minimize} & f_1(\boldsymbol{x}) = 4{x_1}^2 + 4{x_2}^2, \\
\text{Minimize} & f_2(\boldsymbol{x}) = (x_1-5)^2 + (x_2-5)^2, \\
\text{subject to} & g_1(\boldsymbol{x}) = (x_1-5)^2 + {x_2}^2 \leq 25, \\
& g_2(\boldsymbol{x}) = (x_1-8)^2 + (x_2+3)^2 \geq 7.7, \\
& x_1 \in [0,5], x_2 \in [0,3] \ .
\end{array}
\tag{I.13}
$$

Pareto-optimal front approximations for BNH are shown in Chapter 3, Figure 3.20.

OSY

$$\begin{aligned}
\text{Minimize} \quad & f_1(\boldsymbol{x}) = -\left[25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2\right], \\
\text{Minimize} \quad & f_2(\boldsymbol{x}) = {x_1}^2 + {x_2}^2 + {x_3}^2 + {x_4}^2 + {x_5}^2 + {x_6}^2, \\
\text{subject to} \quad & g_1(\boldsymbol{x}) = x_1 + x_2 - 2 \geq 0, \\
& g_2(\boldsymbol{x}) = 6 - x_1 - x_2 \geq 0, \\
& g_3(\boldsymbol{x}) = 2 - x_2 + x_1 \geq 0, \\
& g_4(\boldsymbol{x}) = 2 - x_1 + 3x_2 \geq 0, \\
& g_5(\boldsymbol{x}) = 4 - (x_3-3)^2 - x_4 \geq 0, \\
& g_6(\boldsymbol{x}) = (x_5-3)^2 + x_6 - 4 \geq 0, \\
& x_1, x_2, x_6 \in [0,10], \quad x_3, x_5 \in [1,5], \quad x_4 \in [0,6] \ .
\end{aligned} \tag{I.14}$$

Pareto-optimal front approximations for BNH are shown in Chapter 3, Figure 3.21.

SRN

$$\begin{aligned}
\text{Minimize} \quad & f_1(\boldsymbol{x}) = 2 + (x_1-2)^2 + (x_2-1)^2, \\
\text{Minimize} \quad & f_2(\boldsymbol{x}) = 9x_1 - (x_2-1)^2, \\
\text{subject to} \quad & g_1(\boldsymbol{x}) = {x_1}^2 + {x_2}^2 \leq 225, \\
& g_2(\boldsymbol{x}) = x_1 - 3x_2 + 10 \leq 0, \\
& x_1, x_2 \in [-20, 20] \ .
\end{aligned} \tag{I.15}$$

Pareto-optimal front approximations for BNH are shown in Chapter 3, Figure 3.22.

TNK

$$\begin{aligned}
\text{Minimize} \quad & f_1(\boldsymbol{x}) = x_1, \\
\text{Minimize} \quad & f_2(\boldsymbol{x}) = x_2, \\
\text{subject to} \quad & g_1(\boldsymbol{x}) = {x_1}^2 + {x_2}^2 - 1 - 0.1\cos\left(16\arctan\tfrac{x_1}{x_2}\right) \geq 0, \\
& g_2(\boldsymbol{x}) = (x_1-0.5)^2 + (x_2-0.5)^2 \leq 0.5, \\
& x_1, x_2 \in [0, \pi] \ .
\end{aligned} \tag{I.16}$$

Pareto-optimal front approximations for BNH are shown in Chapter 3, Figure 3.23.

APPENDIX II

Program Codes for Performance Metrics

Matlab program codes of the performance metrics used in the publications are presented here. The program code for calculating unary hypervolume values in *Publication VII*, has been obtained from [137].

Generational Distance

Generational distance measures the average distance of solutions to the Pareto-optimal front [33, pp. 326–327]. Thus, from each solution, the shortest distance to the Pareto optimal front is measured and then the mean of these distances is calculated. This metric has been used in *Publication II*, *Publication III*, and *Publication VI* with the Euclidean distance ($p = 2$).

```
function [GD,V] = gen_dist(P, Q, p)

% Generational distance: [GD,V] = gen_dist(P, Q, p)
%
%       Generational distance for a set of solutions measures the average
%       distance from the Pareto-optimal front.
%
%       P = Set of solutions representing the global Pareto-optimal front.
%       Q = Set of solutions obtained.
%       p = Parameter for defining L_p distance metric (p = 2 is
%           Euclidean).
%
%       GD = Generational distance.
%       V = Variance of distances.
%
%       More information about the metric:
%
```

```
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 326 - 327.

%       (c) Saku Kukkonen

card = size(Q,1);
D = zeros(1,card);

for i=1:card,

  temp = repmat(Q(i,:),size(P,1),1);
  temp = sum(abs(temp - P).^p,2).^(1/p);
  ind = find(min(temp) == temp);

  D(i) = temp(ind(1));

end;

GD = sum(D.^p)^(1/p)/card;
V = var(D);
```

Error Ratio

Error ratio measures the fraction of solutions that are not Pareto-optimal [33, pp. 324–325]. This metric has been used in *Publication III* with $\delta = 0.01$, which was an experimentally determined value.

```
function ER = error_ratio(P, Q, delta)

% Error ratio: ER = error_ratio(P, Q, delta)
%
%       Error ratio measures the fraction of solutions that are not
%       Pareto-optimal.
%
%       P = Set of solutions representing the global Pareto-optimal front.
%       Q = Set of solutions obtained.
%       delta = Allowed error between obtained and Pareto-optimal solutions.
%
%       ER = Error ratio.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 324 - 325.

%       (c) Saku Kukkonen
```

```
card = size(Q,1);
count = 0;

for i=1:card,

  temp = repmat(Q(i,:),size(P,1),1);
  temp = sqrt(sum((temp - P).^2,2));
  temp = find(temp <= delta);

  if (length(temp) == 0)
     count = count + 1;
  end;

end;

ER = count/card;
```

Spacing

Spacing measures the standard deviation of the distances from each solution to the nearest solution in the obtained non-dominated set [33, pp. 327–328]. This metric has been used in *Publication II*, *Publication III*, *Publication IV*, *Publication V*, *Publication VI*, and *Publication VII*.

```
function S = spacing(Q)

% Spacing: S = spacing(Q)
%
%       Spacing measures the standard deviation of the distances from each
%       solution to the nearest solution in the obtained non-dominated
%       set.
%
%       Q = Set of solutions obtained.
%
%       S = Spacing.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 327 - 328.

%       (c) Saku Kukkonen

card = size(Q,1);
```

```
% For a set of one solution, spacing is not defined.

if (card == 1),
   S = nan;
else
   D = zeros(1,card);

   % Normalization of objective values

   mi = repmat(min(Q),card,1);
   ma = repmat(max(Q),card,1);
   warning off MATLAB:divideByZero
   Q = (Q - mi) ./ (ma - mi);
   warning on MATLAB:divideByZero
   Q(isnan(Q)) = 0;

   for i=1:card,

      temp = repmat(Q(i,:),card-1,1);
      temp = sum(abs(temp - Q([1:i-1 i+1:end],:)),2);
      ind = find(min(temp) == temp);

      D(i) = temp(ind(1));

   end;

   S = sqrt(sum((D - mean(D)).^2)/card);
end;
```

Spread

Spread measures both the uniformity of the obtained non-dominated set and the distance to extreme values of the Pareto-optimal front [33, pp. 328–330]. This metric has been used in *Publication III* with the Euclidean distance ($p = 2$).

```
function Delta = spread(P, Q, p)

% Spread: Delta = spread(P, Q, p)
%
%       Spread measures both uniformity of the obtained non-dominated set
%       and distance to extreme values of the Pareto-optimal front.
%
%       P = Set of solutions representing the global Pareto-optimal front.
%       Q = Set of solutions obtained.
%       p = Parameter for defining L_p distance metric (p = 2 is
%           Euclidean).
```

```
%
%       Delta = Spread.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 328 - 330.

%       (c) Saku Kukkonen

[card,M] = size(Q);
d_e = 0;

[foo, I1] = min(Q,[],1);
[foo, I2] = min(P,[],1);

for i=1:M,
   d_e = d_e + sum(abs(Q(I1(i),:) - P(I2(i),:)).^p)^(1/p);
end;

temp = sum(abs(diff(sortrows(Q,1))).^p,2).^(1/p);
d_avg = mean(temp);
d_std = std(temp);
temp = sum(abs(temp - d_avg));

Delta = (d_e + temp) / (d_e + (card-1)*d_avg);
```

Maximum Spread, normalized version

The normalized version of maximum spread measures the distance between the extreme solutions in the obtained set in relation to the distance between extreme solutions at the Pareto-optimal front [33, pp. 330–331]. This metric has been used in *Publication III*, and *Publication VII* with the Euclidean distance ($p = 2$).

```
function D = max_spread(P, Q, p)

% Maximum spread: D = max_spread(P, Q, p)
%
%       Normalized maximum spread measures the distance between extreme
% solutions in the obtained set in relation to the distance
% between extreme solution in the Pareto-optimal front.
%
%       P = Set of solutions representing the global Pareto-optimal front.
%       Q = Set of solutions obtained.
%       p = Parameter for defining L_p distance metric (p = 2 is
%           Euclidean).
```

```
%
%       D = Normalized maximum spread.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 330 - 331.

%       (c) Saku Kukkonen

[card,M] = size(Q);

% For a set of one solution, maximum spread is zero.

if (card==1)
   D = 0;
else
   D = (1/M*sum(((max(Q) - min(Q))./(max(P) - min(P))).^p))^(1/p);
end;
```

Set Coverage Metric

Set coverage metric, $C(A, B)$, between two non-dominated sets 'A' and 'B' measures the fraction of members of 'B' that are dominated by members of 'A'. This metric has been used in *Publication IV*.

```
function C = c_metric(A,B)

% Set coverage metric: C = c_metric(A,B)
%
%       Set coverage metric between two non-dominated sets A and B
%       measures the fraction of members of B that are dominated by
%       members of A. Minimization of objectives is assumed.
%
%       C = Set coverage metric.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 325 - 326.

%       (c) Saku Kukkonen

nA = size(A,1);
nB = size(B,1);
```

```
% Every row of A is replicated nB times

A = A(reshape(repmat(1:nA,nB,1),1,nA*nB),:);

% B is replicated nA times

B = repmat(B,nA,1);

% When A and B are compared, each element of A is compared with each
% element of B

%   |----  sum of elements B, which are dominated by A  ---------------|
%        |----  dominated elements in B  ----------------------------|
%           |----  dominated element for each element of A  ----|
%                  |----  a in A dominates b in B?  ----|

C = sum(max(reshape(min((A <= B),[],2) & max((A < B),[],2),nB,nA),[],2))/nB;
```

Binary Hypervolume Indicator

Hypervolume, $HV(A, B)$, between two non-dominated sets of solutions A and B calculates the hypervolume of the objective space which is dominated by A but not B. This metric has been used in *Publication IV* with $numsamp = 50000$, which was an experimentally determined value.

```
function V = v_metric(A,B,numsamp)

% Binary hypervolume indicator: V = v_metric(A,B,numsamp)
%
%       Hypervolume between two non-dominated sets of solutions 'A'
%       and 'B' calculates the hypervolume of the objective space which
%       is dominated by 'A' but not 'B'.
%
%       numsamp = number of samples in Monte Carlo sampling.
%
%       H = Hypervolume.
%
%       More information about the metric:
%
%       K. Deb, Multi-Objective Optimization using Evolutionary
%       Algorithms, 2001, pp. 332 - 333.
%
%       Jonathan Fieldsend, Richard M. Everson, Sameer Singh, Using
%       Unconstrained Elite Archives for Multiobjective Optimization, IEEE
%       Transactions on Evolutionary Computation, 7(3), June 2003, pp. 305
%       - 323.
```

```
%
%       E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert
%       de Fonseca, Performance Assessment of Multiobjective Optimizers: An
%       Analysis and Review, IEEE Transactions on Evolutionary
%       Computation, 7 (2), April 2003, pp. 117 - 132.

%       (c) Saku Kukkonen


if (nargin < 3),
   numsamp = 50000;
end;

[nA, m] = size(A);
nB = size(B,1);

% Low and high coordinates of the hypercube which encloses A and B

low = min([A;B]);
high = max([A;B]);

% Random samples in hypercube

R = repmat(low,numsamp,1) + rand(numsamp,m).*repmat(high-low,numsamp,1);

% Every row of A is replicated numsamp times

A = A(reshape(repmat(1:nA,numsamp,1),1,nA*numsamp),:);

% R is replicated nA times

R = repmat(R,nA,1);

% When A and R are compared, each element of A is compared with each
% element of R

% ind defines indices of elements in R, which are dominated by A

ind = find(max(reshape(min((A <= R),[],2) & ...
           max((A < R),[],2),numsamp,nA),[],2));

% dA defines number of elements in R which are dominated by A

dA = length(ind);

if (dA == 0),
   V = 0;
```

```
else

   % Each element of B is compared with each element of R which is
   % dominated by A

   B = B(reshape(repmat(1:nB,dA,1),1,nB*dA),:);
   R = repmat(R(ind,:),nB,1);

   % dB defines number of elements in R which are dominated by A and B

   dB = sum(max(reshape(min((B <= R),[],2) & max((B < R),[],2),dA,nB),[],2));

   V = (dA - dB)/numsamp;

end;
```

APPENDIX III

Publications

Publication I

Kukkonen, S., Lampinen, J.,
Comparison of Generalized Differential Evolution Algorithm to Other Multi-Objective Evolutionary Algorithms

Reprinted, with permission, from
Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004), Jyväskylä, Finland, July, 2004, 20 pages.

European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2004
P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer (eds.)
Jyväskylä, 24–28 July 2004

COMPARISON OF GENERALIZED DIFFERENTIAL EVOLUTION TO OTHER MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

Saku Kukkonen and Jouni Lampinen

Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20, FIN-53851 Lappeenranta, Finland
e-mail: Saku.Kukkonen@lut.fi, Jouni.Lampinen@lut.fi

Key words: multi-objective optimization, Pareto-optimization, constraints, evolutionary algorithms, differential evolution

Abstract. *In this paper an Evolutionary Algorithm, Differential Evolution, and its extension for constrained multi-objective (Pareto-)optimization, Generalized Differential Evolution, are described. Performance of Generalized Differential Evolution is tested with a set of five benchmark multi-objective test problems. Suitable control parameter values for these test problems are surveyed and the results are compared numerically with other multi-objective evolutionary algorithms including the Strength Pareto Evolutionary Algorithm and the Non-dominated Sorting Genetic Algorithm. Several metrics commonly used in the literature are applied to measure convergence to the Pareto-optimal front and diversity of the obtained solution. The results are suggesting that the performance of Generalized Differential Evolution is well comparable to the performance of the compared multi-objective evolutionary algorithms.*

1 INTRODUCTION

Many situations in engineering and economics deal with optimization. One may want to optimize, *e.g.*, manufacturing processes, shape of products, and number of different products to be manufactured. Typical goals are minimizing costs, maximizing profits, and improving performance. Several natural aspects limits feasible solutions, *e.g.*, resources may cause limitations and/or the number of products cannot be a negative number.

Optimization is an intensively studied problem field in mathematics. However, functions to be optimized in traditional mathematics are relatively simple, *e.g.*, continuous, convex, unimodal, differentiable, *etc.*, yet functions to be optimized in practice are often far more complicated, *e.g.*, discontinuous, non-convex, multi-modal, non-differentiable, *etc.* In these cases different stochastic optimization methods have shown their effectiveness.

Most optimization research deals with single-objective optimization problems. The basic nature of many optimization problems is, however, multi-objective and these problems are usually first converted to single-objective problems. Single-objective problems are commonly considered easier to solve but conversion from multi-objective problem to single-objective problem requires some *a priori* knowledge which is not necessarily available or which is hard to determine, *e.g.*, the relative importance of each individual sub-objective. For this reason interest exists in solving multi-objective problems in multi-objective form.

Several extensions of Differential Evolution (DE) for multi-objective optimization have already been proposed. Most of these methods use a non-dominated sorting for reproduction in each generation and some distance metric to prevent crowding. Chang *et.al.* introduce an extension of DE for solving multi-objective optimization problems [1]. This method uses selection of non-dominated solutions for further generations and uses a distance metric to maintain diversity. Abbass *et.al.* propose several extensions of DE which modifies the original algorithm in several ways, *e.g.*, the use of only non-dominated solutions for reproduction in each generation and a distance metric to prevent crowding [2–4]. Madavan describes an extension which uses the non-dominated sorting and a ranking selection procedure [5]. Babu and Jehan apply DE for multiple objectives using a penalty function and a weighted sum approaches [6]. Graves *et.al.* and Xue use the non-dominated sorting for reproduction and a distance measure to prevent crowding [7, 8].

This paper continues with the following parts: In Section 2 the concept of multi-objective optimization with constraints is handled briefly. Section 3 describes Differential Evolution algorithm and Section 4 describes its extension for constrained multi-objective optimization. Section 5 describes an empirical evaluation of the extension and finally conclusions are given in Section 6.

2 MULTI-OBJECTIVE OPTIMIZATION WITH CONSTRAINTS

Many practical problems have multiple objectives. For example, designing a wing of an aircraft may have objectives such as maximizing strength, minimizing weight, minimizing manufacturing costs, maximizing lifting force, minimizing drag, *etc.* Multiple objectives are almost always more or less conflicting.

Several aspects cause constraints to problems. In the previous example of the wing, the thickness of the metal parts used must be a positive number, shape limitations exist, some parts are available only in some predefined standard sizes, *etc.* Constraints can be divided into box or boundary constraints and constraint functions. Boundary constraints are used when the value of some optimized variable is limited to some range and constraint functions are representing more complicated constraints which are expressed as functions.

Multi-objective problems are often converted to single-objective problems by predefining weighting factors for different objectives, expressing the relative importance of each objective. However, this is impossible in many cases because a decision maker does not necessarily know beforehand how he/she wants to weight different objectives. Thus, a more convenient way is to keep multiple objectives of multi-objective problems and try to solve them in this form even though this may be harder to do in practice. Optimizing several objectives simultaneously without articulating the relative importance for each objective *a priori*, is often called as Pareto-optimization [9]. An obtained solution is Pareto-optimal if none of the objectives cannot be improved without impairing at least one other objective [10, p. 11–12]. If the obtained solution can be improved in such way that at least one objective improves and other objectives do not decline, then the new solution dominates the original solution. A set of Pareto-optimal solutions form a Pareto-optimal front. Approximation of the Pareto-optimal front is called as a set of non-dominated solutions because the solutions of this set are not dominating each others in the space of objective functions. From the set of the non-dominated solutions the decision maker may select one which has suitable values for different objectives. This can be viewed as *a posteriori* articulation of decision-makers preferences concerning the relative importance of each objective.

A mathematically constrained multi-objective optimization problem can be presented in the form [10, p. 37]

$$\begin{aligned} &\text{minimize} \quad \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_K(\vec{x})\} \\ &\text{subject to} \quad \vec{x} \in S = \{\vec{x} \in \mathbf{R}^D | \vec{g(x)} = (g_1(\vec{x}), g_2(\vec{x}), \ldots, g_M(\vec{x}))^T \leq \vec{0}\}. \end{aligned} \tag{1}$$

Thus, there are K functions to be optimized and M constraint functions.

Maximization problems can be converted to minimization problems by multiplying the objective function by -1 because $\max f_k(\vec{x}) \Leftrightarrow \min -f_k(\vec{x})$. All constraints can be converted to the form $g_j(\vec{x}) \leq 0$ in the following way: $g_j(\vec{x}) \geq 0 \Leftrightarrow -g_j(\vec{x}) \leq 0$, $g_j(\vec{x}) = 0 \Leftrightarrow g_j(\vec{x}) \leq 0 \wedge -g_j(\vec{x}) \leq 0$. Boundary constraints can be presented also in the form of constraint functions, *e.g.*, $a \leq x_i \leq b \Leftrightarrow a - x_i \leq 0 \wedge x_i - b \leq 0$. Thereby the formulation in Eq. 1 is without loss of generality.

The major part of earlier mathematical research has concentrated on optimization problems where the functions are linear, differentiable, convex, or otherwise mathematically well behaving. However, in practical problems objective functions are often nonlinear, non-differentiable, discontinuous, multi-modal, *etc.* and no presumptions can be made about their behavior. Variables may also be integers or discrete instead of being continuous. Most traditional optimization methods cannot handle such complexity or do not perform in these cases in which the assumptions they are based on do not hold. For such problems stochastic optimization methods such as Simulated Annealing (SA) and Evolution Algorithms (EAs) have been demonstrated to be effective because they do not rely assumptions concerning the objective and constraint functions.

3 DIFFERENTIAL EVOLUTION

The Differential Evolution (DE) algorithm [11, 12] [13, pp. 79–108] belongs to the family of Evolution Algorithms and was introduced by Storn and Price in 1995 [14]. Design principles in DE were simplicity, efficiency, and use of floating-point encoding instead of binary numbers, which is usual way of coding in Genetic Algorithms (GA).

Like in a typical EA, the idea in DE is to have some random initial population which is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criteria for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides appropriate stopping condition.

3.1 Initialization of population

Values for the initial population in DE are typically drawn from uniform distribution. Formally this can be presented as [13, p. 81]:

$$\begin{aligned} &P_G = \{\vec{x}_{1,G}, \vec{x}_{2,G}, \ldots, \vec{x}_{NP,G}\}, \quad \vec{x}_{i,G} = x_{j,i,G} \\ &x_{j,i,G=0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ &i = 1, 2, \ldots, NP, \quad NP \geq 4, \quad j = 1, 2, \ldots, D \end{aligned} \tag{2}$$

In this representation P_G denotes a population after G generations (0 is an initial generation), $\vec{x}_{i,G}$ denotes an object variable vector (or individual) of the population, and $rand_j[0,1]$ denotes an uniformly distributed random variable in the range $[0,1]$. Terms $x_j^{(lo)}$ and $x_j^{(hi)}$ denote lower and upper parameter bounds, respectively. The size of the population is denoted by NP and the dimension of objective vectors is denoted by D.

Other ways of initialization also exist, *e.g.*, if some knowledge exist about the position of the optimum, part of the initial population may be initialized around the possible position of the optimum using normal distribution.

3.2 Mutation and crossover

DE goes through each objective vector $\vec{x}_{i,G}$ of the population and creates a corresponding trial vector $\vec{u}_{i,G}$ as follows [13, p. 82]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \\
&\text{(randomly selected, except: } r_1 \neq r_2 \neq r_3 \neq i) \\
&j_{rand} = \text{int}\left(rand_i[0,1) \cdot D\right) + 1 \\
&\text{for}(j = 1; j \leq D; j = j + 1) \\
&\{ \\
&\quad \text{if}(rand_j[0,1) < CR \vee j = j_{rand}) \\
&\quad\quad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\
&\quad \text{else} \\
&\quad\quad u_{j,i,G} = x_{j,i,G} \\
&\}
\end{aligned}
\tag{3}
$$

Indices r_1, r_2, and r_3 are mutually different and drawn from the set of the population indices. Both CR and F are user defined control parameters for the DE algorithm and they remain fixed during the whole execution of the algorithm. Parameter CR, controlling the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors instead of from the old objective vector $\vec{x}_{i,G}$. Parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$. The condition "$j = j_{rand}$" is to make sure that at least one element is different compared to elements of the old population member.

The difference between two randomly chosen vectors $(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$ defines magnitude and direction of mutation. When the difference is added to a third randomly chosen vector $\vec{x}_{r_3,G}$, this corresponds mutation of this third vector. The basic idea of DE is that mutation is self-adaptive to the objective function space and to the current population. At the beginning of generations a magnitude of mutation is large because vectors in the population are far away in the search space. When evolution proceeds and population converges, the magnitude of mutation gets smaller. The self-adaptive mutation of DE permits to perform global search.

3.3 Selection

After each mutation and crossover operation the trial vector $\vec{u}_{i,G}$ is compared to the old objective vector $\vec{x}_{i,G}$. If the trial vector has equal or lower cost value, then it replaces the old vector. This can be presented as follows [13, p. 82]:

$$
\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if} \quad f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}
\tag{4}
$$

The average cost value of the population will never increase, because the trial vector replaces the old vector only if it has equal or lower cost value.

Overall presentation of the whole DE/*rand*/*1*/*bin* algorithm is presented in Eq. 5 [13, p. 83]. Several variations of the basic DE algorithm exist and this is a reason for complicated notation. In the notation DE/x/y/z x indicates how the mutated vector is selected (it could be selected also to be *best* among the current population), y indicates number of vector differences used in the mutation, and z indicates the way the old vector and the trial vector are recombined (an alternative exponential recombination procedure is also mentioned in the literature [13, p. 98]).

$$
\begin{aligned}
&\text{Input :} D, G_{max}, NP \geq 4, F \in (0, 1+), CR \in [0,1], \text{and initial bounds: } \vec{x}^{(lo)}, \vec{x}^{(hi)} \\
&\text{Initialize :} \begin{cases} \forall i \leq NP \wedge \forall j \leq D : x_{j,i,G=0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)} \right) \\ i = \{1, 2, \ldots, NP\}, j = \{1, 2, \ldots, D\}, G = 0, rand_j[0,1] \in [0,1] \end{cases}
\end{aligned}
$$

$$
\begin{cases}
\text{While } G < G_{max} \\
\quad \forall i \leq NP \begin{cases}
\text{Mutate and recombine:} \\
r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \text{randomly selected,} \\
\qquad \text{except: } r_1 \neq r_2 \neq r_3 \neq i \\
j_{rand} \in \{1, 2, \ldots, D\}, \text{randomly selected for each } i \\
\\
\forall j \leq D, u_{j,i,G} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \qquad \text{if} \quad rand_j[0,1) < CR \vee j = j_{rand} \\ x_{j,i,G} \quad \text{otherwise} \end{cases} \\
\text{Select :} \\
\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if} f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}
\end{cases} \\
G = G + 1
\end{cases}
\tag{5}
$$

4 GENERALIZED DIFFERENTIAL EVOLUTION

Several extensions of DE for multi-objective optimization exists as well for constrained optimization [1–8, 15–18]. The approach presented in this paper combines these by modifying the selection operation of the basic DE algorithm [19–22]. Compared to other DE extensions for multi-objective optimization, this approach makes DE suitable for constrained multi-objective optimization with minimum changes to the original algorithm. The modified selection operation for M constraint and K objective functions is presented

formally in Eq. 6 [21].

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if} \begin{cases} \begin{cases} \exists j \in \{1,\ldots,M\} : g_j(\vec{u}_{i,G}) > 0 \\ \wedge \\ \forall j \in \{1,\ldots,M\} : g'_j(\vec{u}_{i,G}) \leq g'_j(\vec{x}_{i,G}) \end{cases} \\ \vee \\ \begin{cases} \forall j \in \{1,\ldots,M\} : g_j(\vec{u}_{i,G}) \leq 0 \\ \wedge \\ \exists j \in \{1,\ldots,M\} : g_j(\vec{x}_{i,G}) > 0 \end{cases} \\ \vee \\ \begin{cases} \forall j \in \{1,\ldots,M\} : g_j(\vec{u}_{i,G}) \leq 0 \wedge g_j(\vec{x}_{i,G}) \leq 0 \\ \wedge \\ \forall k \in \{1,\ldots,K\} : f_k(\vec{u}_{i,G}) \leq f_k(\vec{x}_{i,G}) \end{cases} \end{cases} \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \tag{6}$$

where $g'_j(\vec{x}_{i,G}) = \max(g_j(\vec{x}_{i,G}), 0)$ and $g'_j(\vec{u}_{i,G}) = \max(g_j(\vec{u}_{i,G}), 0)$ are representing the constraint violations

The selection rule given in Eq. 6 selects the trial vector $\vec{u}_{i,G}$ to replace the old vector $\vec{x}_{i,G}$ in three cases:

1. Both the trial vector $\vec{u}_{i,G}$ and the old vector $\vec{x}_{i,G}$ violate at least one constraint but the trial vector does not violate any of the constraints more than the old vector.

2. The old vector $\vec{x}_{i,G}$ violates at least one constraint whereas the trial vector $\vec{u}_{i,G}$ is feasible.

3. Both vectors are feasible and the trial vector $\vec{u}_{i,G}$ has less or equal cost value for each objective than the old vector $\vec{x}_{i,G}$.

Otherwise the old vector $\vec{x}_{i,G}$ is preserved.

The basic idea in the selection rule is that the trial vector $\vec{u}_{i,G}$ is required to dominate the compared old population member $\vec{x}_{i,G}$ in constraint violation space or in objective function space, or at least provide an equally good solution as $\vec{x}_{i,G}$. The principle is effectively rather similar to the method described in [23, pp. 131–132] even though the formulation is different. In this other method selection is based on the value of a penalized objective function $F(\vec{x})$ [24]:

$$F(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if } \vec{x} \text{ is feasible} \\ f_{max} + \sum_{j=1}^{M} \max(g_j(\vec{x}), 0) & \text{otherwise} \end{cases} \tag{7}$$

Here, f_{max} denotes the objective function value of the worst feasible solution in the population. The selection rules given in Eq. 6 and based on Eq. 7 perform in the same way

when two feasible solutions are compared or when a feasible solution is compared to an infeasible solution. In the case of two infeasible solutions, the selection rule in Eq. 6 compares solutions based on dominance of the constraint violations whereas the selection method based on Eq. 7 compares solutions based on a sum of the constraint violations. The selection method based on Eq. 7 needs search of the objective function value of the worst feasible solution in the population, calculation of all constraint function values in the case of infeasible solution, and it also permits worsening of individual constraint function values because sums of the constraint violations are compared instead of individual constraint violations. The selection method based on Eq. 7 may also need normalization of different constraints in the case of different order of magnitude whereas this is not needed in the selection rule in Eq. 6.

The selection rule in Eq. 6 can be implemented in such a way that the number of function evaluations is reduced because not always all the constraints and objectives need to be evaluated, *e.g.*, inspecting constraint violations (even one constraint) is often enough to determine which vector to select for the next generation [21, 22].

One should note that the selection rule in Eq. 6 handles any number M of constraints and any number K of objectives, including cases $M = 0$ (unconstrained problem) and $K = 0$ (constraint satisfaction problem). When $M = 0$ and $K = 1$, the selection rule is identical to the selection rule of the basic DE algorithm. Because the described selection method extends DE for constrained multi-objective optimization and the basic DE algorithm is a special case, the method with the selection rule described is named *Generalized Differential Evolution (GDE)*.

After the selected number of generations the final population presents a solution for the optimization problem. The non-dominated solutions can be separated from the final population if desired. There is no sorting of non-dominated solutions during the optimization process or explicit mechanism for maintaining diversity of solutions.

5 EXPERIMENTS

GDE was implemented in C and tested with a set of five multi-objective benchmark problems described in [25] and [26, pp. 57–59]. These problems are known as ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [23, pp. 356–360]. They are designed to test the ability of a multi-objective optimization method to handle convexity (ZDT1), non-convexity (ZDT2), discontinuity (ZDT3), multi-modality (ZDT4), and non-uniformity (ZDT6) of the Pareto-optimal front. Preliminary tests with these problems and problems including constraint functions are reported in [27, 28].

$$\text{ZDT1}: \begin{cases} \text{Minimize} & \\ f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}) &= g \times \left(1 - \sqrt{f_1/g}\right) \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{subject to} & \\ & x_i \in [0,1], n = 30 \end{cases} \tag{8}$$

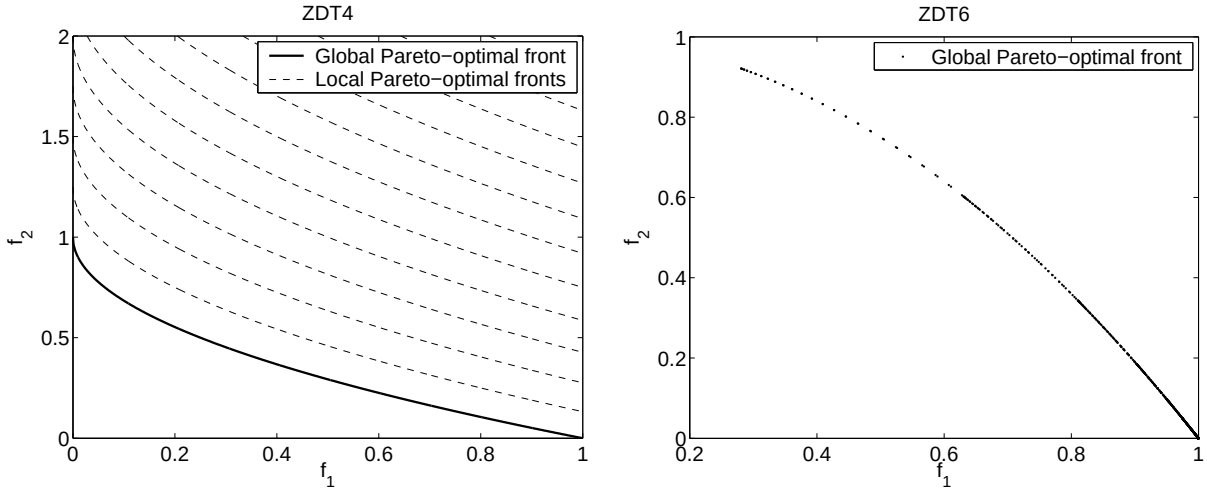
$$\text{ZDT2}: \begin{cases} \text{Minimize} & \\ f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}) &= g \times (1 - (f_1/g)^2) \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{subject to} & \\ & x_i \in [0,1], n = 30 \end{cases} \tag{9}$$

$$\text{ZDT3}: \begin{cases} \text{Minimize} & \\ f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}) &= g \times \left(1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)\right) \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{subject to} & \\ & x_i \in [0,1], n = 30 \end{cases} \tag{10}$$

$$\text{ZDT4}: \begin{cases} \text{Minimize} & \\ f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}) &= g \times \left(1 - \sqrt{f_1/g}\right) \\ g &= 1 + 10(n-1) + \sum_{i=2}^{n} \left({x_i}^2 - 10\cos(4\pi x_i)\right) \\ \text{subject to} & \\ & x_1 \in [0,1], x_i \in [-5,5], n = 10 \end{cases} \tag{11}$$

$$\text{ZDT6}: \begin{cases} \text{Minimize} & \\ f_1(\vec{x}) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(\vec{x}) &= g \times (1 - (f_1/g)^2) \\ g &= 1 + 9\left(\left(\sum_{i=2}^{n} x_i\right)/9\right)^{0.25} \\ \text{subject to} & \\ & x_i \in [0,1], n = 10 \end{cases} \tag{12}$$

To illustrate two of these problems, a partial search region with the global Pareto-optimal front for ZDT4 and a Pareto-optimal front for ZDT6 are presented in Figure 1. Solution for ZDT6 contains points which are uniformly distributed in the decision variable space. One can notice that the density of solutions across the Pareto-optimal front is non-uniform.



(a) (b)

Figure 1: a) The partial search region near the global Pareto-optimal front for ZDT4 b) global Pareto-optimal front for ZDT6.

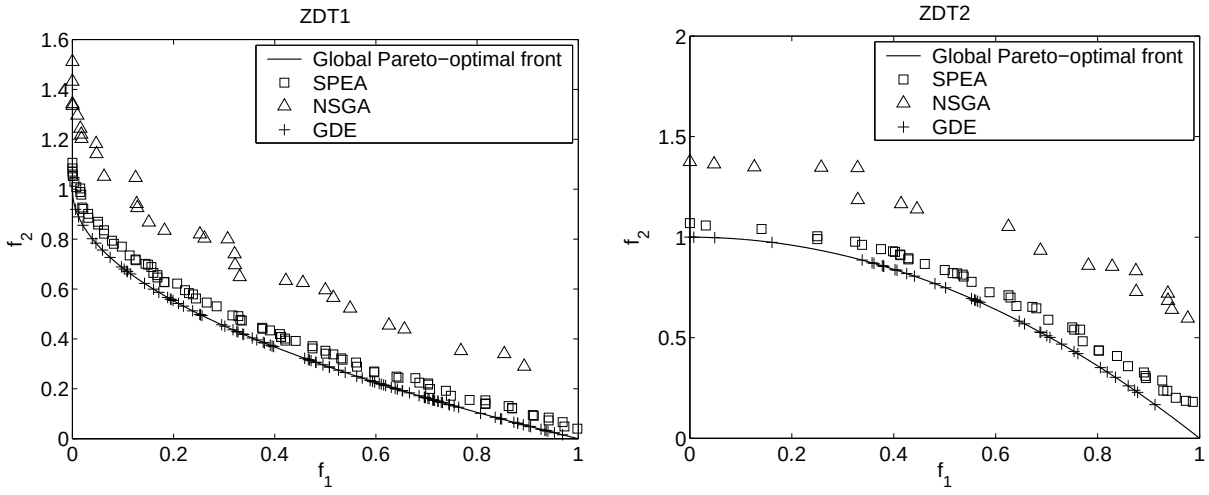
Boundary constraint violations were handled here according to the following rule:

$$u_{j,i,G} = \begin{cases} x_j^{(lo)} & \text{if} \quad u_{j,i,G} < x_j^{(lo)} \\ x_j^{(hi)} & \text{if} \quad u_{j,i,G} > x_j^{(hi)} \\ u_{j,i,G} & \text{otherwise} \end{cases} \tag{13}$$

5.1 Experimental results and discussions

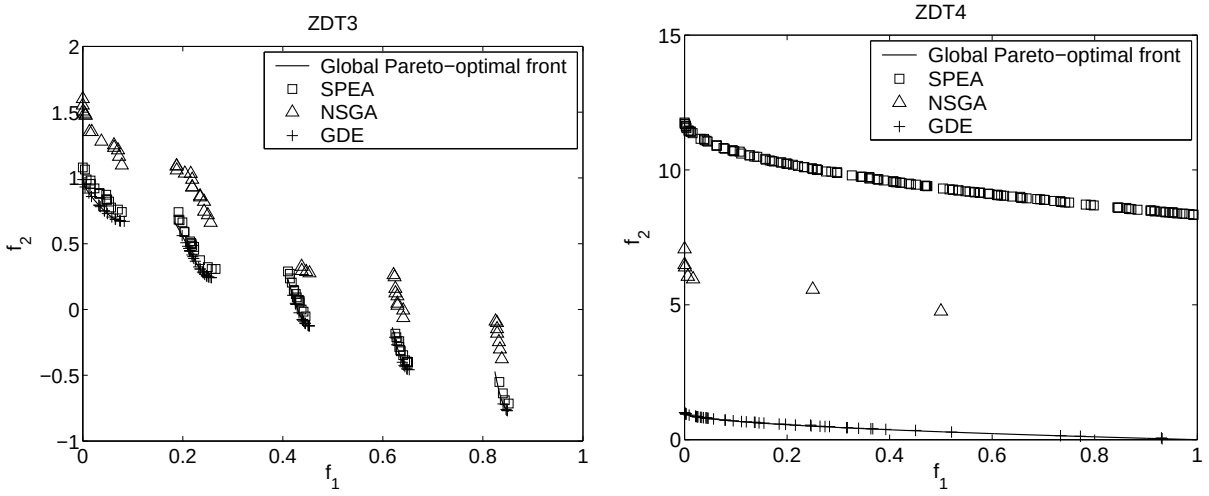
In all the test problems the size of the population was 100, the number of generations was 250, and the control parameter values were $CR = 0.10$, $F = 0.10$. In preliminary tests different control parameter values in the range $CR \in [0, 1]$ and $F \in [0, 5]$ with a resolution of 0.05 were tested and a suitable crossover rate and mutation factor was visually thereby approximately determined. The results for GDE solving the multi-objective benchmark problems are shown in Figures 2–4, where results obtained with the Strength Pareto Evolutionary Algorithm (SPEA) [29] and the Non-dominated Sorting Genetic Algorithm (NSGA) [23, pp. 209–218] are also shown with known global Pareto-optimal fronts. SPEA and NSGA were selected for comparison because of their good performance in previous comparison tests with the other multi-objective optimization methods [25] and since both are well known within the multi-objective optimization community. The results for SPEA, NSGA, and GDE given in Figures 2–4 are after one run, and solutions shown for GDE contain the non-dominated members of the final population.

The tests for GDE were repeated 30 times with different seeds of the random number generator and these results were compared with the corresponding results of SPEA,



(a) (b)

Figure 2: Global Pareto-optimal front and solutions obtained with SPEA, NSGA, and GDE for a) ZDT1 b) ZDT2.



(a) (b)

Figure 3: Global Pareto-optimal front and solutions obtained with SPEA, NSGA, and GDE for a) ZDT3 b) ZDT4.

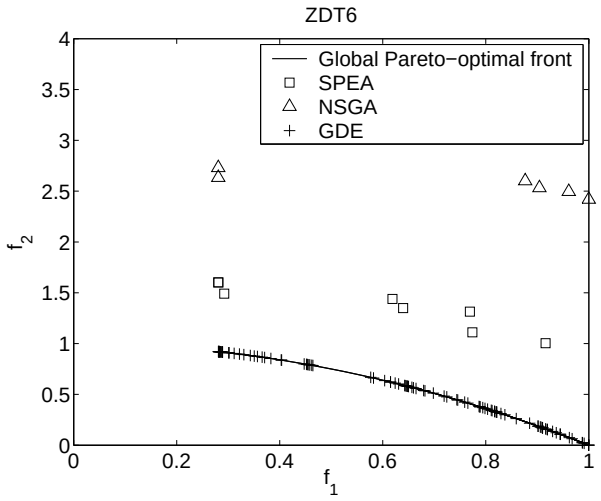


Figure 4: Global Pareto-optimal front and solutions obtained with SPEA, NSGA, and GDE for ZDT6.

NSGA, the Fonseca and Fleming's multi-objective GA (FFGA) [23, pp. 200–209], the Niched-Pareto Genetic Algorithm (NPGA) [23, pp. 218–223], and the Vector Evaluated Genetic Algorithm (VEGA) [23, pp. 179–188]. The results for these MOEAs were obtained from the Internet [30]. From the methods in comparison, GDE and SPEA are elitist MOEAs and the rest methods are non-elitist MOEAs.

Closeness to the Pareto-optimal front was measured with an error ratio (ER) and a generational distance (GD) [23, pp. 324–327]. Diversity of the obtained solution was measured using spacing (S), spread (Δ), and maximum spread (D) metrics [23, pp. 328–331]. Smaller values for the error ratio, generational distance, spacing, and spread are preferable. The optimal value for the maximum spread is 1. Results of the MOEAs were compared against each other using a set coverage $\mathcal{C}$ metric [26] and a $\mathcal{V}$ measure [31, 32]. The $\mathcal{C}(\mathcal{A}, \mathcal{B})$ metric measures the fraction of members of $\mathcal{B}$ that are dominated by members of $\mathcal{A}$. The $\mathcal{V}(\mathcal{A}, \mathcal{B})$ measures the fraction of the volume of the minimal hypercube containing both fronts that is dominated by members of $\mathcal{A}$ but is not dominated by members of $\mathcal{B}$. Greater values for the $\mathcal{C}$ and the $\mathcal{V}$ metrics are desirable.

Average execution times and average numbers of needed function evaluations of GDE for the benchmark problems are reported in Table 1. Tables 2–6 contain the performance measurements for different MOEAs solving the benchmark problems. Solutions of GDE contained the non-dominated members of the final population.

In the most of the test cases GDE has more non-dominated solution members than the other MOEAs in this comparison. The results show that GDE converged closer to the true Pareto-optimal front than the other MOEAs but the diversity metrics show that obtained solution is not optimally diverse. Even though GDE has members in the final solution close to the true Pareto-optimal fronts, the end points of the Pareto-optimal fronts are not exactly reached and the distribution of solutions could be better. It was

		Number of function evaluations	
	Execution time	f_1	f_2
ZDT1	1.019 (0.01) s	25100 (0.0)	23474.2 (32.4)
ZDT2	1.019 (0.01) s	25100 (0.0)	23558.9 (69.0)
ZDT3	1.041 (0.01) s	25100 (0.0)	23490.0 (38.7)
ZDT4	0.547 (0.01) s	25100 (0.0)	22732.3 (41.2)
ZDT6	0.570 (0.01) s	25100 (0.0)	22184.2 (93.1)

Table 1: Average execution time and average number of needed function evaluations (standard deviations in parenthesis) of GDE for the multi-objective benchmark test problems. All the tests were run on a Sun Sparc Ultra2.

	$\aleph$	ER	GD	S	Δ	D
FFGA	25.8(5.8)	1.000(0.000)	0.280(0.047)	0.043(0.013)	0.831(0.060)	2.178(0.306)
NPGA	21.7(4.6)	1.000(0.000)	0.230(0.025)	0.049(0.014)	0.818(0.039)	1.439(0.194)
VEGA	22.3(4.7)	1.000(0.000)	0.175(0.020)	0.057(0.020)	0.805(0.044)	1.236(0.193)
SPEA	72.4(8.5)	0.143(0.109)	0.005(0.001)	0.013(0.002)	0.573(0.037)	1.006(0.014)
NSGA	43.4(6.0)	1.000(0.000)	0.037(0.004)	0.024(0.005)	0.689(0.053)	1.079(0.038)
GDE	99.9(0.4)	0.000(0.000)	0.000(0.000)	0.012(0.002)	0.761(0.044)	0.947(0.023)

$\mathcal{C}(\text{row}, \text{column})$

FFGA	0.437(0.251)	0.035(0.076)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.535(0.221)	NPGA	0.050(0.093)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.914(0.101)	0.900(0.143)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	SPEA	1.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.000(0.000)	NSGA	0.000(0.000)
0.839(0.066)	0.860(0.057)	0.838(0.080)	0.936(0.036)	0.920(0.038)	GDE

$\mathcal{V}(\text{row}, \text{column})$

FFGA	0.033(0.027)	0.002(0.003)	0.000(0.000)	0.000(0.000)	0.001(0.001)
0.037(0.030)	NPGA	0.004(0.009)	0.000(0.000)	0.000(0.000)	0.001(0.001)
0.123(0.049)	0.180(0.068)	VEGA	0.000(0.000)	0.000(0.000)	0.001(0.002)
0.331(0.051)	0.460(0.046)	0.388(0.044)	SPEA	0.187(0.018)	0.000(0.001)
0.279(0.050)	0.395(0.048)	0.303(0.040)	0.000(0.000)	NSGA	0.001(0.001)
0.337(0.050)	0.469(0.046)	0.401(0.047)	0.049(0.006)	0.216(0.015)	GDE

Table 2: Means of the solution cardinality ($\aleph$), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), maximum spread (D), $\mathcal{C}$, and $\mathcal{V}$ of MOEAs for ZDT1. Standard deviations are in parenthesis.

	ℵ	ER	GD	S	Δ	D
FFGA	15.2(4.8)	1.000(0.000)	0.522(0.069)	0.080(0.040)	0.884(0.044)	1.711(0.552)
NPGA	9.8(2.4)	1.000(0.000)	0.333(0.043)	0.109(0.046)	0.916(0.053)	0.828(0.103)
VEGA	2.9(1.3)	1.000(0.000)	0.450(0.111)	0.244(0.231)	0.920(0.096)	0.369(0.212)
SPEA	43.5(9.1)	0.872(0.196)	0.011(0.003)	0.026(0.007)	0.645(0.048)	0.944(0.021)
NSGA	19.9(3.3)	1.000(0.000)	0.074(0.007)	0.061(0.027)	0.807(0.071)	0.826(0.036)
GDE	45.2(11.0)	0.000(0.000)	0.000(0.000)	0.029(0.010)	0.836(0.076)	0.956(0.040)

$\mathcal{C}$(row, column)

FFGA	0.016(0.049)	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.963(0.128)	NPGA	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	0.860(0.168)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	SPEA	1.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.000(0.000)	NSGA	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.964(0.048)	1.000(0.000)	GDE

$\mathcal{V}$(row, column)

FFGA	0.002(0.007)	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.228(0.134)	NPGA	0.005(0.018)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.365(0.189)	0.389(0.159)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.427(0.118)	0.552(0.048)	0.563(0.026)	SPEA	0.287(0.025)	0.001(0.001)
0.393(0.123)	0.512(0.057)	0.497(0.045)	0.000(0.000)	NSGA	0.000(0.000)
0.435(0.114)	0.562(0.050)	0.574(0.025)	0.086(0.022)	0.328(0.023)	GDE

Table 3: Means of the solution cardinality (ℵ), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), maximum spread (D), $\mathcal{C}$, and $\mathcal{V}$ of MOEAs for ZDT2. Standard deviations are in parenthesis.

	ℵ	ER	GD	S	Δ	D
FFGA	28.0(6.0)	0.989(0.024)	0.268(0.037)	0.044(0.019)	0.849(0.043)	1.405(0.193)
NPGA	28.1(4.5)	0.953(0.076)	0.194(0.029)	0.044(0.015)	0.840(0.044)	1.229(0.081)
VEGA	23.1(5.0)	0.981(0.032)	0.166(0.029)	0.058(0.025)	0.838(0.052)	0.990(0.100)
SPEA	69.7(9.7)	0.034(0.017)	0.002(0.001)	0.013(0.011)	0.784(0.045)	0.931(0.019)
NSGA	46.1(6.8)	0.530(0.064)	0.029(0.006)	0.022(0.012)	0.813(0.041)	0.977(0.038)
GDE	78.1(4.6)	0.000(0.000)	0.000(0.000)	0.016(0.003)	1.085(0.064)	0.901(0.018)

$\mathcal{C}$(row, column)

FFGA	0.160(0.195)	0.018(0.030)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.813(0.153)	NPGA	0.025(0.035)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.933(0.079)	0.896(0.075)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	SPEA	0.992(0.023)	0.000(0.002)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.001(0.006)	NSGA	0.000(0.000)
0.856(0.063)	0.835(0.076)	0.824(0.092)	0.795(0.069)	0.928(0.053)	GDE

$\mathcal{V}$(row, column)

FFGA	0.015(0.026)	0.000(0.001)	0.000(0.000)	0.000(0.000)	0.001(0.001)
0.072(0.036)	NPGA	0.001(0.001)	0.000(0.000)	0.000(0.000)	0.001(0.002)
0.223(0.060)	0.198(0.048)	VEGA	0.000(0.000)	0.000(0.000)	0.001(0.001)
0.399(0.051)	0.416(0.036)	0.324(0.031)	SPEA	0.144(0.017)	0.001(0.001)
0.347(0.051)	0.357(0.034)	0.245(0.034)	0.000(0.001)	NSGA	0.000(0.001)
0.402(0.052)	0.421(0.036)	0.334(0.031)	0.036(0.008)	0.166(0.014)	GDE

Table 4: Means of the solution cardinality (ℵ), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), maximum spread (D), $\mathcal{C}$, and $\mathcal{V}$ of MOEAs for ZDT3. Standard deviations are in parenthesis.

	ℵ	ER	GD	S	Δ	D
FFGA	8.7(3.4)	1.000(0.000)	19.699(3.403)	0.203(0.154)	0.912(0.062)	30.917(10.658)
NPGA	4.7(1.8)	1.000(0.000)	9.266(2.834)	0.232(0.168)	0.926(0.062)	6.755(3.890)
VEGA	2.2(1.1)	1.000(0.000)	8.301(2.574)	0.346(0.342)	0.964(0.049)	1.053(1.203)
SPEA	84.8(50.5)	1.000(0.000)	0.570(0.322)	0.029(0.049)	0.950(0.041)	1.861(0.388)
NSGA	7.6(2.7)	1.000(0.000)	1.983(0.784)	0.216(0.102)	0.924(0.038)	1.492(0.558)
GDE	39.3(21.2)	0.316(0.416)	0.016(0.025)	0.054(0.036)	0.865(0.113)	0.902(0.105)

$\mathcal{C}$(row, column)

FFGA	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	NPGA	0.067(0.221)	0.000(0.000)	0.000(0.000)	0.000(0.000)
1.000(0.000)	0.885(0.250)	VEGA	0.054(0.212)	0.000(0.000)	0.000(0.000)
1.000(0.000)	1.000(0.000)	0.933(0.254)	SPEA	0.574(0.484)	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.404(0.477)	NSGA	0.000(0.000)
1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	GDE

$\mathcal{V}$(row, column)

FFGA	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.386(0.117)	NPGA	0.024(0.109)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.498(0.122)	0.519(0.314)	VEGA	0.022(0.095)	0.000(0.000)	0.000(0.000)
0.489(0.119)	0.621(0.210)	0.747(0.227)	SPEA	0.280(0.261)	0.000(0.000)
0.511(0.111)	0.654(0.200)	0.774(0.164)	0.186(0.255)	NSGA	0.000(0.000)
0.532(0.109)	0.724(0.174)	0.900(0.095)	0.684(0.069)	0.737(0.071)	GDE

Table 5: Means of the solution cardinality (ℵ), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), maximum spread (D), $\mathcal{C}$, and $\mathcal{V}$ of MOEAs for ZDT4. Standard deviations are in parenthesis.

	ℵ	ER	GD	S	Δ	D
FFGA	11.8(3.2)	1.000(0.000)	1.541(0.245)	0.098(0.055)	0.939(0.030)	2.010(0.703)
NPGA	5.9(2.3)	1.000(0.000)	1.473(0.352)	0.191(0.132)	0.945(0.030)	0.881(0.292)
VEGA	3.0(1.3)	1.000(0.000)	2.016(0.477)	0.207(0.196)	0.948(0.052)	0.987(0.906)
SPEA	12.0(5.1)	0.944(0.216)	0.151(0.057)	0.136(0.095)	0.956(0.168)	0.847(0.087)
NSGA	6.7(2.3)	1.000(0.000)	0.724(0.159)	0.154(0.095)	0.951(0.039)	0.747(0.076)
GDE	99.8(1.0)	0.067(0.254)	0.013(0.050)	0.013(0.003)	0.906(0.064)	0.979(0.064)

$\mathcal{C}$(row, column)

FFGA	0.000(0.000)	0.027(0.101)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.993(0.027)	NPGA	0.198(0.256)	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.930(0.117)	0.747(0.312)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.982(0.081)	0.982(0.072)	0.933(0.254)	SPEA	0.988(0.047)	0.067(0.254)
1.000(0.000)	1.000(0.000)	1.000(0.000)	0.000(0.000)	NSGA	0.042(0.168)
0.762(0.161)	0.506(0.244)	0.472(0.347)	0.732(0.302)	0.726(0.191)	GDE

$\mathcal{V}$(row, column)

FFGA	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.000(0.001)
0.445(0.180)	NPGA	0.029(0.148)	0.000(0.000)	0.000(0.000)	0.000(0.001)
0.523(0.168)	0.340(0.246)	VEGA	0.000(0.000)	0.000(0.000)	0.000(0.000)
0.660(0.116)	0.735(0.074)	0.694(0.180)	SPEA	0.700(0.038)	0.046(0.177)
0.598(0.146)	0.651(0.126)	0.602(0.253)	0.000(0.000)	NSGA	0.007(0.031)
0.671(0.115)	0.753(0.065)	0.710(0.171)	0.444(0.152)	0.699(0.183)	GDE

Table 6: Means of the solution cardinality (ℵ), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), maximum spread (D), $\mathcal{C}$, and $\mathcal{V}$ of MOEAs for ZDT6. Standard deviations are in parenthesis.

also noticed that suitable values for the crossover rate and the mutation factor should be drawn from a rather narrow range for some problems, *e.g.*, problem ZDT4, while the underlying reason for this remains open.

Usually large values (such as 0.9) are suggested as initial settings for the crossover rate CR and the mutation factor F in the case of single-objective problems. In the case of multiple objectives, it was observed that using a large crossover rate often leads to faster convergence along one objective compared to another. This causes the solution to converge to one point of the Pareto-optimal front, which is not desired. When $CR \approx 0.05 \ldots 0.5$, the obtained final population typically remains diverse as desired. Thus, searching the decision variable space along the directions of the coordinate axis performs better than a rotationally invariant search. Use of small values for CR was also reported in [2, 3, 5]. However, the current recommendations are based on limited experimentation and the problem of selecting the control parameter values is remaining mostly open.

Increasing the size of the population seemed to provide better approximation of the Pareto-optimal front without needing to compute more generations. However, the value NP = 100 was used here for comparison with the results of other MOEAs.

6 CONCLUSIONS AND FUTURE RESEARCH

In this paper the Differential Evolution algorithm and its extension for constrained multi-objective optimization are described. The method described extends the basic DE algorithm for constrained multi-objective optimization with minor changes to the original algorithm of DE and for this reason the method is named Generalized DE (GDE). It is effective and does not introduce any extra control parameters.

GDE is tested with five benchmark multi-objective test problems. The numerical results show that the method is able to provide a solution for all the test problems and performs well compared to other MOEAs in comparison, providing a relatively good approximation of the Pareto-optimal front. However, despite the distribution of the solution points along the approximated Pareto-optimal front was well comparable to the other MOEAs, it is still far from ideal. When the method is used for multi-objective optimization problems, our preliminary recommendations for the control parameter values are $CR \in [0.05, 0.5]$ and $F \in [0.05, 1+)$ for initial settings. The recommendation for CR differ clearly from those given in literature for solving single-objective problems, *e.g.*, $CR = 0.9$ [12, p. 129].

More intensive research of the effect of parameters on the optimization process, extensive comparison of GDE with latest multi-objective evolutionary algorithms and test problems, and applying GDE for practical multi-objective problems remains to be studied. Also, distribution of solutions and extent of the obtained non-dominated front could be improved because now GDE does not contain any mechanism for maintaining these.

ACKNOWLEDGEMENTS

S. Kukkonen gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE) and the Academy of Finland. The authors would like to thank K. E. Parsopoulos for useful discussions about performance metrics and E. Zitzler for providing results of other MOEAs.

REFERENCES

[1] C. S. Chang, D. Y. Xu, and H. B. Quek, "Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system," *IEE Proceedings on Electric Power Applications*, vol. 146, no. 5, pp. 577–583, Sept 1999.

[2] Hussein A. Abbass, Ruhul Sarker, and Charles Newton, "PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proceedings of the 2001 Congress on Evolutionary Computation, CEC'01*, Seoul, South Korea, 2001, pp. 971–978.

[3] Hussein A. Abbass and Ruhul Sarker, "The Pareto differential evolution algorithm," *International Journal on Artificial Intelligence Tools*, vol. 11, no. 4, pp. 531–552, 2002.

[4] Hussein A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, Honolulu, Hawaii, May 2002, pp. 831–836.

[5] Nateri K. Madavan, "Multiobjective optimization using a Pareto differential evolution approach," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, Honolulu, Hawaii, May 2002, pp. 1145–1150.

[6] B. V. Babu and M. Mathew Leenus Jehan, "Differential evolution for multi-objective optimization," in *Proceedings of the 2003 Congress on Evolutionary Computation, CEC'03*, Canberra, Australia, Dec 2003, pp. 2696–2703.

[7] Robert. J. Graves, Arthur. C. Sanderson, Feng. Xue, S. Bonissone, and R. Subbu, "Advanced research in scalable enterprise systems: Multi-objective optimization," Technical report, GE Global Research, April 2003, [Online] Available: www.crd.ge.com/cooltechnologies/pdf/2003grc127.pdf, 3.5.2004.

[8] Robert J. Graves Feng Xue, Arthur C. Sanderson, "Multi-objective differential evolution and its application to enterprise planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, Taiwan, May 2003, pp. 3535–3541.

[9] Vilfredo Pareto, *Cours D'Economie Politique*, Libraire Droz, Geneve, 1964 (the first edition in 1896).

[10] Kaisa Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, 1998.

[11] Rainer Storn and Kenneth V. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[12] Jouni Lampinen and Rainer Storn, *New Optimization Techniques in Engineering*, chapter Differential Evolution, pp. 123–166, Springer, 2004.

[13] David Corne, Margo Dorigo, and Fred Glover, *New Ideas in Optimization*, McGraw-Hill, London, 1999.

[14] Rainer Storn and Kenneth V. Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical report, ICSI, March 1995, [Online] Available: ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.gz, 3.5.2004.

[15] Feng-Sheng Wang and Ji-Pung Chiou, "Differential evolution for dynamic optimization of differential-algebraic systems," in *Proceedings of the IEEE International Conference on Evolutionary Computation, ICEC'97*, Indianapolis, IN, 1997, pp. 531–536.

[16] Tien-Ting Chang and Hong-Chan Chang, "Application of differential evolution to passive shunt harmonic filter planning," in *8th International Conference On Harmonics and Quality of Power*, Athens, Greece, Oct 1998, pp. 149–153.

[17] Rainer Storn, "System design by constraint adaptation and differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22–34, April 1999.

[18] Feng-Sheng Wang and Jyh-Woei Sheu, "Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast," *Chemical Engineering Science*, vol. 55, no. 18, pp. 3685–3695, Sept 2000.

[19] Jouni Lampinen, "A constraint handling approach for the differential evolution algorithm," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, Honolulu, Hawaii, May 2002, pp. 1468–1473.

[20] Jouni Lampinen, *Intelligent Technologies - Theory and Applications*, chapter A Constraint Handling Approach for the Differential Evolution Algorithm, pp. 152–158, IOS Press, 2002.

[21] Jouni Lampinen, "DE's selection rule for multiobjective optimization," Tech. Rep., Lappeenranta University of Technology, Department of Information Technology, 2001, [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 3.5.2004.

[22] Jouni Lampinen, "Multi-constrained nonlinear optimization by the differential evolution algorithm," Tech. Rep., Lappeenranta University of Technology, Department of Information Technology, 2001, [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/DECONSTR.PDF, 3.5.2004.

[23] Kalyanmoy Deb, *Multi-Objective Optimization using Evolutionary algorithms*, John Wiley & Sons, Chichester, England, 2001.

[24] Kalyanmoy Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[25] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000, Also available: ftp.tik.ee.ethz.ch/pub/people/zitzler/ZDT2000.ps, 15.1.2004.

[26] Eckart Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, Ph.D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany, Dec 1999.

[27] Saku Kukkonen and Jouni Lampinen, "A differential evolution algorithm for constrained multi-objective optimization: Initial assessment," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria, Feb 2004, pp. 96–102.

[28] Saku Kukkonen and Jouni Lampinen, "Mechanical component design for multiple objectives using generalized differential evolution," in *Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM2004)*, Bristol, United Kingdom, April 2004, pp. 261–272.

[29] Eckart Zitzler and Lothar Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 257–271, 1999.

[30] Eckart Zitzler and Marco Laumanns, "Test problem suite: Test problems and test data for multiobjective optimizers," 2003, [Online] Available: http://www.tik.ee.ethz.ch/∼zitzler/testdata.html, 3.5.2004.

[31] Jonathan E. Fieldsend, Richard M. Everson, and Sameer Singh, "Using unconstrained elite archives for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 305–323, June 2003.

[32] Marco Laumanns, Eckart Zitzler, and Lothar Thiele, "A unified model for multiobjective evolutionary algorithms with elitism," in *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, Piscataway, NJ, 2000, vol. 1, pp. 46–53.

Publication II

Kukkonen, S., Lampinen, J.,
An Empirical Study of Control Parameters for Generalized Differential Evolution

Reprinted, with permission, from
The Sixth Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005), Munich, Germany, September, 2005, 12 pages.

Evolutionary and Deterministic Methods for Design,
Optimization and Control with Applications to Industrial and Societal Problems
EUROGEN 2005
R. Schilling, W.Haase, J. Periaux, H. Baier, G. Bugeda (Eds)

AN EMPIRICAL STUDY OF CONTROL PARAMETERS FOR GENERALIZED DIFFERENTIAL EVOLUTION

Saku Kukkonen and Jouni Lampinen

Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20
FIN-53851 Lappeenranta, Finland
e-mail: saku.kukkonen@lut.fi

Key words: multi-objective optimization, Pareto-optimization, constraint handling, evolutionary algorithms, differential evolution, control parameters

Abstract. *In this paper the influence of control parameters to the search process of previously introduced Generalized Differential Evolution is empirically studied. Besides number of generations and size of the population, Generalized Differential Evolution has two control parameters, which have to be set before applying the method for solving a problem. The effect of these two control parameters is studied with a set of common bi-objective test problems and performance metrics.*

Objectives of this study are to investigate sensitivity of the control parameters according to different performance metrics, understand better the effect and tuning of the control parameters on the search process, and to find rules for the initial settings of the control parameters in the case of multi-objective optimization.

It seems that in the case of multi-objective optimization, there exists same kind of relationship between the control parameters and the progress of population variance as in the case of single-objective optimization with Differential Evolution. Based on the results, recommendations choosing control parameter values are given.

1 INTRODUCTION

Many practical problems have multiple objectives and several aspects cause constraints to problems. Generalized Differential Evolution (GDE) [1–4] is an extension of Differential Evolution (DE) [5, 6] for constrained multi-objective (Pareto-)optimization. DE is a relatively new real-coded Evolutionary Algorithm (EA), which has been demonstrated to be efficient for solving many real-world problems.

In the earlier studies [4,7,8] GDE is compared with other multi-objective EAs (MOEAs). In these studies it has been noted that GDE performs better with different control parameter values from those usually used for the basic DE algorithm in single-objective optimization. It has been also noted that the control parameter values should be drawn from a rather narrow range for some problems. However, more complete tests with different control parameter values has not been performed and reported before.

2 DIFFERENTIAL EVOLUTION

The DE algorithm [5,6] was introduced by Storn and Price in 1995. Design principles of DE are simplicity, efficiency, and use of floating-point encoding instead of binary numbers. As a typical EA, DE has a random initial population that is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides an appropriate stopping condition. Other control parameters for DE are crossover control parameter CR, mutation factor F, and the population size NP.

In each generation G, DE goes through each D dimensional decision vector $\vec{x_{i,G}}$ of the population and creates corresponding trial vector $\vec{u_{i,G}}$ as follows [9]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}\,, \text{(randomly selected,}\\
&\quad \text{except mutually different and different from } i)\\
&j_{rand} = \text{floor}\,(rand_i[0,1) \cdot D) + 1\\
&\text{for}(j = 1; j \leq D; j = j + 1)\\
&\{\\
&\quad \text{if}(rand_j[0,1) < CR \vee j = j_{rand})\\
&\qquad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G})\\
&\quad \text{else}\\
&\qquad u_{j,i,G} = x_{j,i,G}\\
&\}
\end{aligned}
\tag{1}
$$

Both CR and F remain fixed during the whole execution of the algorithm. Parameter CR, controlling the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors instead of from old vector $\vec{x_{i,G}}$. The condition "$j = j_{rand}$" is to make sure that at least one element is different compared to elements of the old vector. Parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$. In practice, CR controls the rotational

invariance of the search, and its small value (*e.g.* 0.1) is practicable with separable problems while larger values (*e.g.* 0.9) are for non-separable problems. Control parameter F controls the speed and robustness of the search, *i.e.*, a lower value for F increases the convergence rate but also the risk of stacking into a local optimum. Parameters CR and NP have the same kind of effect on the convergence rate as F has.

After the mutation and crossover operations trial vector $u_{i,G}^{\vec{}}$ is compared to old vector $x_{i,G}^{\vec{}}$. If the trial vector has equal or better objective value, then it replaces the old vector in the next generation. DE is an elitist method since the best population member is always preserved and the average objective value of the population will never worsen.

3 GENERALIZED DIFFERENTIAL EVOLUTION

Several extensions of DE for multi-objective optimization and for constrained optimization has been proposed [10–18]. Generalized Differential Evolution (GDE) [1–4, 7, 8] extends the basic DE algorithm for constrained multi-objective optimization by modifying the selection rule of the basic DE algorithm. Compared to the other DE extensions for multi-objective optimization, GDE makes DE suitable for constrained multi-objective optimization with noticeable fewer changes to the original DE algorithm. The modified selection operation for solving a constrained multi-objective optimization problem

$$\begin{array}{ll} \text{minimize} & \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_M(\vec{x})\} \\ \text{subject to} & (g_1(\vec{x}), g_2(\vec{x}), \ldots, g_K(\vec{x}))^T \leq \vec{0} \end{array} \tag{2}$$

with M objectives f_m and K constraint functions g_k is presented formally as follows [1]:

$$x_{i,\vec{G+1}} = \begin{cases} u_{i,G}^{\vec{}} & \text{if} \begin{cases} \begin{cases} \exists k \in \{1,\ldots,K\} : g_k(u_{i,G}^{\vec{}}) > 0 \\ \wedge \\ \forall k \in \{1,\ldots,K\} : g_k'(u_{i,G}^{\vec{}}) \leq g_k'(x_{i,G}^{\vec{}}) \end{cases} \\ \vee \\ \begin{cases} \forall k \in \{1,\ldots,K\} : g_k(u_{i,G}^{\vec{}}) \leq 0 \\ \wedge \\ \exists k \in \{1,\ldots,K\} : g_k(x_{i,G}^{\vec{}}) > 0 \end{cases} \\ \vee \\ \begin{cases} \forall k \in \{1,\ldots,K\} : g_k(u_{i,G}^{\vec{}}) \leq 0 \wedge g_k(x_{i,G}^{\vec{}}) \leq 0 \\ \wedge \\ \forall m \in \{1,\ldots,M\} : f_m(u_{i,G}^{\vec{}}) \leq f_m(x_{i,G}^{\vec{}}) \end{cases} \end{cases} \\ x_{i,G}^{\vec{}} & \text{otherwise} \end{cases}, \tag{3}$$

where $g_k'(x_{i,G}^{\vec{}}) = \max(g_k(x_{i,G}^{\vec{}}), 0)$ and $g_k'(u_{i,G}^{\vec{}}) = \max(g_k(u_{i,G}^{\vec{}}), 0)$ are representing the constraint violations.

The selection rule given in Eq. 3 selects trial vector $u_{i,G}^{\vec{}}$ to replace old vector $x_{i,G}^{\vec{}}$ in the next generation in the following cases:

1. Both trial vector $u_{i,G}^{\vec{}}$ and old vector $x_{i,G}^{\vec{}}$ violate at least one constraint but the trial vector does not violate any constraint more than the old vector.

2. Old vector $\vec{x_{i,G}}$ violates at least one constraint whereas trial vector $\vec{u_{i,G}}$ is feasible.

3. Both vectors are feasible and trial vector $\vec{u_{i,G}}$ has at least as good value for each objective than old vector $\vec{x_{i,G}}$ has.

Otherwise old vector $\vec{x_{i,G}}$ is preserved.

The basic idea in the selection rule given in Eq. 3 is that trial vector $\vec{u_{i,G}}$ is required to dominate compared old population member $\vec{x_{i,G}}$ in a constraint violation space or in an objective function space, or at least provide an equally good solution as $\vec{x_{i,G}}$.

After the selected number of generations the final population represents the solution for the optimization problem. Unique non-dominated vectors can be separated from the final population if desired. There is no sorting of non-dominated vectors during the optimization process or any mechanism for maintaining diversity of the solution.

In the earlier studies [4,7,8] GDE was compared with other MOEAs. In these studies, it was observed that relatively small values for crossover control parameter CR and mutation factor F should be used in the case of used multi-objective test problems. It was also noticed that suitable values for these control parameters should be drawn from a rather narrow range for some problems to obtain a converged solution along the Pareto front. However, performance was not thoroughly tested with commonly adopted metrics for MOEAs by applying different control parameter combinations.

4 EXPERIMENTS

In this investigation GDE was tested with a set of bi-objective benchmark problems commonly used in the MOEA literature. These problems are known as SCH1, SCH2, FON, KUR, POL, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [19, pp. 338–360] and they all have two objectives to be minimized.

With all the test problems the size of the population and the number of generations were kept fixed. From the final population unique non-dominated solutions were extracted to form the final solution, an approximation of the Pareto front. Characteristics of the problems, the size of the population, and the number of generations are collected to Table 1. Relatively small number of generations was used for some problems to observe the effect of the control parameters.

Different control parameter value combinations in the range $CR \in [0, 1]$ and $F \in [0, 3]$ with a resolution of 0.05 were tested for all the test problems. Tests were repeated 100 times with each control parameter combination and the results were evaluated using convergence and diversity metrics. Closeness to the Pareto front was measured with a generational distance (GD) [19, pp. 326–327], which measures the average distance of the solution vectors from the Pareto front. Diversity of the obtained solution was measured using a spacing (S) metric [19, pp. 327–328], which measures the standard deviation of the distances from each vector to the nearest vector in the obtained non-dominated set. Smaller values for both metrics are preferable, and the optimal values are zero. The number of unique non-dominated vectors ($\aleph$) in the final solution was also registered.

Problem	D	Separability	Pf, decision space	Pf, objective space	NP	G_{max}
SCH1	1	N/A	1 line	1 curve, convex	100	30
SCH2	1	N/A	2 lines	2 curves	100	5
FON	10	separable	1 line	1 curve, non-convex	100	250
KUR	3	separable	2 points & 4 curves	1 point & 3 curves	100	150
POL	2	non-separable	2 curves	2 curves	100	30
ZDT1	30	separable	1 line	1 curve, convex	100	250
ZDT2	30	separable	1 line	1 curve, concave	100	250
ZDT3	30	separable	5 lines	5 curves	100	250
ZDT4	10	separable	1 line	1 curve, convex	100	250
ZDT6	10	separable	1 line	1 curve, concave	100	250

Table 1: Characteristics of the bi-objective optimization test problems (Pf = Pareto front) [19, pp. 338–360] [20, pp. 106–108, 124–125], the size of the population, and the number of generations.

Results for the different problems are shown in Figs. 1–10 as surfaces in the control parameter space. The results for all the problems show that with a small F value the final solution has greatest amount of unique non-dominated vectors and also values for performance metrics GD and S are best. For the multidimensional (*i.e.* $D > 1$) test problems, values for the performance metrics are best with low CR values, whereas for the one-dimensional test problems, CR does not seem to have notable impact on the solution.

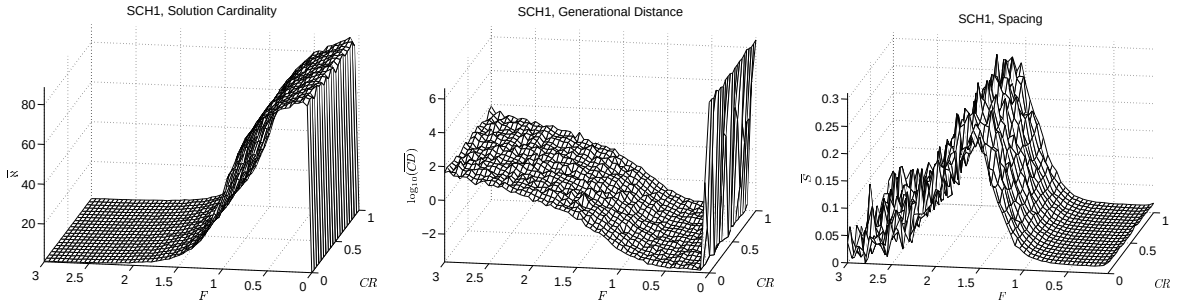


Figure 1: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD (note logarithmic scale), and spacing S for SCH1 shown as surfaces in the control parameter space. Strange shape of the spacing surface is due to only a few non-dominated vectors when F is large.

The value range for F in the tests is larger than usually used for single-objective optimization. Based on the results, a larger F value does not give any extra benefit in the case of multi-objective optimization and therefore its value can be chosen from the same value range as in the case of single-objective optimization.

From the results of the multidimensional test problems, one can observe a non-linear

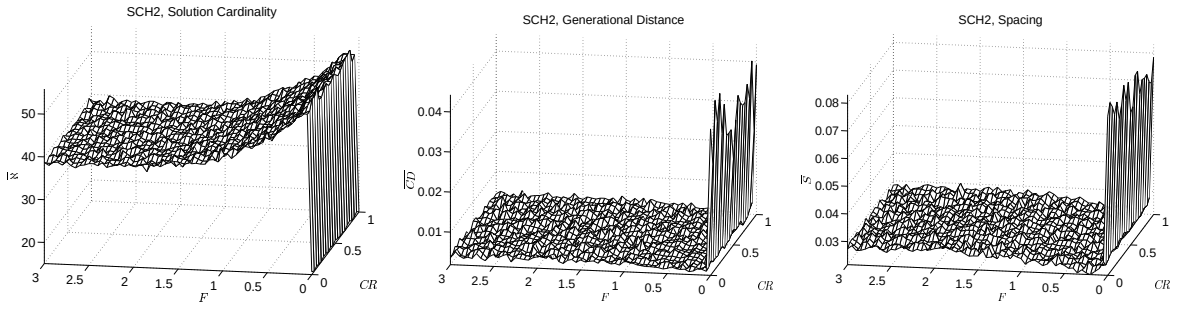


Figure 2: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for SCH2 shown as surfaces in the control parameter space.

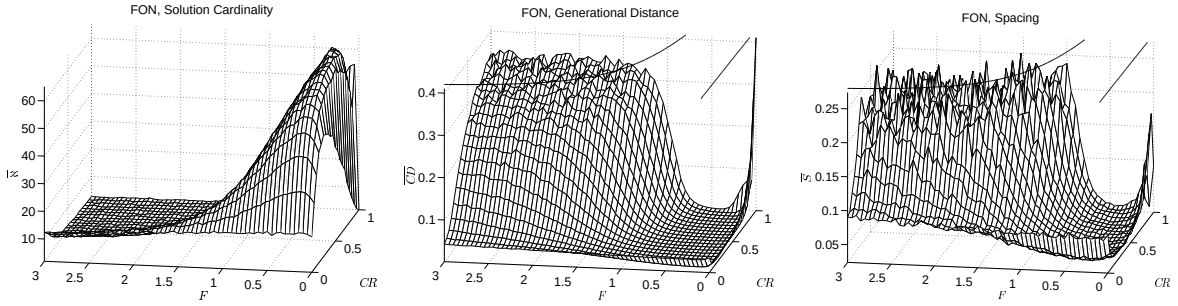


Figure 3: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for FON shown as surfaces in the control parameter space.

relationship between CR and F values, *i.e.*, a larger F value can be used with a small CR value than with a large CR value, and this relationship is non-linear. An explanation for this can be found from a theory for the single-objective DE algorithm. A formula for the relationship between the control parameters of DE and the evolution of the population variance has been conducted in [21]. The change of the population variance between successive generations due to the crossover and mutation operations is denoted with c and its value is calculated as $c = \sqrt{2F^2CR - 2CR/NP + CR^2/NP + 1}$. When $c < 1$, the crossover and mutation operations decrease the population variance. When $c = 1$, the variance do not change, and when $c > 1$, the variance increases. Since a selection operation of an EA usually decreases the population variance, $c > 1$ is recommended to prevent too high convergence rate with a poor search coverage, which typically results in a premature convergence. On the other hand, if c is too large, the search process proceeds reliably, but too slowly.

When the size of the population is relatively large (*e.g.* $NP > 50$), the value of c depends mainly on the values of CR and F. Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$ in the control parameter space are shown in Fig. 11. These curves are also shown with

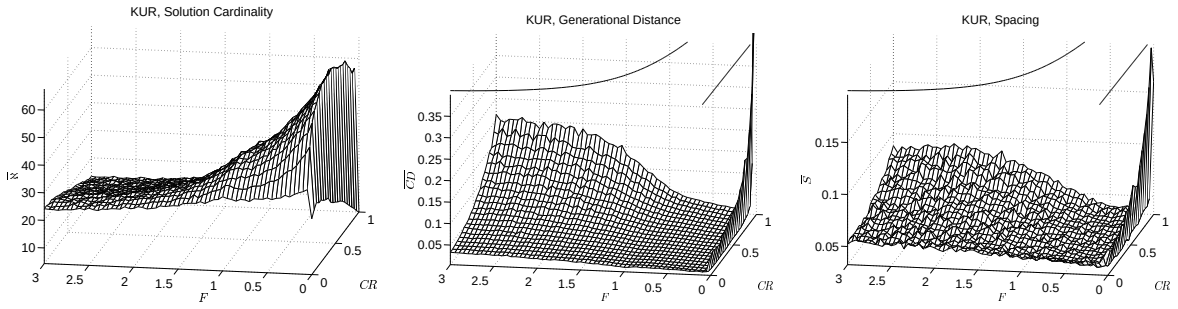


Figure 4: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for KUR shown as surfaces in the control parameter space.

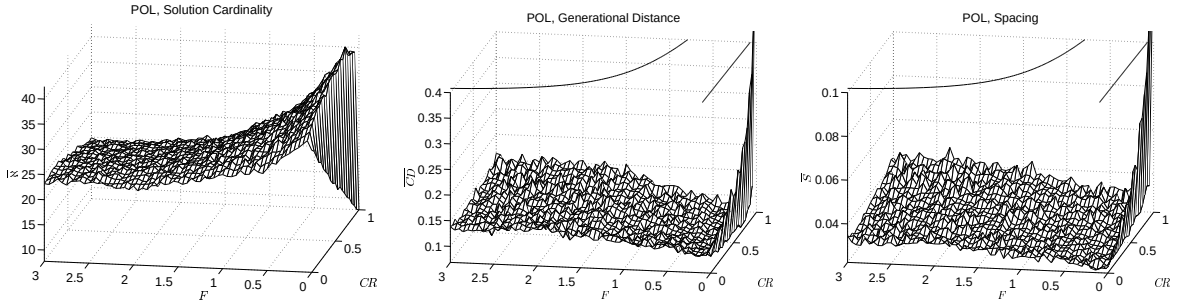


Figure 5: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for POL shown as surfaces in the control parameter space.

the GD and S surfaces in Figs. 3–10. As it can be observed, the surface for c in Fig. 11 corresponds well with the surfaces for GD and S in most of the test problems proposing that the theory for the population variance between successive generations is applicable also in the case of multi-objective optimization with GDE. This is natural, since single-objective optimization can be seen as a special case of multi-objective optimization, and a large c means slow but reliable search while a small c means opposite.

In general, the results are good with small CR and F values meaning also a low c value close to one. One reason for the good performance with small control parameter values is that the solution of a multi-objective problem is usually a set of non-dominated vectors instead of one vector, and conflicting objectives of the problem reduce overall selection pressure, maintain the diversity of the population, and prevent the premature convergence. Another reason is that all the multidimensional test problems here are separable or almost separable. With strongly non-separable objective functions, a larger CR might work better. Also, most of the problems have relatively easy objective functions to solve leading faster convergence with a small F, while a bigger F might be needed for harder functions with several local optima.

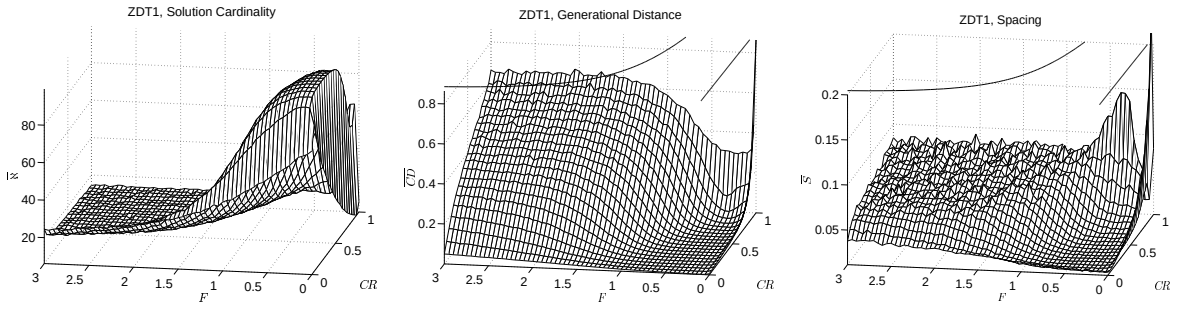


Figure 6: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for ZDT1 shown as surfaces in the control parameter space.

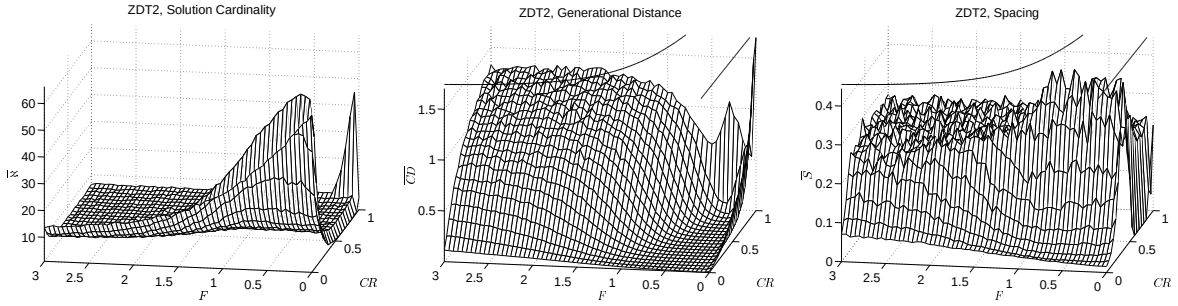


Figure 7: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for ZDT2 shown as surfaces in the control parameter space.

The results for most of the ZDT problems show that the value of control parameter CR must be kept low (*e.g.* $0.1 - 0.3$) to obtain good results. This is due to the fact that in the ZDT problems, the first objective depends on one decision variable while the second objective depends on rest of the decision variables. This leads easily to a situation when the first objective gets optimized faster than the second one and the solution converges to a single point on the Pareto front since in GDE there is no mechanism for maintaining diversity of the solution. Therefore, if the optimization difficulty for different objectives varies, then the value for CR must be close to zero to prevent the solution converge to a single point of the Pareto-optimal front.

Based on the results, our recommendations for the control parameter values are $CR \in [0.05, 0.8]$ and $F \in [0.1, 0.6]$ for initial settings, and it is safer to use smaller values for CR. However, these recommendations are based on the results of the used test functions. In overall, it is wise to select values for control parameters CR and F satisfying a condition $1.0 < c < 1.5$.

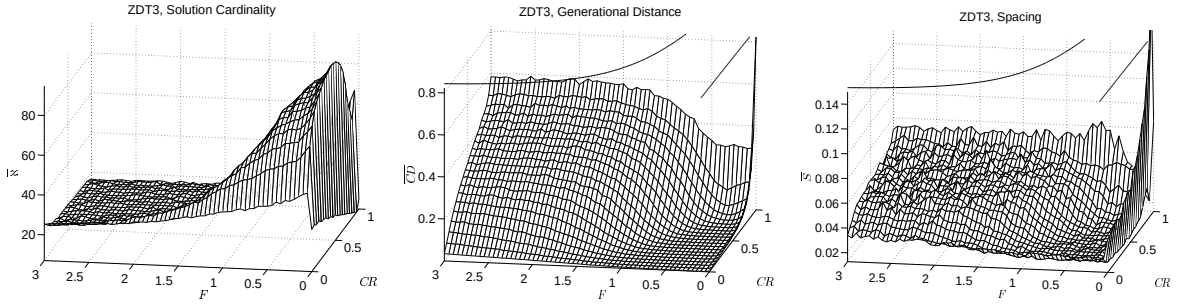


Figure 8: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for ZDT3 shown as surfaces in the control parameter space.

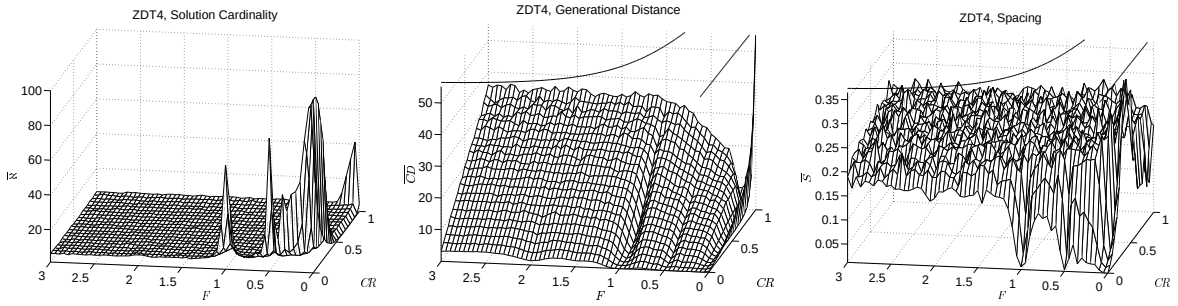


Figure 9: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for ZDT4 shown as surfaces in the control parameter space.

5 CONCLUSIONS

An extension of Differential Evolution (DE), Generalized Differential Evolution (GDE), is described. GDE has the same control parameters, crossover control parameter CR and mutation factor F, as the basic DE has. Suitable values for these control parameters has been reported earlier to be different compared to those usually used with the basic single-objective DE, but no extensive study was performed.

Different control parameter values for GDE were tested using common bi-objective test problems and metrics measuring the convergence and diversity of the solutions. Based on the empirical results, suitable initial control parameter values are $CR \in [0.05, 0.8]$ and $F \in [0.1, 0.6]$. The results also propose that a larger F value than usually used in the case of single-objective optimization does not give any extra benefit in the case of multi-objective optimization. It also seems that in some cases it is better to use a smaller CR value to prevent the solution converge to a single point of the Pareto front. However, these results are based on mostly separable problems with conflicting objectives.

Also in the case of multi-objective optimization, the non-linear relationship between CR and F was observed according to the theory of the basic single-objective DE about the

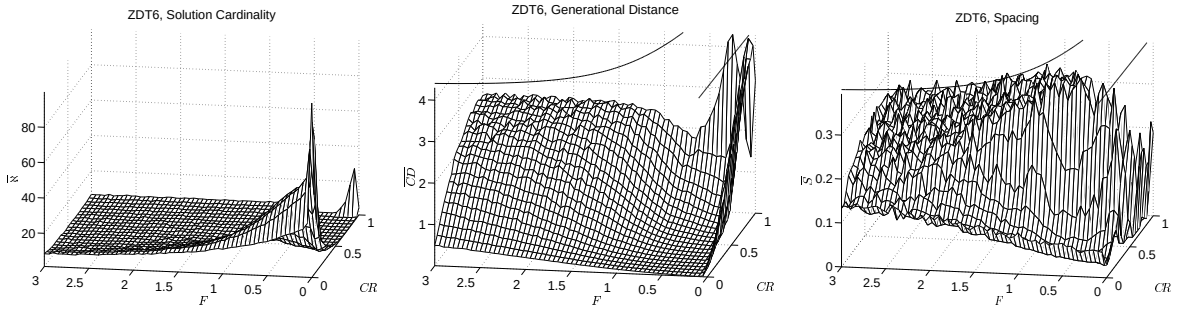


Figure 10: Mean values of the number of unique non-dominated vectors $\aleph$ in the final solution, generational distance GD, and spacing S for ZDT6 shown as surfaces in the control parameter space.

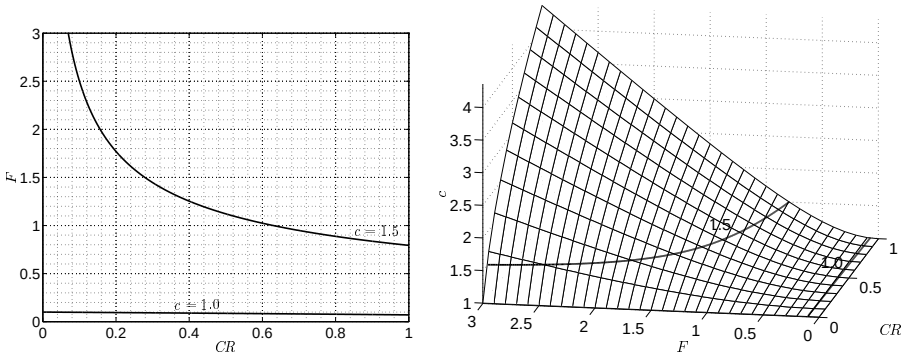


Figure 11: Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$ in the control parameter space.

relationship between the control parameters and the evolution of the population variance. It is advisable to select values for CR and F satisfying the condition $1.0 < c < 1.5$.

In the future, the correlation between c and performance metrics should be studied also numerically. Furthermore, the further investigation of the generalization of the theory for the evolution of the population variance to multi-objective problems is necessary. Also, extending studies for strongly non-separable multi-objective problems and problems with more than two objectives remains to be studied. Special cases, as multi-objective problems without conflicting objectives and single-objective problems transformed into multi-objective forms, might be interesting to investigate.

ACKNOWLEDGMENTS

S. Kukkonen gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), the Academy of Finland, Technological Foundation of Finland, and Finnish Cultural Foundation. He also wishes to thank Dr. K. Deb and all the KanGAL people for help during his visit in Spring 2005.

REFERENCES

[1] Jouni Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001. [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 1.6.2005.

[2] Jouni Lampinen. Multi-constrained nonlinear optimization by the Differential Evolution algorithm. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001. [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/DECONSTR.PDF, 1.6.2005.

[3] Jouni Lampinen. A constraint handling approach for the Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 1468–1473, Honolulu, Hawaii, May 2002.

[4] Saku Kukkonen and Jouni Lampinen. Comparison of Generalized Differential Evolution algorithm to other multi-objective evolutionary algorithms. In *Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, page 445, Jyväskylä, Finland, July 2004.

[5] Rainer Storn and Kenneth V. Price. Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[6] Kenneth V. Price, Rainer Storn, and Jouni Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer-Verlag, Berlin, 2005.

[7] Saku Kukkonen and Jouni Lampinen. A Differential Evolution algorithm for constrained multi-objective optimization: Initial assessment. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, pages 96–102, Innsbruck, Austria, Feb 2004.

[8] Saku Kukkonen and Jouni Lampinen. Mechanical component design for multiple objectives using Generalized Differential Evolution. In *Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM 2004)*, pages 261–272, Bristol, United Kingdom, April 2004.

[9] Kenneth V. Price. *New Ideas in Optimization*, chapter An Introduction to Differential Evolution, pages 79–108. McGraw-Hill, London, 1999.

[10] Rainer Storn. System design by constraint adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, April 1999.

[11] Feng-Sheng Wang and Jyh-Woei Sheu. Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. *Chemical Engineering Science*, 55(18):3685–3695, Sept 2000.

[12] Hussein A. Abbass and Ruhul Sarker. The Pareto Differential Evolution algorithm. *International Journal on Artificial Intelligence Tools*, 11(4):531–552, 2002.

[13] Hussein A. Abbass. The self-adaptive Pareto Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 831–836, Honolulu, Hawaii, May 2002.

[14] Nateri K. Madavan. Multiobjective optimization using a Pareto Differential Evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 1145–1150, Honolulu, Hawaii, May 2002.

[15] B. V. Babu and M. Mathew Leenus Jehan. Differential Evolution for multi-objective optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 2696–2703, Canberra, Australia, Dec 2003.

[16] Feng Xue, Arthur C. Sanderson, and Robert J. Graves. Multi-objective Differential Evolution and its application to enterprise planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3535–3541, Taiwan, May 2003.

[17] Efrén Mezura-Montes, Carlos A. Coello Coello, and Edy I. Tun-Morales. Simple feasibility rules and Differential Evolution for constrained optimization. In *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI 2004)*, pages 707–716, Mexico City, Mexico, April 2004.

[18] Tea Robič and Bogdan Filipič. DEMO: Differential Evolution for multiobjective optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 520–533, Guanajuato, Mexico, March 2005.

[19] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, England, 2001.

[20] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York, United States of America, 2002.

[21] Daniela Zaharie. Critical values for the control parameters of Differential Evolution algorithms. In *Proceedings of Mendel 2004, 8th International Conference on Soft Computing*, pages 62–67, Brno, Czech Republic, June 2002.

Publication III

Kukkonen, S., Lampinen, J.,
An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints

Reprinted, with kind permission of Springer Science and Business Media, from *The 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Lecture Notes in Computer Science (LNCS), Vol. 3242, Birmingham, England, September, 2004, pages 752–761.*

An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints

Saku Kukkonen and Jouni Lampinen

Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20
FIN-53851 Lappeenranta, Finland
Saku.Kukkonen@lut.fi

Abstract. In this paper an extension of Generalized Differential Evolution for constrained multi-objective (Pareto-)optimization is proposed. The proposed extension adds a mechanism for maintaining extent and distribution of the obtained non-dominated solutions approximating a Pareto front. The proposed extension is tested with a set of five benchmark multi-objective test problems and results are numerically compared to known global Pareto fronts and to results obtained with the elitist Non-Dominated Sorting Genetic Algorithm and Generalized Differential Evolution. Results show that the extension improves extent and distribution of solutions of Generalized Differential Evolution.

Keywords: multi-objective optimization, Pareto-optimization, constraint handling, evolutionary algorithms, differential evolution

1 Introduction

Many situations in engineering and economics deal with optimization. One may want to optimize, *e.g.*, manufacturing processes, shape of products, and number of different products to be manufactured. Typical goals are minimizing costs, maximizing profits, and improving performance. Several natural aspects limit feasible solutions, *e.g.*, resources may cause limitations and/or the number of products cannot be a negative number.

Optimization is an intensively studied problem field in mathematics. However, functions to be optimized in traditional mathematics are relatively simple (continuous, convex, unimodal, differentiable, *etc.*), yet functions to be optimized in practice are often far more complicated (discontinuous, non-convex, multi-modal, non-differentiable, *etc.*). In such cases various stochastic optimization methods have shown their effectiveness.

Most optimization research deals with single-objective optimization problems. The basic nature of many optimization problems is, however, multi-objective and these problems are usually first converted to single-objective problems.

X. Yao et al. (Eds.): PPSN VIII, LNCS 3242, pp. 752–761, 2004.

Single-objective problems are commonly considered easier to solve but conversion from a multi-objective problem to a single-objective problem requires some *a priori* knowledge which is not necessarily available or which is hard to determine, *e.g.*, the relative importance of each individual sub-objective. For this reason interest exists in solving multi-objective problems in multi-objective form.

Several extensions of Differential Evolution (DE) for multi-objective optimization have already been proposed [1–5]. Most of these methods use a non-dominated sorting for reproduction in each generation and some distance metric to prevent crowding.

This paper continues with the following parts: In Section 2 the concept of multi-objective optimization with constraints is handled briefly. Section 3 describes Differential Evolution algorithm and Section 4 describes proposed extension for constrained multi-objective optimization. Section 5 describes experiments and finally conclusions are given in Section 6.

2 Multi-objective Optimization with Constraints

Many practical problems have multiple objectives. For example, designing a wing of an aircraft may have objectives such as maximizing strength, minimizing weight, minimizing manufacturing costs, maximizing lifting force, minimizing drag, *etc.* Multiple objectives are almost always more or less conflicting.

Several aspects cause constraints to problems. In the previous example of the wing, the thickness of the metal parts used must be a positive number, shape limitations exist, some parts are available only in some predefined standard sizes, *etc.* Constraints can be divided into box or boundary constraints and constraint functions. Boundary constraints are used when the value of some optimized variable is limited to some range and constraint functions are representing more complicated constraints which are expressed as functions.

Multi-objective problems are often converted to single-objective problems by predefining weighting factors for different objectives, expressing the relative importance of each objective. However, this is impossible in many cases because a decision-maker does not necessarily know beforehand how different objectives should be weighted. Thus, a more convenient way is to keep multiple objectives of multi-objective problems and try to solve them in this form even though this may be harder to do in practice. Optimizing several objectives simultaneously without articulating the relative importance of each objective *a priori*, is often called Pareto-optimization [6]. An obtained solution is Pareto-optimal if none of the objectives can be improved without impairing at least one other objective [7, p. 11–12]. If the obtained solution can be improved in such way that at least one objective improves and other objectives do not decline, then the new solution dominates the original solution. A set of Pareto-optimal solutions form a Pareto front. An approximation of the Pareto front is called a set of non-dominated solutions because the solutions in this set are not dominating each other in the space of objective functions. From the set of non-dominated solutions the decision-maker may select one which has suitable values for different objectives.

This can be viewed as *a posteriori* articulation of the decision-makers preferences concerning the relative importance of each objective.

A mathematically constrained multi-objective optimization problem can be presented in the form [7, p. 37]

$$\begin{aligned} &\text{minimize} \quad \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})\} \\ &\text{subject to } \boldsymbol{x} \in S = \{\boldsymbol{x} \in \mathbf{R}^D | \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_M(\boldsymbol{x}))^T \leq \mathbf{0}\} \end{aligned} \tag{1}$$

Thus, there are K functions to be optimized and M constraint functions.

The major part of earlier mathematical research has concentrated on optimization problems where the functions are linear, differentiable, convex, or otherwise mathematically well behaving. However, in practical problems objective functions are often nonlinear, non-differentiable, discontinuous, multi-modal, *etc.* and no presumptions can be made about their behavior. Variables may also be integers or discrete instead of being continuous. Most traditional optimization methods cannot handle such complexity or do not perform well in these cases in which the assumptions they are based on do not hold. For such problems stochastic optimization methods such as Simulated Annealing (SA) and Evolution Algorithms (EAs) have been demonstrated to be effective because they do not rely on any assumptions concerning the objective and constraint functions.

3 Differential Evolution

The Differential Evolution (DE) algorithm [8, 9] [10, pp. 79–108] belongs to the family of Evolution Algorithms and was introduced by Storn and Price in 1995 [11]. Design principles in DE were simplicity, efficiency, and use of floating-point encoding instead of binary numbers.

Like in a typical EA, the idea in DE is to have some random initial population which is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides an appropriate stopping condition.

A trial vector $\boldsymbol{u}_{i,G}$ created by mutation and crossover operations is compared to an old objective vector $\boldsymbol{x}_{i,G}$. Here i is an index of the vector in the population and G is a generation index. If the trial vector has equal or lower objective value, then it replaces the old vector. This selection operation can be presented as follows [10, p. 82]:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} \text{ if } & f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} \text{ otherwise} & \end{cases} \tag{2}$$

The average objective value of the population will never increase, because the trial vector replaces the old vector only if it has equal or lower objective value.

4 An Extension of Generalized Differential Evolution

Generalized Differential Evolution (GDE) [12–17] extends the selection operation of the basic DE algorithm for constrained multi-objective optimization. GDE

has been demonstrated to have good convergence properties but distribution of solutions and extent of the obtained non-dominated front need to be improved. GDE does not contain any mechanism for maintaining these. As an attempt to improve GDE from this point of view, a modified selection operation for GDE is proposed in this paper. The proposed selection operation for M constraint and K objective functions is presented formally in (3).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} \text{ if } \begin{cases} \begin{cases} \exists j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) > 0 \\ \wedge \\ \forall j \in \{1,\ldots,M\} : g'_j(\boldsymbol{u}_{i,G}) \leq g'_j(\boldsymbol{x}_{i,G}) \end{cases} \\ \vee \\ \begin{cases} \forall j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) \leq 0 \\ \wedge \\ \exists j \in \{1,\ldots,M\} : g_j(\boldsymbol{x}_{i,G}) > 0 \end{cases} \\ \vee \\ \begin{cases} \forall j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) \leq 0 \wedge g_j(\boldsymbol{x}_{i,G}) \leq 0 \\ \wedge \\ \begin{cases} \forall k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) \leq f_k(\boldsymbol{x}_{i,G}) \\ \vee \\ \begin{cases} \neg \left[\forall k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) \geq f_k(\boldsymbol{x}_{i,G}) \wedge \right. \\ \quad \left. \exists k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) > f_k(\boldsymbol{x}_{i,G})\right] \\ \wedge \\ d_{\boldsymbol{u}_{i,G}} \geq d_{\boldsymbol{x}_{i,G}} \end{cases} \end{cases} \end{cases} \\ \boldsymbol{x}_{i,G} \text{ otherwise} \end{cases} \quad , \quad (3)$$

where $g'_j(\boldsymbol{x}_{i,G}) = \max(g_j(\boldsymbol{x}_{i,G}), 0)$ and $g'_j(\boldsymbol{u}_{i,G}) = \max(g_j(\boldsymbol{u}_{i,G}), 0)$ are representing the constraint violations, and d_i is a distance measure for measuring the distance from a particular solution i to its neighbor solutions.

The selection rule given in (3) selects the trial vector $\boldsymbol{u}_{i,G}$ to replace the old vector $\boldsymbol{x}_{i,G}$ in the following cases:

1. Both the trial vector and the old vector violate at least one constraint but the trial vector does not violate any of the constraints more than the old vector does.
2. The old vector violates at least one constraint whereas the trial vector is feasible.
3. Both vectors are feasible and
   - the trial vector dominates the old vector or has equal value for all objectives, or
   - the old vector does not dominate the trial vector and the old vector resides in a more crowded region of the objective space.

Otherwise the old vector $\boldsymbol{x}_{i,G}$ is preserved.

The basic idea in the selection rule is that the trial vector is required to dominate the compared old population member in constraint violation space or in objective function space, or at least provide an equally good solution as the old population member. If both vectors are feasible and they do not dominate

each other, then the one residing in a less crowded region of the objective space is chosen to the population of the next generation. The principle of constraint handling is effectively rather similar to the method described in [18] even though the formulation is different. The main difference is in the case of two infeasible solutions. In this case the selection rule given in (3) compares solutions based on dominance of the constraint violations whereas the selection method described in [18] compares solutions based on a sum of the constraint violations which needs evaluation of all constraint functions and normalization of their values.

The whole selection rule given in (3) is effectively almost same as a constrained tournament method in [19, pp. 301–308]. In the selection rule given in (3) the trial vector is preferred over the old vector also in the cases when the trial vector is equally good as the old vector, *i.e.*, constraint violations are equal or objective function values are equal.

The selection rule given in (3) can be implemented in such a way that the number of function evaluations is reduced because not always all the constraints and objectives need to be evaluated, *e.g.*, inspecting constraint violations (even one constraint) is often enough to determine which vector to select for the next generation [13, 14]. However, in the case of feasible solutions all the objectives need to be evaluated which was not always necessary in earlier GDE. This will increase the total number of function evaluations as well as execution time. Also calculation of the distance measure, d_i, will increase execution time. In principle, any measure of distance from a solution to its neighbor solutions can be applied. A crowding distance [19, pp. 248–249] was applied here as the distance measure, d_i, because it does not need any extra parameters.

After the selected number of generations the final population presents a solution for the optimization problem. The non-dominated solutions can be separated from the final population if desired. There is no sorting of non-dominated solutions during the optimization process.

Later on in this paper the proposed method with the selection rule given in (3) is called *Generalized Differential Evolution 2 (GDE2)*. The selection rule given in (3) handles any number M of constraints and any number K of objectives. When $M = 0$ and $K = 1$, the selection rule is identical to the selection rule of the basic DE algorithm.

Usually large values (such as 0.9) are suggested as initial settings for the crossover rate CR and mutation factor F in the case of single-objective problems. In the case of multiple objectives, it was observed that using a large crossover rate often leads to faster convergence along one objective compared to another [15–17]. This causes the solution to converge to a single point of the Pareto front, which is not desired. Based on this observation, our initial recommendations for the control parameter values used for multi-objective optimization problems are $CR \in [0.05, 0.5]$ and $F \in [0.05, 1+)$ for initial settings. In line with these observations, use of small values for CR was also reported by other researchers in [2, 3]. However, the current recommendations are based on limited experimentation, and the problem of selecting the control parameter values is remaining mostly open.

5 Experiments

GDE and GDE2 were implemented in C and tested with a set of five bi-objective benchmark problems described in [20] and [21, pp. 57–59]. These problems are known as ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [19, pp. 356–360]. They are designed to test the ability of a multi-objective optimization method to handle convexity (ZDT1), non-convexity (ZDT2), discontinuity (ZDT3), multi-modality (ZDT4), and non-uniformity (ZDT6) of the Pareto front.

5.1 Experimental Results and Discussions

In all the test problems the size of the population was 100, the number of generations was 250, and the control parameter values for GDE and GDE2 were $CR = 0.05$ and $F = 0.1$. In preliminary tests control parameter values 0.05, 0.1, 0.2, 0.3, and 0.4 were tested and suitable crossover rate and mutation factor were thereby approximately determined. It was noticed that suitable values for the crossover rate and mutation factor should be drawn from a rather narrow range for some problems, *e.g.*, problem ZDT4, while the underlying reason for this remains open.

The results of a single run of GDE and GDE2 for solving the multi-objective benchmark problems are shown in Fig. 1, where known global Pareto fronts and the results obtained with the Strength Pareto Evolutionary Algorithm (SPEA) [22] and the elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [23] are also shown for comparison and visual assessment.

Tests for NSGA-II, GDE, and GDE2 were repeated 100 times with different random number generator seeds and the results were compared with different metrics. NSGA-II was selected for comparison because of its good performance in previous comparison tests [23] and since it is well known within the multi-objective optimization community.

Closeness to the Pareto front was measured with an error ratio (ER) and a generational distance (GD) [19, pp. 324–327]. Diversity of the obtained solution was measured using spacing (S), spread (Δ), and maximum spread (D) metrics [19, pp. 328–331]. Smaller values for the error ratio, generational distance, spacing, and spread are preferable. The optimal value for the maximum spread is 1.

Average numbers of needed function evaluations for GDE and average execution times for the methods are reported in Table 1. For NSGA-II and GDE2 2×25100 function evaluations were needed on each run. All the tests were run on a Sun Sparc Ultra2. Table 2 contains the performance measurements solving the benchmark problems. Solutions contained the non-dominated members of the final population.

The results show that GDE2 improved extent and diversity of solutions over GDE without impairing the convergence property of GDE and increasing execution time only by little. NSGA-II was slightly better than GDE2 in most of the problems according to the metrics but NSGA-II needed more execution

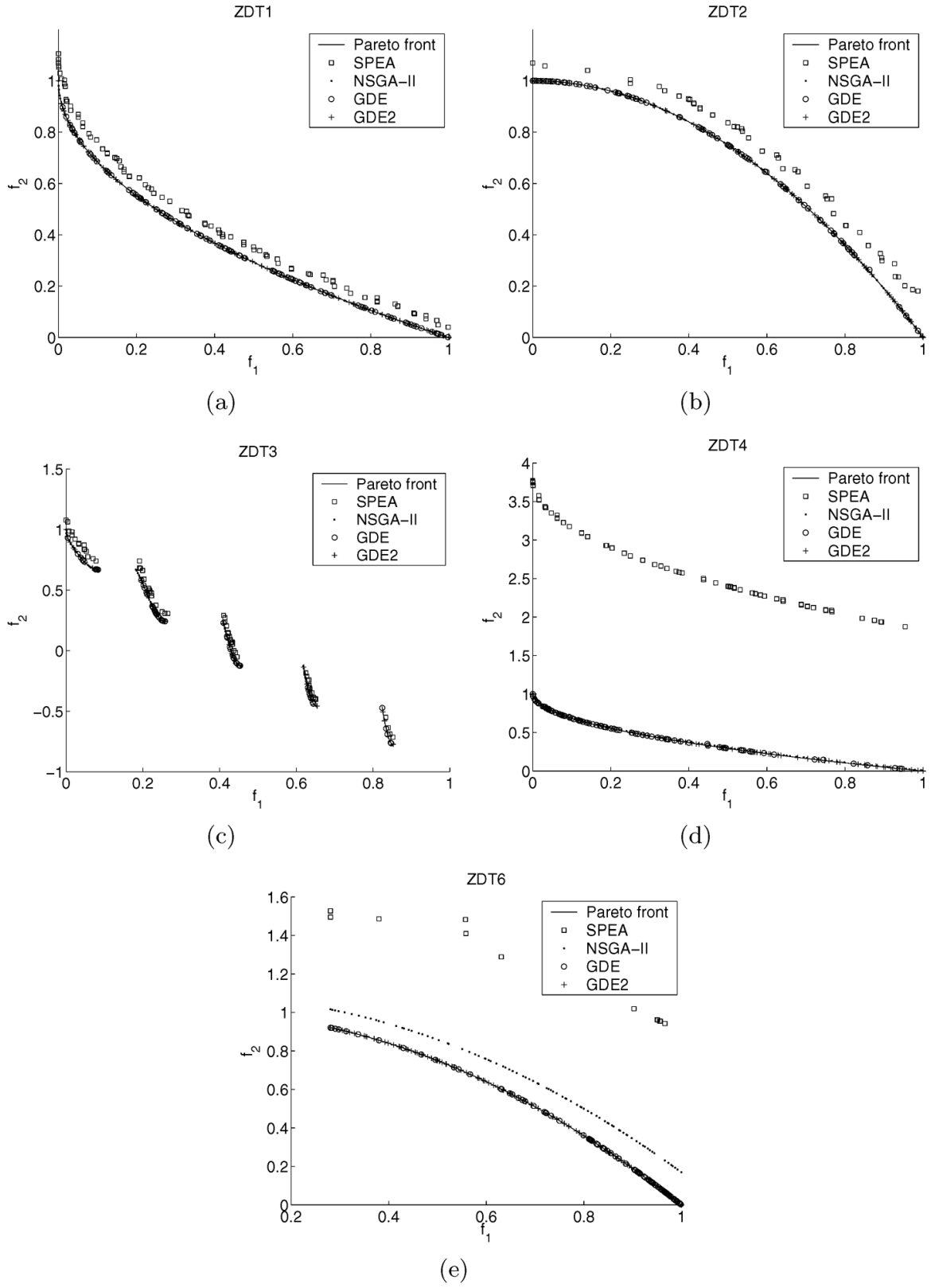


Fig. 1. Global Pareto front and solutions obtained with SPEA, NSGA-II, GDE, and GDE2 for a) ZDT1 b) ZDT2 c) ZDT3 d) ZDT4 e) ZDT6.

Table 1. Average execution times of NSGA-II, GDE, and GDE2 solving multi-objective benchmark test problems. Average number of needed function evaluations (f_1 and f_2) of GDE are also shown. Standard deviations are in parenthesis.

	NSGA-II	GDE			GDE2
	Execution time	Execution time	f_1	f_2	Execution time
ZDT1	4.0040(0.3661) s	1.0166(0.0071) s	25100(0.0)	24079.2(29.3)	1.0875(0.0091) s
ZDT2	4.0315(0.2389) s	1.0230(0.0163) s	25100(0.0)	24080.2(27.9)	1.0820(0.0067) s
ZDT3	3.9735(0.3963) s	1.0477(0.0085) s	25100(0.0)	24078.3(32.8)	1.1169(0.0081) s
ZDT4	3.9341(0.4769) s	0.5537(0.0085) s	25100(0.0)	23280.5(37.1)	0.6329(0.0151) s
ZDT6	4.0762(2.6311) s	0.5782(0.0097) s	25100(0.0)	22885.3(68.0)	0.6554(0.0106) s

Table 2. Means of the solution cardinality ($\aleph$), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), and maximum spread (D) of NSGA-II, GDE, and GDE2 for the multi-objective benchmark test problems. Standard deviations are in parenthesis.

ZDT1	$\aleph$	ER	GD	S	Δ	D
NSGA-II	91.7(2.7)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	**0.418**(0.036)	**1.000**(0.000)
GDE	**98.9**(1.3)	**0.000**(0.000)	**0.000**(0.000)	0.012(0.003)	0.764(0.045)	0.949(0.028)
GDE2	83.6(4.7)	**0.000**(0.000)	**0.000**(0.000)	0.011(0.001)	0.518(0.048)	**1.000**(0.000)
ZDT2	$\aleph$	ER	GD	S	Δ	D
NSGA-II	74.7(37.1)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	0.535(0.236)	0.800(0.402)
GDE	78.1(9.8)	**0.000**(0.001)	**0.000**(0.000)	0.018(0.005)	0.864(0.067)	0.978(0.024)
GDE2	**87.9**(4.4)	0.020(0.141)	**0.000**(0.000)	0.010(0.001)	**0.470**(0.052)	**1.000**(0.001)
ZDT3	$\aleph$	ER	GD	S	Δ	D
NSGA-II	**92.9**(2.3)	**0.000**(0.000)	**0.000**(0.000)	**0.006**(0.001)	**0.573**(0.036)	0.971(0.083)
GDE	69.1(4.8)	0.003(0.007)	**0.000**(0.000)	0.018(0.008)	1.044(0.068)	0.968(0.020)
GDE2	40.3(3.8)	0.007(0.014)	**0.000**(0.000)	0.020(0.005)	0.712(0.063)	**1.000**(0.001)
ZDT4	$\aleph$	ER	GD	S	Δ	D
NSGA-II	**95.5**(16.8)	**0.031**(0.113)	**0.001**(0.001)	**0.007**(0.001)	**0.389**(0.113)	0.971(0.172)
GDE	66.7(25.4)	0.235(0.357)	0.003(0.007)	0.026(0.015)	0.775(0.123)	0.968(0.052)
GDE2	55.2(21.1)	0.318(0.384)	0.004(0.006)	0.019(0.010)	0.532(0.067)	**1.006**(0.025)
ZDT6	$\aleph$	ER	GD	S	Δ	D
NSGA-II	89.5(2.9)	1.000(0.000)	0.008(0.001)	**0.008**(0.001)	0.513(0.031)	0.965(0.006)
GDE	**99.8**(2.1)	0.010(0.100)	0.002(0.021)	0.017(0.005)	1.018(0.079)	0.988(0.050)
GDE2	97.2(2.3)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	**0.388**(0.046)	**1.000**(0.001)

time than GDE and GDE2, probably because of additional operations, *e.g.*, the non-dominated sorting. GDE2 outperforms NSGA-II in the problem ZDT6.

6 Conclusions and Future Research

In this paper an extension of Generalized Differential Evolution algorithm is proposed. The extension, GDE2, adds to GDE a mechanism for improving extent and diversity of the obtained Pareto front approximation without impairing convergence speed of GDE and increasing execution time only little. GDE2 is

demonstrated to be effective, and does not introduce any extra control parameters to be preset by the user.

GDE and GDE2 were tested with a set of five benchmark multi-objective test problems. The numerical results show that GDE2 is able to provide a solution for all the test problems and performs comparably to NSGA-II and GDE providing a relatively good approximation of the Pareto front. However, the proposed method was found rather sensitive to control parameter values.

The effect of parameters on the optimization process, extensive comparison of GDE2 with latest multi-objective evolutionary algorithms and test problems, and applying GDE2 for practical multi-objective problems remains among the topics to be studied.

Acknowledgements

Authors wish to thank K. Deb and E. Zitzler for providing source codes of NSGA-II and results of SPEA respectively, and reviewers for useful commends.

References

1. Chang, C.S., Xu, D.Y., Quek, H.B.: Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. IEE Proceedings on Electric Power Applications **146** (1999) 577–583
2. Abbass, H.A., Sarker, R.: The Pareto Differential Evolution algorithm. International Journal on Artificial Intelligence Tools **11** (2002) 531–552
3. Madavan, N.K.: Multiobjective optimization using a Pareto Differential Evolution approach. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii (2002) 1145–1150
4. Babu, B.V., Jehan, M.M.L.: Differential Evolution for multi-objective optimization. In: Proceedings of the 2003 Congress on Evolutionary Computation, CEC'03, Canberra, Australia (2003) 2696–2703
5. Feng Xue, Arthur C. Sanderson, R.J.G.: Multi-objective differential evolution and its application to enterprise planning. In: Proceedings of IEEE International Conference on Robotics and Automation, Taiwan (2003) 3535–3541
6. Pareto, V.: Cours D'Economie Politique. Libraire Droz, Geneve (1964 (the first edition in 1896))
7. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1998)
8. Storn, R., Price, K.V.: Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11** (1997) 341–359
9. Lampinen, J., Storn, R.: Differential Evolution. In: New Optimization Techniques in Engineering. Springer (2004) 123–166
10. Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimization. McGraw-Hill, London (1999)
11. Storn, R., Price, K.V.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI (1995) [Online] Available: ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.gz, 3.5.2004.

12. Lampinen, J.: A constraint handling approach for the Differential Evolution algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii (2002) 1468–1473
13. Lampinen, J.: DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology (2001) [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 3.5.2004.
14. Lampinen, J.: Multi-constrained nonlinear optimization by the Differential Evolution algorithm. Technical report, Lappeenranta University of Technology, Department of Information Technology (2001) [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/DECONSTR.PDF, 3.5.2004.
15. Kukkonen, S., Lampinen, J.: A Differential Evolution algorithm for constrained multi-objective optimization: Initial assessment. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria (2004) 96–102
16. Kukkonen, S., Lampinen, J.: Mechanical component design for multiple objectives using Generalized Differential Evolution. In: Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM2004), Bristol, United Kingdom (2004) 261–272
17. Kukkonen, S., Lampinen, J.: Comparison of Generalized Differential Evolution algorithm to other multi-objective evolutionary algorithms. In: Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering, Jyväskylä, Finland (2004) Accepted for publication.
18. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186** (2000) 311–338
19. Deb, K.: Multi-Objective Optimization using Evolutionary algorithms. John Wiley & Sons, Chichester, England (2001)
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8** (2000) 173–195 Also available: ftp.tik.ee.ethz.ch/pub/people/zitzler/ZDT2000.ps, 15.1.2004.
21. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany (1999)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. IEEE Transactions on Evolutionary Computation **4** (1999) 257–271
23. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197

Publication IV

Kukkonen, S., Lampinen, J.,
GDE3: The third Evolution Step of Generalized Differential Evolution

IEEE Congress on Evolutionary Computation (CEC 2005), Edinburgh, UK, September, 2005, pages 443–450.

GDE3: The third Evolution Step of Generalized Differential Evolution

Saku Kukkonen and **Jouni Lampinen**
Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20, FIN-53851 Lappeenranta, Finland
saku.kukkonen@lut.fi

Abstract-

A developed version of Generalized Differential Evolution, GDE3, is proposed. GDE3 is an extension of Differential Evolution (DE) for global optimization with an arbitrary number of objectives and constraints. In the case of a problem with a single objective and without constraints GDE3 falls back to the original DE. GDE3 improves earlier GDE versions in the case of multi-objective problems by giving a better distributed solution. Performance of GDE3 is demonstrated with a set of test problems and the results are compared with other methods.

1 Introduction

During the last 15 years, Evolutionary Algorithms (EAs) have gained popularity in solving difficult multi-objective optimization problems (MOOPs) since EAs are capable of dealing with objective functions, which are not mathematically well behaving, *e.g.*, discontinuous, non-convex, multi-modal, and non-differentiable. Multi-objective EAs (MOEAs) are also capable of providing multiple solution candidates in a single run, which is desirable with MOOPs.

Differential Evolution (DE) is a relatively new EA and it has been gaining popularity during previous years. Several extensions of DE for multi-objective optimization have already been proposed. Some basic approaches just convert MOOPs to single-objective forms and use DE to solve these [3, 5, 36].

The first method extending DE for multi-objective optimization using the Pareto approach was Pareto-based DE approach [6]. Pareto Differential Evolution [4] was also mentioned about the same time, unfortunately without an explicit description of the method. After these, the Pareto(-frontier) Differential Evolution (PDE) algorithm [2] and a first version of Generalized Differential Evolution (GDE) [22] were introduced. Later on, Self-adaptive PDE (SPDE) [1], the Pareto DE Approach (PDEA) [26], Adaptive Pareto DE (APDE) [37], Multi-Objective DE (MODE) [13], and Vector Evaluated DE (VEDE) [29] have been proposed. The latest proposals are a second version of GDE [21] and DE for Multiobjective Optimization (DEMO) [32]. Research demonstrating the performance of PDEA over the elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [9] with rotated MOOPs has also been reported [17].

Besides solving problems with multiple objectives, DE has also been modified for handling problems with constraints [7, 23, 25, 27, 33, 35]. Most of these are based on applying penalty functions.

Earlier GDE versions had already the ability to handle any number of objectives and constraints. The latest version, GDE3, introduced in this paper is an attempt to improve earlier versions in the case of multiple objectives.

2 Multi-Objective Optimization with Constraints

Many practical problems have multiple objectives and several aspects cause multiple constraints to problems. For example, mechanical design problems have several objectives such as obtained performance and manufacturing costs, and available resources may cause limitations. Constraints can be divided into boundary constraints and constraint functions. Boundary constraints are used when the value of a decision variable is limited to some range, and constraint functions represent more complicated constraints, which are expressed as functions.

A mathematically constrained MOOP can be presented in the form:

$$\begin{aligned} &\text{minimize} && \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_M(\vec{x})\} \\ &\text{subject to} && (g_1(\vec{x}), g_2(\vec{x}), \ldots, g_K(\vec{x}))^T \leq \vec{0}. \end{aligned} \tag{1}$$

Thus, there are M functions to be optimized and K constraint functions. Maximization problems can be easily transformed to minimization problems and different constraints can be converted to form $g_j(\vec{x}) \leq 0$, Thereby the formulation in Eq. 1 is without loss of generality.

Typically, MOOPs are often converted to single-objective optimization problems by predefining weighting factors for different objectives, expressing the relative importance of each objective. Optimizing several objectives simultaneously without articulating the relative importance of each objective *a priori*, is often called Pareto-optimization. An obtained solution is Pareto-optimal if none of the objectives can be improved without impairing at least one other objective [28, p. 11]. If the obtained solution can be improved in such a way that at least one objective improves and the other objectives do not decline, then the new solution dominates the original solution. The objective of Pareto-optimization is to find a set of solutions that are not dominated by any other solution.

A set of Pareto-optimal solutions form a Pareto front, and an approximation of the Pareto front is called a set of non-dominated solutions. From the set of non-dominated solutions the decision-maker may pick a solution, which provides a suitable compromise between the objectives. This can be viewed as *a posteriori* articulation of the decision-makers preferences concerning the relative importance of each objective.

Later on in this paper, the obtained non-dominated set is referred to as a *solution*, and a member of a non-dominated

0-7803-9363-5/05/$20.00 ©2005 IEEE.

set or a population is referred to as a *vector* to distinguish these.

Weak dominance relation $\preceq$ between two vectors is defined such that $\vec{x}_1$ weakly dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \preceq \vec{x}_2$ iff $\forall i : f_i(\vec{x}_1) \leq f_i(\vec{x}_2)$. Dominance relation $\prec$ between two vectors is defined such that $\vec{x}_1$ dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \prec \vec{x}_2$ iff $\vec{x}_1 \preceq \vec{x}_2 \wedge \exists i : f_i(\vec{x}_1) < f_i(\vec{x}_2)$. The dominance relationship can be extended to take into consideration constraint values besides objective values. A constraint-domination $\prec_c$ is defined here so that $\vec{x}_1$ constraint-dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \prec_c \vec{x}_2$ iff any of the following conditions is true [22]:

- $\vec{x}_1$ is feasible and $\vec{x}_2$ is not.
- $\vec{x}_1$ and $\vec{x}_2$ are infeasible and $\vec{x}_1$ dominates $\vec{x}_2$ in constraint function space.
- $\vec{x}_1$ and $\vec{x}_2$ are feasible and $\vec{x}_1$ dominates $\vec{x}_2$ in objective function space.

The definition for weak constraint-domination $\preceq_c$ is analogous. This constraint-domination definition differs from the approach presented in [8, pp. 301–302] in the case of two infeasible vectors and was developed independently.

3 Differential Evolution

The DE algorithm [31, 34] was introduced by Storn and Price in 1995. Design principles in DE were simplicity, efficiency, and the use of floating-point encoding instead of binary numbers. Like a typical EA, DE has some random initial population, which is then improved using selection, mutation, and crossover operations. Several methods exist to determine a stopping criterion for EAs but usually a predefined upper limit for the number of generations or function evaluations to be computed provides an appropriate stopping condition.

In each generation DE goes through each decision vector $\vec{x}_{i,G}$ of the population and creates a corresponding trial vector $\vec{u}_{i,G}$. Here, i is an index of the vector in the population and G is a generation index. Creation of the trial vector is done as follows in the most common DE version, DE/rand/1/bin [30]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}\,, \text{(randomly selected,}\\
&\quad \text{except mutually different and different from } i)\\
&j_{rand} = \text{floor}\,(rand_i[0,1) \cdot D) + 1\\
&\text{for}(j = 1; j \leq D; j = j + 1)\\
&\{\\
&\quad \text{if}(rand_j[0,1) < CR \vee j = j_{rand})\\
&\qquad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G})\\
&\quad \text{else}\\
&\qquad u_{j,i,G} = x_{j,i,G}\\
&\}
\end{aligned}
\tag{2}
$$

The scaled difference between two randomly chosen vectors, $F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$, defines magnitude and direction of the mutation. When the difference is added to a third randomly chosen vector $\vec{x}_{r_3,G}$, this corresponds to the mutation of the third vector.

Both CR and F are user defined control parameters for the DE algorithm and they remain fixed during the whole execution of the algorithm. Parameter CR, controlling the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors instead of from the old vector $\vec{x}_{i,G}$. The condition "$j = j_{rand}$" is to make sure that at least one element is different compared to elements of the old vector. Parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$. In practice, CR controls the rotational invariance of the search, and its small value (e.g. 0.1) is practicable with separable problems while larger values (e.g. 0.9) are for non-separable problems. Control parameter F controls the speed and robustness of the search, *i.e.*, a lower value for F increases the convergence rate but also the risk of stacking into a local optimum.

The basic idea of DE is that the mutation is self-adaptive to the objective function surface and to the current population in the same way as in Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [16] but without the computational burden of covariance matrix calculations that are scaling unfavorably with the dimensionality of the problem. At the beginning of generations the magnitude of the mutation is large because vectors in the population are far away in the search space. When evolution proceeds and the population converges, the magnitude of the mutation gets smaller. The self-adaptability of DE permits a global search

A trial vector $\vec{u}_{i,G}$ created by mutation and crossover operations is compared to an old vector $\vec{x}_{i,G}$. If the trial vector has an equal or better objective value, then it replaces the old vector in the next generation. Therefore, the average objective value of the population will never worsen making DE an elitist method.

4 Generalized Differential Evolution

The first version of a Generalized Differential Evolution (GDE) extended DE for constrained multi-objective optimization and was obtained by modifying the selection rule of the basic DE [22]. The basic idea in the selection rule was that the trial vector was selected to replace the old vector in the next generation if it weakly constraint-dominated the old vector. There was no sorting of non-dominated vectors during the optimization process or any mechanism for maintaining the distribution and extent of the solution. Also, there was no extra repository for non-dominated vectors. Still, GDE was able to provide a surprisingly good solution but was too sensitive for the selection of the control parameters [20].

Later on GDE was modified to make a decision based on the crowdedness when the trial and old vector were feasible and non-dominating each other in the objective function space [21]. This improved the extent and distribution of the solution but slowed down the convergence of the overall population because it favored isolated vectors far from the Pareto front before all the vectors were converged near the Pareto front. This version, GDE2, was still too sensitive for the selection of the control parameters.

The third version of GDE proposed in this paper extending the DE/rand/1/bin method to problems with M objectives and K constraint functions formally presented in Eq. 3.

$$
\begin{array}{l}
\text{Input} : D, G_{max}, NP \geq 4, F \in (0, 1+], CR \in [0,1], \text{and initial bounds: } \vec{x}^{(lo)}, \vec{x}^{(hi)} \\
\text{Initialize} : \left\{ \begin{array}{l} \forall i \leq NP \wedge \forall j \leq D : x_{j,i,G=0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)} \right) \\ i = \{1, 2, \ldots, NP\}, j = \{1, 2, \ldots, D\}, G = 0, m = 0, rand_j[0,1) \in [0,1), \end{array} \right. \\
\left\{ \begin{array}{l}
\text{While } G < G_{max} \\
\quad \forall i \leq NP \left\{ \begin{array}{l}
\text{Mutate and recombine:} \\
r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \text{randomly selected,} \\
\qquad \text{except mutually different and different from } i \\
j_{rand} \in \{1, 2, \ldots, D\}, \text{randomly selected for each } i \\
\\
\forall j \leq D, u_{j,i,G} = \left\{ \begin{array}{ll} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) & \\ \qquad \text{if} \quad rand_j[0,1) < CR \vee j = j_{rand} & \\ x_{j,i,G} \quad \text{otherwise} & \end{array} \right. \\
\text{Select :} \\
\vec{x}_{i,G+1} = \left\{ \begin{array}{ll} \vec{u}_{i,G} & \text{if} \quad \vec{u}_{i,G} \preceq_c \vec{x}_{i,G} \\ \vec{x}_{i,G} & \text{otherwise} \end{array} \right. \\
\\
\boxed{\begin{array}{l} \text{Set :} \\ \\ \quad m = m + 1 \\ \quad \vec{x}_{NP+m,G+1} = \vec{u}_{i,G} \end{array} \quad \text{if} \left\{ \begin{array}{l} \forall j : g_j(\vec{u}_{i,G}) \leq 0 \\ \wedge \\ \vec{x}_{i,G+1} == \vec{x}_{i,G} \\ \wedge \\ \vec{x}_{i,G} \not\prec \vec{u}_{i,G} \end{array} \right.}
\end{array} \right. \\
\\
\boxed{\begin{array}{l}
\text{While } m > 0 \\
\quad \text{Select } \vec{x} \in \{\vec{x}_{1,G+1}, \vec{x}_{2,G+1}, \ldots, \vec{x}_{NP+m,G+1}\} : \\
\qquad \left\{ \begin{array}{l} \forall i \quad \vec{x} \not\prec_c \vec{x}_{i,G+1} \\ \wedge \\ \forall (\vec{x}_{i,G+1} : \vec{x}_{i,G+1} \not\prec_c \vec{x}) \quad CD(\vec{x}) \leq CD(\vec{x}_{i,G+1}) \end{array} \right. \\
\quad \text{Remove } \vec{x} \\
\quad m = m - 1
\end{array}} \\
\\
G = G + 1
\end{array} \right.
\end{array}
\tag{3}
$$

Notation *CD* means Crowding Distance [9], which approximates the crowdedness of a vector in its non-dominated set. Also, some other distance measure for crowdedness could be used. The parts that are new compared to previous GDE versions are framed in Eq. 3. Without these parts, the algorithm is identical to the first GDE version. Later on in this paper the proposed method given in Eq. 3 is called the *Generalized Differential Evolution 3 (GDE3)*. It handles any number of M objectives and any number of K constraints, including the cases where $M = 0$ (constraint satisfaction problem) and $K = 0$ (unconstrained problem), and the original DE is a special case of GDE3. GDE3 can been seen as a combination of earlier GDE versions and PDEA [26]. A similar approach was also proposed in DEMO [32] without constraint handling, and DEMO does not fall back to the original DE in the case of single objective as GDE3 does.

Selection in GDE3 is based on the following rules:

- In the case of infeasible vectors, the trial vector is selected if it weakly dominates the old vector in constraint violation space, otherwise the old vector is selected.
- In the case of the feasible and infeasible vectors, the feasible vector is selected.
- If both vectors are feasible, then the trial is selected if it weakly dominates the old vector in the objective function space. If the old vector dominates the trial vector, then the old vector is selected. If neither vector dominates each other in the objective function space, then both vectors are selected for the next generation.

After a generation, the size of the population may have been increased. If this is the case it is then decreased back to the original size based on a similar selection approach used

in NSGA-II. Vectors are sorted based on non-dominance and crowdedness. The worst population members according to these measurements are removed to decrease the size of the population to the original size. Non-dominated sorting is modified to take into consideration also constraints, and selection based on Crowding Distance is improved over the original method of NSGA-II to provide a better distributed set of vectors [19].

When $M = 1$ and $K = 0$, there are no constraints to be evaluated and the selection is simply

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if } f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}, \quad (4)$$

which is same as for the original DE. The population is not increased because this requires that $\vec{u}_{i,G}$ and $\vec{x}_{i,G}$ do not weakly dominate each other, which cannot be true in the case of a single objective. Since the population is not increased, there is no need to remove elements. Therefore, GDE3 is identical to the original DE in this case. This makes it possible to change DE/rand/1/bin strategy to any other DE strategy such as presented in [14,37] or, generally, to any method where a child vector is compared against a parent vector and a better one is preserved.

In NSGA-II and PDEA, the size of the population after a generation is $2NP$, which is then decreased to size NP. In GDE3 and DEMO, the size of the population after a generation is between NP and $2NP$ because the size of the population is increased only if the trial and the old vector are feasible and do not dominate each other.[1] Decreasing the size of the population at the end of a generation is the most complex operation in the algorithm. This needs non-dominated sorting, which in GDE3 uses the concept of constraint-domination defined in 2. Non-dominated sorting can be implemented to run in time $O\left(N \log^{M-1} N\right)$ [18]. Also, niching is done for non-dominated members of the population, which is a complex operation if clustering techniques are applied. Instead of clustering, niching is performed using an approximate distance measure, Crowding Distance, which can be calculated in time $O(MN \log N)$ [18]. Overall running time for GDE3 in Eq. 3 is $O\left(G_{max} N \log^{M-1} N\right)$ for large N.

GDE3 can be implemented in such a way that the number of function evaluations is reduced because not always all the constraints and objectives need to be evaluated, *e.g.*, inspecting constraint violations (even one constraint) is often enough to determine, which vector to select for the next generation [23,31]. However, in the case of feasible vectors all the objectives need to be evaluated.

5 Experiments

GDE3 was evaluated with a set of test problems available from the literature [8, 10, 11]. The idea was to select known representative problems from different problem type categories. In repeated tests, a standard two-sample t-test was used to evaluate the significance of the obtained numerical results. Suitable control parameter values of GDE3 for each problem were found based on problem characteristics and by trying out a couple of different control parameter values.

5.1 Singe-Objective Optimization

The performance of GDE3 in the case of single-objective optimization is illustrated with two classical multi-modal test problems, Rastrigin's and Schwefel's functions with 20 variables. Since both problems are separable, a low value for CR was used. The control parameter value F was set as low as possible while still obtaining a global optimum. The control parameters were $CR = 0.0$, $F = 0.5$ for Rastrigin's function and $CR = 0.2$, $F = 0.4$ for Schwefel's function.

The test functions, initialization ranges, used population sizes, desired target values, and a number of needed function evaluations as shown in Table 1. A minimum, mean, and maximum number of function evaluations after 100 independent runs are reported. The number of needed function evaluations were significantly smaller than with the Omni-Optimizer reported in [12].

5.2 Bi-Objective Test Problem

Improved selection based on the Crowding Distance is demonstrated with a simple bi-objective optimization problem, which is defined as [8, p. 176]:

$$\begin{array}{ll} \text{Minimize} & f_1(\vec{x}) = x_1 \\ \text{Minimize} & f_2(\vec{x}) = \frac{1+x_2}{x_1} \\ \text{subject to} & x_1 \in [0.1, 1], x_2 \in [0, 5] \end{array} \quad (5)$$

This problem is relatively easy to solve for MOEAs, and GDE3 finds a solution converged to the Pareto front in about 20 generations. The problem was solved with GDE3 and NSGA-II having a population size of 100 and 500 generations. Control parameters for GDE3 were $CR = 0.2$ and $F = 0.2$, and for NSGA-II $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 20$, and $\eta_m = 20$ [9]. A large number of generations were used to make sure that the obtained solution converged to the Pareto front and only the diversity of the solution was measured. Results after one run are shown in Figure 1. A better distribution obtained for the solution for GDE3 than NSGA-II can be observed with a careful view.

The problem was solved 100 times and diversity was measured using spacing (S) [8, pp. 327–328], which measures the standard deviation of the distances from each vector to the nearest vector in the obtained non-dominated set. A small value for S is better, and $S = 0$ for ideal distribution. Mean and standard deviations for spacing are 0.0030 ± 0.0003 and 0.0074 ± 0.0007 for GDE3 and NSGA-II, respectively. GDE3 has more than double the lower spacing value than NSGA-II has, *i.e.*, GDE3 obtains a better distributed solution than NSGA-II in the case of this problem.

5.3 Bi-Objective Mechanical Design Problem

A bi-objective spring design problem [8, pp. 453–455] was selected to demonstrate the GDE3's ability to handle several

[1] GDE3 could be modified to preserve the old and the trial vector in the case of constrained-non-domination, but this would increase number of function evaluations needed and slow down convergence.

Function	$f(\vec{x})$	D	Range	N	f^*	Min	Mean	Max
Rastrigin	$\sum_{i=1}^{D} {x_i}^2 + 10\,(1 - \cos(2\pi x_i))$	20	[-10, 10]	20	0.01	8029 (19260)	9085 (24660)	10184 (29120)
Schwefel	$418.982887D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	20	[-500, 500]	50	0.01	14996 (54950)	16540 (69650)	18479 (103350)

Table 1: Single-objective optimization problems, initialization ranges, population size, desired target value, and the needed number of function evaluations for GDE3. Results reported in [12] for the Omni-Optimizer are in parenthesis.

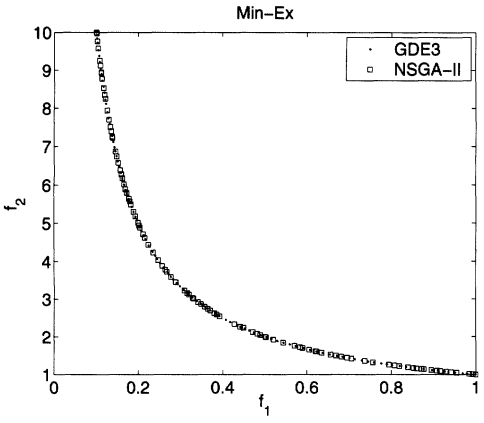


Figure 1: A simple bi-objective optimization problem solved with GDE3 and NSGA-II.

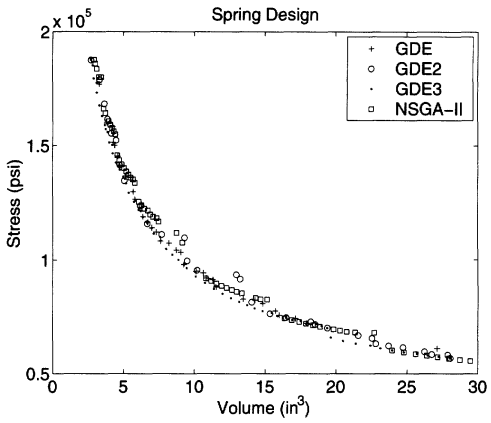


Figure 2: The spring design problem solved with GDE, GDE2, GDE3 and NSGA-II.

constraints and different types of decision variables. GDE3 uses real-coded variables as genotypes, which are converted to corresponding phenotypes before evaluation of the objective and constraint functions [31].

The problem is to design a helical compression spring, which has a minimum volume and minimal stress. Objective functions are nonlinear and the problem has three variables: the number of spring coils x_1 (integer), the wire diameter x_2 (discrete having 42 non-equispaced values), and the mean coil diameter x_3 (real). Besides the boundary constraints, the problem has eight inequality constraint functions from which most are nonlinear.

Results for the different GDE versions and NSGA-II after a single run are shown in Figure 2. The size of the population and the number of generations were 100 for the methods. Control parameter values for GDEs were $CR = 0.9$ and $F = 0.5$. The control parameters for NSGA-II were $p_c = 1.0$, $p_m = 1/D$, $\eta_c = 10$, and $\eta_m = 500$ used in [8, pp. 450]. The number of needed function evaluations for the GDEs are reported in Table 2. NSGA-II needed 10000 function evaluations for each objective and constraint function.

In preliminary tests GDE3 was found to be more stable than earlier GDE versions for the selection of the control parameters. In these tests, GDE and GDE2 also performed poorer compared to GDE3 and therefore they were excluded from further comparison in this paper.

5.4 Constrained Bi-Objective Test Problems

Constrained bi-objective test problems CTP1 and CTP2 [10] having $D = 6$, $x_i \in [0, 1]$, and function $g(\vec{x}) = 1 + \sum_{j=2}^{D} {x_j}^2$ controlling difficulty to converge to the Pareto front were used. These problems were solved 100 times. The size of the population was 100 and the number of generations was 50. Control parameters for GDE3 were $CR = 0.9$ and $F = 0.1$, and for NSGA-II $p_c = 0.9$, $p_m = 1/D$, $\eta_c = 20$, and $\eta_m = 20$ used in [10].

Results were compared using spacing and binary metrics set coverage $\mathcal{C}$ metric [8, pp. 325–326] and a $\mathcal{V}$ measure [15, 24]. The $\mathcal{C}(\mathrm{A}, \mathrm{B})$ metric measures the fraction of members of B that are dominated by members of A. The $\mathcal{V}(\mathrm{A}, \mathrm{B})$ measures the fraction of the volume of the minimal hypercube containing both fronts that is dominated by the members of A but is not dominated by the members of B. Greater values for $\mathcal{C}$ and the $\mathcal{V}$ metrics are desirable.

The results shown in Table 3. With CTP1, spacing (S) shows strongly and $\mathcal{V}$ metric slightly that GDE3 performs better but $\mathcal{C}$ metric shows strongly opposite implying that NSGA-II has converged closer to the Pareto front. With CTP2, there is no significant difference between obtained S values and the binary metrics show contradicting results. Contradicting results for binary metrics are due to the fact that the $\mathcal{C}$ metric emphasizes convergence over diversity whereas the $\mathcal{V}$ metric considers both issues.

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	f_1	f_2
GDE	10100	8990	8901	8778	8115	8115	5704	5529	5515	2920
GDE2	10100	8504	8419	8008	7741	7741	6348	5874	5846	5846
GDE3	10100	8961	8879	8548	8319	8317	4885	4587	4566	4566

Table 2: Number of needed constraint (g_j) and objective (f_i) function evaluations needed by GDE, GDE2, and GDE3 for the spring design problem.

	S(G)	S(N)	$\mathcal{C}$(G,N)	$\mathcal{C}$(N,G)	$\mathcal{V}$(G,N)	$\mathcal{V}$(N,G)
CTP1	**0.0048±**	0.0075 ±	0.1303 ±	**0.2023±**	**0.0034±**	0.0027 ±
	0.0039	0.0026	0.0415	**0.0904**	**0.0015**	0.0016
CTP2	**0.0092±**	0.0113 ±	0.2655 ±	**0.3588±**	**0.0031±**	0.0022 ±
	0.0073	0.0085	0.0742	**0.0685**	**0.0011**	0.0006
DTLZ1	**0.0179±**	0.0274 ±	**0.3842±**	0.0021 ±	**0.0046±**	0.0012 ±
	0.0007	0.0171	**0.1449**	0.0100	**0.0033**	0.0011
DTLZ4	**0.0214±**	0.0238 ±	**0.0948±**	0.0123 ±	**0.0085±**	0.0059 ±
	0.0010	0.0009	**0.0240**	0.0066	**0.0007**	0.0008

Table 3: Spacing (S), $\mathcal{C}$, and $\mathcal{V}$ metrics for the CTP and DTLZ problems (G = GDE3 and N = NSGA-II).

5.5 Tri-Objective Test Problems

Finally, GDE3 was used to solve problems with more than two objectives. Tri-objective test problems DTLZ1 and DTLZ4 [11] were selected for this purpose. The size of the population was 500 and the number of generations was 150 for DTLZ1 and 50 for DTLZ4. Control parameters for GDE3 were $CR = 0.2$ and $F = 0.2$, and for NSGA-II $p_c = 1.0$, $p_m = 1/D$, $\eta_c = 15$, and $\eta_m = 20$ used in [11]. Results after a single run are shown in Figures 3–5. Tests were repeated 100 times and the same metrics were measured as for the CTP problems earlier.[2] Obtained values are reported in Table 3. GDE3 outperforms NSGA-II with these problems according to metrics.

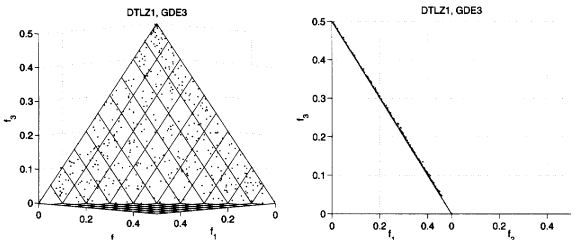


Figure 3: Two projections of the result for the DTLZ1 problem solved with GDE3.

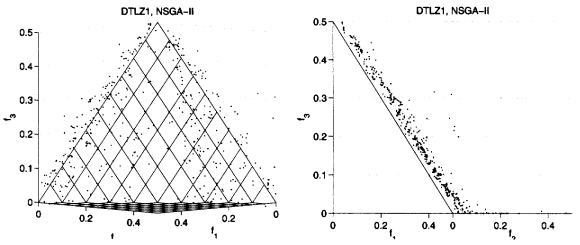


Figure 4: Two projections of the result for the DTLZ1 problem solved with NSGA-II.

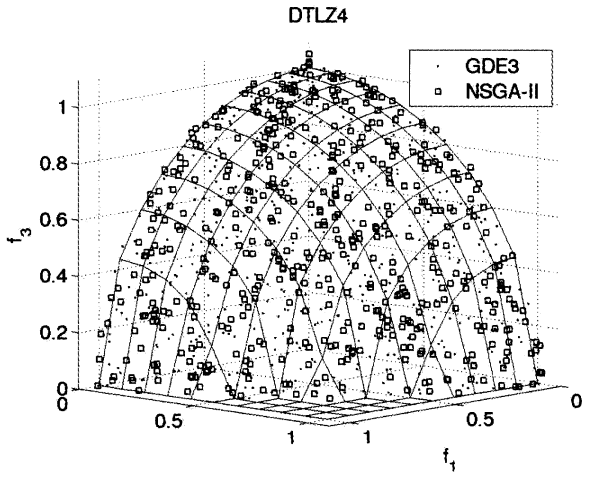


Figure 5: The DTLZ4 problem solved with GDE3 and NSGA-II.

6 Conclusions and Future Research

The third evolution version of Generalized Differential Evolution, GDE3, is proposed. GDE3 is designed for any number of objectives and constraints without introducing any extra control parameters to the original DE. In the case

[2]Even thought spacing might not give reliable result when the number of objectives is greater that two [11].

of unconstrained single-objective optimization problems, GDE3 is exactly the same as the original DE.

GDE3 modifies earlier GDE versions using a growing population and non-dominated sorting with pruning of non-dominated solutions to decrease the population size at the end of each generation. This improves obtained diversity and makes the method more stable for the selection of control parameter values. The constraint handling method used in GDEs reduces the number of needed function evaluations.

GDE3 was tested with a set of different types of test problems and results show an improved diversity of the solution over the NSGA-II method as well as demonstrating a reduction in the number of needed function evaluations. In some test problems, GDE3 found also a better converged solution. However, results are based on limited tests with a limited number of test problems and they are mainly indicative.

A more extensive comparison of GDE3 with other multi-objective DE methods, latest multi-objective evolutionary algorithms and test problems, parallelization of the algorithm, and applying GDE3 for practical constrained multi-objective problems remains as future work.

Acknowledgments

S. Kukkonen gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), the Academy of Finland, Technological Foundation of Finland, and Finnish Cultural Foundation. He also wishes to thank Dr. K. Deb and all the KanGAL people for help and fruitful discussions.

Bibliography

[1] H. A. Abbass. The self-adaptive Pareto Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 831–836, Honolulu, Hawaii, May 2002.

[2] H. A. Abbass, R. Sarker, and C. Newton. PDE: a Pareto-frontier Differential Evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 971–978, Seoul, South Korea, 2001.

[3] B. V. Babu and M. M. L. Jehan. Differential Evolution for multi-objective optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 2696–2703, Canberra, Australia, Dec 2003.

[4] P. K. Bergey. An agent enhanced intelligent spreadsheet solver for multi-criteria decision making. In *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 1999)*, pages 966–968, Milwaukee, Wisconsin, USA, Aug 1999.

[5] C. S. Chang and D. Y. Xu. Differential Evolution based tuning of fuzzy automatic train operation for mass rapid transit system. *IEE Proceedings on Electric Power Applications*, 147(3):206–212, May 2000.

[6] C. S. Chang, D. Y. Xu, and H. B. Quek. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. *IEE Proceedings on Electric Power Applications*, 146(5):577–583, Sept 1999.

[7] T.-T. Chang and H.-C. Chang. Application of Differential Evolution to passive shunt harmonic filter planning. In *8th International Conference On Harmonics and Quality of Power*, pages 149–153, Athens, Greece, Oct 1998.

[8] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, England, 2001.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[10] K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 284–298, Zurich, Switzerland, March 2001.

[11] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 825–830, Honolulu, Hawaii, May 2002.

[12] K. Deb and S. Tiwari. Omni-optimizer: A procedure for single and multi-objective optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 47–61, Guanajuato, Mexico, March 2005.

[13] R. J. G. Feng Xue, Arthur C. Sanderson. Pareto-based multi-objective Differential Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 862–869, Canberra, Australia, Dec 2003.

[14] V. Feoktistov and S. Janaqi. New strategies in Differential Evolution. In *Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM2004)*, pages 335–346, Bristol, United Kingdom, April 2004.

[15] J. E. Fieldsend, R. M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, June 2003.

[16] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation sistributions in Evolution Strategies: The covariance matrix adaptation. In *Proceedings of the 1996 Congress on Evolutionary Computation (CEC 1996)*, pages 312–317, Nayoya University, Japan, May 1996.

[17] A. Inorio and X. Li. Solving rotated multi-objective optimization problems using Differential Evolution. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004)*, pages 861–872, Cairns, Australia, Dec 2004.

[18] M. T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515, Oct 2003.

[19] S. Kukkonen and K. Deb. Improved pruning of non-dominated solutions based on crowding distance. Unpublished manuscript, 2005.

[20] S. Kukkonen and J. Lampinen. Comparison of Generalized Differential Evolution algorithm to other multi-objective evolutionary algorithms. In *Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS2004)*, page 445, Jyväskylä, Finland, July 2004.

[21] S. Kukkonen and J. Lampinen. An extension of Generalized Differential Evolution for multi-objective optimization with constraints. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 752–761, Birmingham, Finland, Sept 2004.

[22] J. Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001. [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 1.4.2005.

[23] J. Lampinen. A constraint handling approach for the Differential Evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 1468–1473, Honolulu, Hawaii, May 2002.

[24] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, volume 1, pages 46–53, Piscataway, NJ, 2000.

[25] Y.-C. Lin, K.-S. Hwang, and F.-S. Wang. Hybrid Differential Evolution with multiplier updating method for nonlinear constrained optimization problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 872–877, Honolulu, Hawaii, May 2002.

[26] N. K. Madavan. Multiobjective optimization using a Pareto Differential Evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 1145–1150, Honolulu, Hawaii, May 2002.

[27] E. Mezura-Montes, C. A. Coello Coello, and E. I. Tun-Morales. Simple feasibility rules and Differential Evolution for constrained optimization. In *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI 2004)*, pages 707–716, Mexico City, Mexico, April 2004.

[28] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1998.

[29] K. E. Parsopoulos, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. Vector evaluated Differential Evolution for multiobjective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, pages 204–211, Portland, Oregon, USA, June 2004.

[30] K. V. Price. *New Ideas in Optimization*, chapter An Introduction to Differential Evolution, pages 79–108. McGraw-Hill, London, 1999.

[31] K. V. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, 2005.

[32] T. Robič and B. Filipič. DEMO: Differential Evolution for multiobjective optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 520–533, Guanajuato, Mexico, March 2005.

[33] R. Storn. System design by constraint adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, April 1999.

[34] R. Storn and K. V. Price. Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[35] F.-S. Wang and J.-P. Chiou. Differential Evolution for dynamic optimization of differential-algebraic systems. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 531–536, Indianapolis, IN, 1997.

[36] F.-S. Wang and J.-W. Sheu. Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. *Chemical Engineering Science*, 55(18):3685–3695, Sept 2000.

[37] D. Zaharie. Multi-objective optimization with adaptive Pareto Differential Evolution. In *Proceedings of Symposium on Intelligent Systems and Applications (SIA 2003)*, Iasi, Romania, Sept 2003.

Publication V

Kukkonen, S., Deb, K.,
Improved Pruning of Non-Dominated Solutions Based on Crowding Distance for Bi-Objective Optimization Problems

IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada, July, 2006, pages 3995–4002.

2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Improved Pruning of Non-Dominated Solutions Based on Crowding Distance for Bi-Objective Optimization Problems

Saku Kukkonen and Kalyanmoy Deb, *Senior Member, IEEE*

***Abstract*— In this paper an algorithm for pruning a set of non-dominated solutions is proposed. The algorithm is based on the crowding distance calculation used in the elitist non-dominated sorting genetic algorithm (NSGA-II). The time complexity class of the new algorithm is estimated and in most cases it is the same as for the original pruning algorithm. Numerical results also support this estimate.**

For used bi-objective test problems, the proposed pruning algorithm is demonstrated to provide better distribution compared to the original pruning algorithm of NSGA-II. However, with tri-objective test problems there is no improvement and this study reveals that crowding distance does not estimate crowdedness well in this case and presumably also in cases of more objectives.

I. Introduction

Pruning a set of non-dominated solutions is a common task for multi-objective evolutionary algorithms (MOEAs) such as the strength Pareto evolutionary algorithm (SPEA2) [1] and the elitist non-dominated sorting genetic algorithm (NSGA-II) [2]. An idea is to prune a non-dominated set to have desired number of solutions in such a way that remaining solutions would have as good diversity as possible, *i.e.*, the spread of extreme solutions is as high as possible and the relative distance between solutions is as equal as possible. Probably the best way to obtain a good distribution would be to use some clustering algorithm as in SPEA2. However, this is computationally expensive since clustering algorithms take usually time $O\left(MN^2\right)$ to prune a set of size N with M objectives [3]. In NSGA-II the pruning of non-dominated solutions is done in time $O\left(MN\log N\right)$ based on *crowding distance (*CD*)*. However, this method often gives non-optimal distribution as it is demonstrated later in this paper. Therefore, an improved pruning based on *CD* is proposed here.

Besides NSGA-II, *CD* has been used as a distribution maintenance method in many other MOEAs [4]–[13]. There also exist studies where the original *CD* has been slightly modified to use, *e.g.*, different kind of normalization techniques for objective values [14], polar coordinates in the objective space [15], problem specific measures [16], and a different distance metric for crowdedness calculation [17].

This paper continues with the following parts: In Section II the original pruning algorithm of NSGA-II is described and its drawback in some cases is demonstrated. Section III describes the proposed pruning algorithm and its time complexity is analyzed in Section IV. The proposed method is tested in Section V and observations are discussed in Section VI. Finally, conclusions are given in Section VII.

Saku Kukkonen is with the Department of Information Technology, Lappeenranta University of Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland; email: saku.kukkonen@lut.fi.

Kalyanmoy Deb is with the Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology Kanpur, Kanpur, PIN 208 016, India; email: deb@iitk.ac.in.

II. Pruning of Non-Dominated Solutions in NSGA-II

At the end of one generation of NSGA-II, the size of the population is twice bigger than the original size. This bigger population is pruned based on the non-dominated sorting [18, pp. 33–44], [3] and *CD*. *CD* for a member of a non-dominated set tries to approximate the perimeter of a cuboid formed by using the nearest neighbors of the member. The cuboid of a non-dominated set member i in the case of two objectives is illustrated in Figure 1 (objectives are minimized in this paper). For a member of non-dominated set, *CD* is calculated by finding distance between two nearest solutions on either side of the member along each of the objectives. These distances are normalized dividing them by the difference between maximum and minimum values of corresponding objectives, and then these normalized distances are summed up giving a *CD* for the corresponding member. For those members of the non-dominated, which have maximum or minimum value for any objective, *CD* is assigned to have an infinite value. Finally, the members of the non-dominated set are sorted in monotonically decreasing order according to *CD*s and a desired number of members having the largest *CD* values are selected.

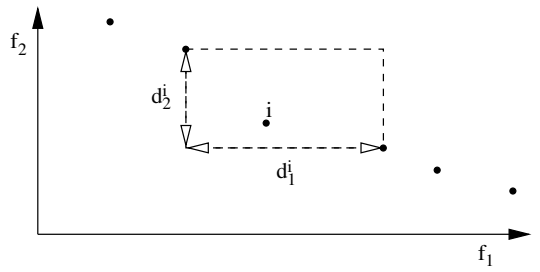


Fig. 1. An example of the cuboid of a solution i in the case of two objectives.

For selecting n members out of N based on *CD*, members are first sorted according to objective values taking time $O\left(N\log N\right)$ for each of M objective. Then *CD*s are calculated taking time $O\left(MN\right)$. Finally, members are sorted according to the *CD* values taking time $O\left(N\log N\right)$. The overall computation time of the algorithm is dominated by

0-7803-9487-9/06/$20.00/©2006 IEEE

the first sorting of members according to the objective values for each objective taking time $O(MN \log N)$.

Although the idea of this algorithm is reasonable, it does not provide good result in all cases. In Figure 2 a non-dominated set of 11 members and the six selected members according to *CD*s calculated for the members are presented in the case of two objectives. As it can be seen, leaving out the rest five members having the smallest *CD*s leaves a gap in the resulting set and it is clear that this set of solutions is not optimal in the sense of distribution. There are also other cases where pruning based on *CD* does not provide good distribution [17].

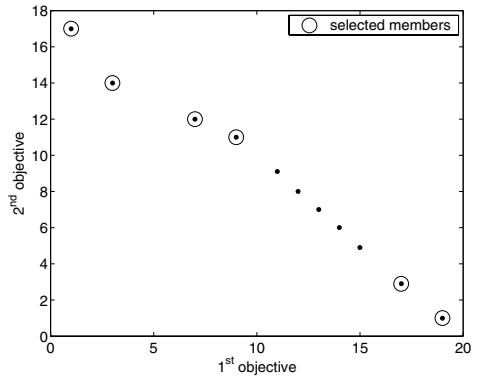


Fig. 2. A set of 11 non-dominated members from which six are selected based on the pruning method used in NSGA-II.

III. Proposed Algorithm

The proposed algorithm first calculates *CD*s for the members of a non-dominated set. Instead of selecting n members having the largest *CD* values, $N - n$ members having the smallest *CD* values are removed one by one, updating the *CD* values for the remaining members of the set after each removal. The efficient implementation of this algorithm needs an implementation of a priority queue such as a heap [19, pp. 140–152]. The proposed algorithm is:

Pruning of non-dominated set

input: a non-dominated set $\mathcal{F}$,
the size n of a desired pruned set
output: elements of a heap H

1 calculate *CD* for each member of the set $\mathcal{F}$
2 create a data structure D containing information about neighbors on either side of the members of $\mathcal{F}$ along each objectives
3 create an ascending heap H from the members of $\mathcal{F}$ using *CD*s as ordering keys
4 while $|H| > n$
5 remove an element with a minimum *CD* value from H and update H
6 update D to have correct neighbor information for the neighbors of the removed element
7 for all the neighbors of the removed element
8 calculate a new *CD*
9 replace old *CD* value in H with the new one and update H

The output of this algorithm is demonstrated for the same non-dominated set as with the original pruning algorithm. Figure 3 shows the output, which is better distributed than in Figure 2. The proposed method would, presumably, also improve the performance of other methods based on *CD*, *e.g.*, those mentioned in Section I.

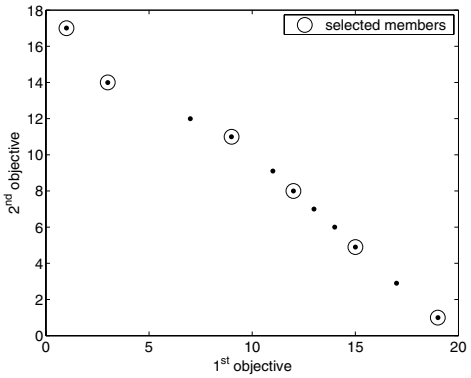


Fig. 3. A set of 11 non-dominated members from which six are selected based on the proposed pruning algorithm.

IV. Complexity Analysis of the Algorithm

The first operation in line 1 can be done in time $O(MN \log N)$ as in the original algorithm. Creating data structure D containing the neighborhood information along each of the objectives in line 2 can be done in time $O(MN)$. The creation of a heap takes time $O(N)$. The while-loop in lines 4–9 is executed at most N times, on an average $N/2$ times. Removing a minimum element from the heap and updating the heap to have correct structure in line 5 takes time $O(\log N)$. Updating D in line 6 takes time $O(M)$ since the removed element has at most $2M$ neighbors along the objectives. Calculating new *CD* for a neighbor element with the help of D in line 8 takes time $O(M)$ since there are at most $2M$ objective values to be taken for calculations. Updating *CD* value to the heap and updating the heap to have correct structure in line 9 takes time $O(\log N)$. Therefore, the computation time for the whole for-loop in lines 7–9 is bounded by $O(M \log N)$ or $O(M^2)$, whichever is greater. The while-loop in lines 4–9 is bounded by $O(MN \log N)$ or $O(M^2 N)$, whichever is greater. These are also overall complexities for the whole algorithm. In many cases, where $M < \log N$, the former complexity dominates. This leads to complexity class $O(MN \log N)$ for the algorithm. Anyway, for large N the proposed algorithm is faster than the typical clustering methods except in some rare cases when $M > N$.

V. Experiments

The original and proposed pruning algorithms were implemented in the Generalized Differential Evolution 3 (GDE3) [12], which was then used to solve test problems.

GDE3 is an extension of Differential Evolution (DE) [20] for constrained multi-objective optimization. Roughly speaking, the evolutionary part of the algorithm is DE and the multi-objective part is from NSGA-II [2]. This kind of combination has been shown to give benefit over NSGA-II with rotated problems [21]. Further, GDE3 has some other improvements over NSGA-II, and an interested reader is advised to look reference [12].

In repetition tests, the diversity of obtained result was measured using the spacing (S) metric [18, pp. 327–328] and variance of *CD*s. The spacing metric measures the standard deviation of distances (according to Manhattan, *i.e.*, L_1 distance metric) from each vector to nearest vector in the obtained non-dominated set. A smaller value of S is better and for an ideal distribution $S = 0$. Also, the CPU time needed for pruning and for the entire process was registered. The hardware used was a PC with 2.6 GHz Pentium 4 CPU & 512 MB RAM, and the operating system was Linux.

A. Bi-Objective Problems

The pruning algorithms were used to solve the bi-objective test problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [18, pp. 356–360], using population size 100 and 1000 generations. Sets of solutions to these problems are shown in Figures 4–8, where the solutions generated by using the proposed pruning method have been sifted by -0.05 units along both objectives to alleviate observation. As it can be seen, the solutions obtained with the proposed pruning method have better distribution.

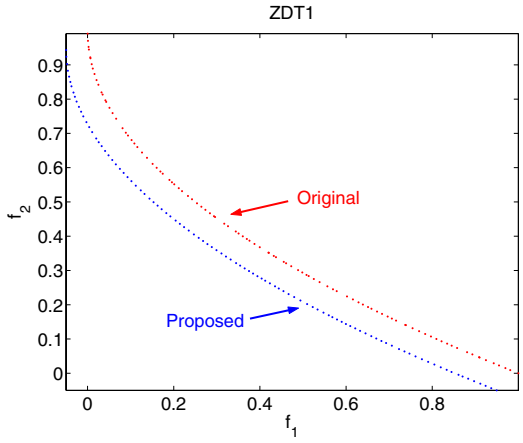


Fig. 4. A result for ZDT1 using the original and proposed pruning methods (the solutions obtained by the proposed method are deliberately moved towards the origin for clarity).

Solving the bi-objective optimization problems was repeated 100 times. The numerical results, *i.e.*, mean and standard deviation values are reported in Table I. The spacing metric suggests that the results obtained with the proposed pruning method are almost three times better compared to those with the original method in terms of diversity, while the total required CPU time has increased less than 10%.

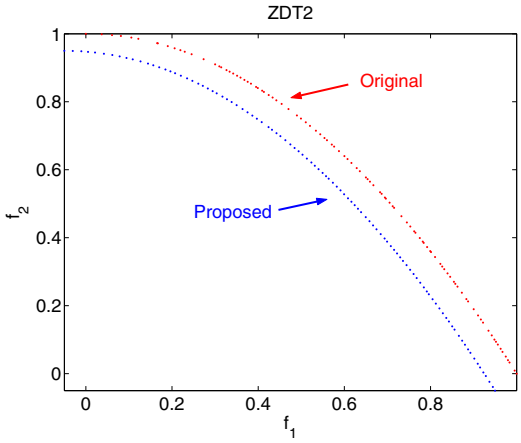


Fig. 5. A result for ZDT2 using the original and proposed pruning methods (the solutions obtained by the proposed method are deliberately moved towards the origin for clarity).

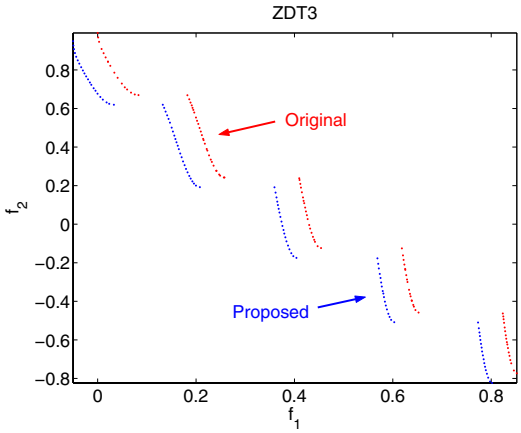


Fig. 6. A result for ZDT3 using the original and proposed pruning methods (the solutions obtained by the proposed method are deliberately moved towards the origin for clarity).

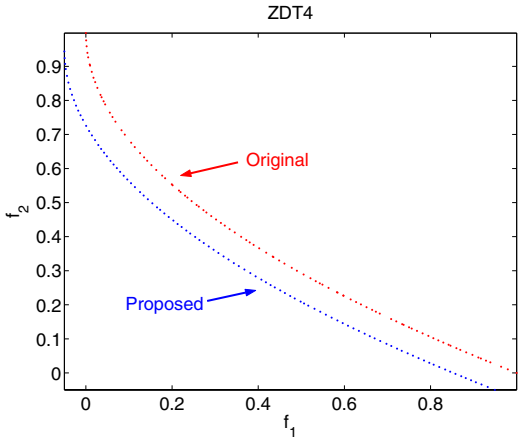


Fig. 7. A result for ZDT4 using the original and proposed pruning methods (the solutions obtained by the proposed method are deliberately moved towards the origin for clarity).

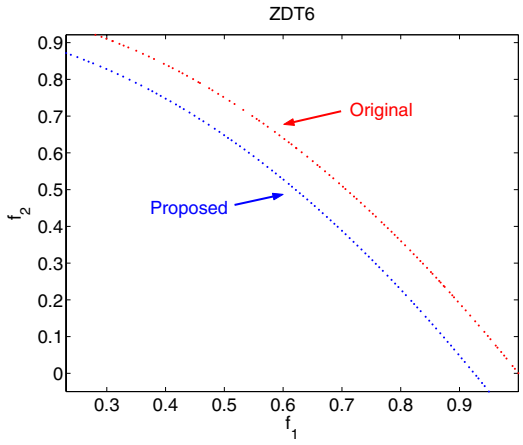


Fig. 8. A result for ZDT6 using the original and proposed pruning methods (the solutions obtained by the proposed method are deliberately moved towards the origin for clarity).

Table I also contains corresponding results for SPEA2[1]. Interestingly, it can be noticed that according to measures, the diversity obtained with SPEA2 is between the original and proposed pruning methods.

The time complexity of the pruning methods was verified experimentally using the ZDT1 problem. The measured pruning times for various population sizes are shown in Figure 9. It can be observed that the proposed pruning method takes about twice the time of the original method and the complexity classes of these methods are the same, whereas the pruning time in SPEA2 is much more and increases drastically when the population size increases.

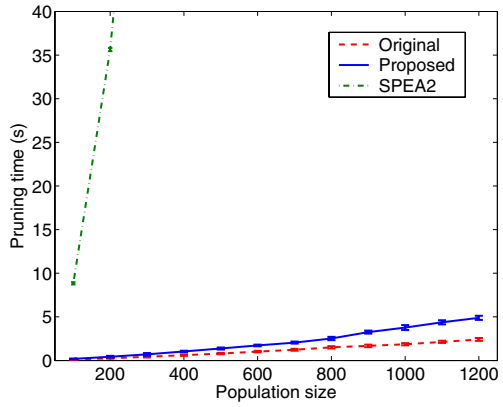


Fig. 9. Mean pruning times (standard deviations as error bars) measured from 100 runs for solving ZDT1 problem with different population sizes.

B. Tri-Objective Problems

The proposed pruning method was also tested with a set of tri-objective test problems, DTLZ1, DTLZ2, DTLZ4, DTLZ5, and DTLZ7 [22], using population size 300 and 1000 generations. The numerical results from 100 repetition runs are shown in Table II. From these results it was noticed that there was no significant improvement in the values of spacing although the results with the proposed pruning method were better (except for DTLZ7) in term of variance of *CD*s. SPEA2 provided the best distribution according to the spacing measure but worst according to the variance of *CD*s. Moreover, there was no significant difference between the original and proposed methods when results were evaluated visually, whereas SPEA2 provided much better results (Figures 10–15). The only exception was DTLZ5, where the Pareto-front is curved in the objective space; in this case the proposed pruning method outperformed the original method and SPEA2. Bad observed performance of the proposed pruning method was unexpected and led to closer analysis of *CD* in the case of three objectives.

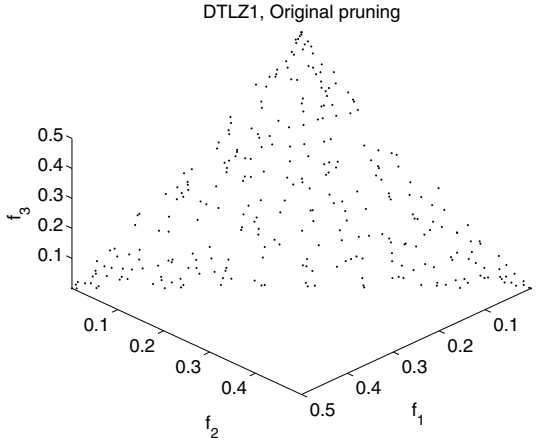


Fig. 10. A set of solutions for DTLZ1 using the original pruning method.

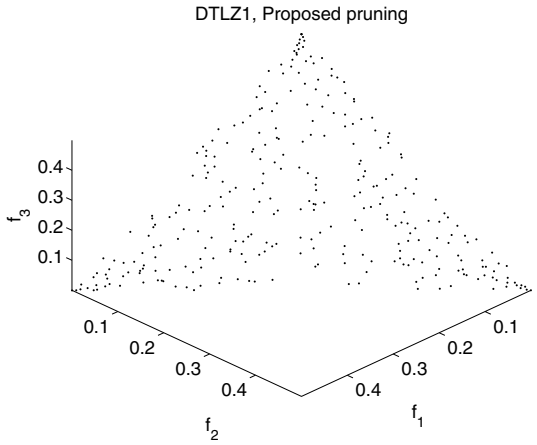


Fig. 11. A set of solutions for DTLZ1 using the proposed pruning method.

In Figures 16 and 17 sets of non-dominated solutions to be pruned are shown for the DTLZ1 and DTLZ2 problems. Both Figures also show three randomly selected solutions with their neighbors along each objective. It can be noticed that these neighbors do not necessarily locate near by corresponding solutions and therefore the calculated *CD*s do not

[1]SPEA2 implementation was taken from the PISA web site: http://www.tik.ee.ethz.ch/pisa/

TABLE I

MEAN AND STANDARD DEVIATION VALUES OF SPACING, VARIANCE OF *CD*S, AND CPU TIMES FOR ZDT TEST PROBLEMS MEASURED FROM 100 INDEPENDENT RUNS

Problem	Method	Spacing	$\sigma^2(CD)$	Total time (s)	Pruning time (s)
ZDT1	Original	$6.4240e-03 \pm 5.5946e-04$	$1.2755e-04 \pm 2.5046e-05$	$\mathbf{1.4877e+00} \pm \mathbf{1.3246e-02}$	$\mathbf{1.0530e-01} \pm \mathbf{3.1669e-02}$
	Proposed	$\mathbf{2.5348e-03} \pm \mathbf{2.6001e-04}$	$\mathbf{2.1163e-05} \pm \mathbf{3.6732e-06}$	$1.5940e+00 \pm 1.5699e-02$	$1.8720e-01 \pm 4.1368e-02$
	SPEA2	$3.2063e-03 \pm 2.9122e-04$	$4.2059e-05 \pm 8.0237e-06$	$1.1756e+01 \pm 7.0249e-02$	$7.6469e+00 \pm 1.9474e-01$
ZDT2	Original	$6.3904e-03 \pm 5.4882e-04$	$1.3523e-04 \pm 2.8336e-05$	$\mathbf{1.4718e+00} \pm \mathbf{6.8048e-03}$	$\mathbf{1.0521e-01} \pm \mathbf{3.1254e-02}$
	Proposed	$\mathbf{2.5981e-03} \pm \mathbf{2.8669e-04}$	$\mathbf{2.1138e-05} \pm \mathbf{3.7108e-06}$	$1.5747e+00 \pm 6.6290e-03$	$1.8227e-01 \pm 3.5693e-02$
	SPEA2	$3.4023e-03 \pm 3.6818e-04$	$4.9467e-05 \pm 1.0613e-05$	$1.2291e+01 \pm 9.7998e-02$	$8.1818e+00 \pm 2.2531e-01$
ZDT3	Original	$4.3573e-03 \pm 4.1398e-04$	$2.0291e-03 \pm 5.1891e-05$	$\mathbf{1.3838e+00} \pm \mathbf{5.4643e-03}$	$\mathbf{8.5400e-02} \pm \mathbf{2.9074e-02}$
	Proposed	$\mathbf{1.5039e-03} \pm \mathbf{1.8103e-04}$	$\mathbf{2.0169e-03} \pm \mathbf{2.0486e-05}$	$1.4421e+00 \pm 6.7112e-03$	$1.2820e-01 \pm 3.1667e-02$
	SPEA2	$3.0892e-03 \pm 3.4177e-04$	$2.0962e-03 \pm 3.0930e-05$	$1.1500e+01 \pm 7.2117e-02$	$7.3613e+00 \pm 2.2574e-01$
ZDT4	Original	$6.0789e-03 \pm 5.5549e-04$	$1.0398e-04 \pm 2.1932e-05$	$\mathbf{9.5180e-01} \pm \mathbf{5.7525e-03}$	$\mathbf{8.6200e-02} \pm \mathbf{2.6772e-02}$
	Proposed	$\mathbf{2.1344e-03} \pm \mathbf{3.1260e-04}$	$\mathbf{1.4066e-05} \pm \mathbf{3.6445e-06}$	$9.8360e-01 \pm 6.8931e-03$	$1.2440e-01 \pm 2.6980e-02$
	SPEA2	$2.8862e-03 \pm 2.9266e-04$	$3.2085e-05 \pm 6.8718e-06$	$1.0639e+01 \pm 9.3916e-02$	$6.3440e+00 \pm 2.8208e-01$
ZDT6	Original	$6.6047e-03 \pm 6.5536e-04$	$1.3420e-04 \pm 2.8494e-05$	$\mathbf{1.2147e+00} \pm \mathbf{5.7753e-03}$	$\mathbf{9.6869e-02} \pm \mathbf{3.4866e-02}$
	Proposed	$\mathbf{2.6662e-03} \pm \mathbf{2.6590e-04}$	$\mathbf{2.2287e-05} \pm \mathbf{3.5135e-06}$	$1.3088e+00 \pm 6.7090e-03$	$1.7740e-01 \pm 4.2035e-02$
	SPEA2	$3.1010e-03 \pm 3.0937e-04$	$3.8489e-05 \pm 8.6676e-06$	$1.1662e+01 \pm 5.7654e-02$	$7.5451e+00 \pm 2.0404e-01$

properly estimate the crowdedness of solutions. Presumably, the same observation is extendable for cases with more than three objectives. *CD* gives relatively good estimation about crowdedness in the bi-objectives cases because the non-dominance property (which implies a monotonic relation between solutions in the objective space) causes solutions to come close together along both the objectives. With three or more objectives this does not need to hold at all, even though it holds with DTLZ5 where the non-dominating solutions form a curve instead of a surface.

For comparison, the original pruning method was also implemented in such a way that random values for *CD*s were used instead of calculated values. For each non-dominated set member having maximum or minimum value for any objective, *CD* was assigned an infinite value. The numerical results for DTLZ problems using this random pruning approach are also shown in Table II. According to the results, the random pruning performs notably worse than the pruning method based on the calculated *CD* values. Therefore *CD* contains some knowledge about crowdedness although the estimation is far from perfect based on Figures 16 and 17.

The proposed method did not improve diversity in the DTLZ7 problem according to the variance of *CD*s, and in the ZDT3 problem the improvement according to *CD* was modest. The probable reason for this is the fact that the Pareto-front for these problems is discontinuous in contrast to the other problems. Visually observed, the diversity was also improved for ZDT3.

VI. DISCUSSION

According to observations, the diversity handling method of NSGA-II should be modified if a good diversity is desired for problems having more than two objectives. Worse obtained diversity with *CD* than with a diversity maintenance technique based on clustering, has already been observed earlier in the case of optimization problems with three objectives [22] but the reason for the bad performance has not been studied or demonstrated earlier according to authors' knowledge. These observations mean that also other methods

TABLE II

MEAN AND STANDARD DEVIATION VALUES OF SPACING, VARIANCE OF *CD*S, AND CPU TIMES FOR DTLZ TEST PROBLEMS MEASURED FROM 100 INDEPENDENT RUNS

Problem	Method	Spacing	$\sigma^2(CD)$	Total time (s)	Pruning time (s)
DTLZ1	Random	$3.0980e-02 \pm 3.7708e-03$	$2.8078e-04 \pm 1.3032e-04$	$1.4591e+01 \pm 1.5585e-01$	$6.4080e-01 \pm 8.4539e-02$
	Original	$2.3927e-02 \pm 1.1410e-03$	$2.5804e-05 \pm 2.8357e-06$	$\mathbf{1.3648e+01 \pm 1.1168e-01}$	$\mathbf{6.2460e-01 \pm 6.3444e-02}$
	Proposed	$2.3549e-02 \pm 1.0009e-03$	$\mathbf{9.7780e-06 \pm 9.0294e-07}$	$1.4872e+01 \pm 9.3362e-02$	$1.3115e+00 \pm 8.0634e-02$
	SPEA2	$\mathbf{1.1661e-02 \pm 1.1200e-02}$	$3.7368e-04 \pm 8.7105e-04$	$1.2211e+02 \pm 1.2640e-01$	$7.6797e+01 \pm 1.7521e-01$
DTLZ2	Random	$3.8560e-02 \pm 3.5166e-03$	$2.1920e-04 \pm 8.9905e-05$	$1.4465e+01 \pm 1.3946e-01$	$6.7530e-01 \pm 7.5324e-02$
	Original	$2.8914e-02 \pm 1.3827e-03$	$2.1855e-05 \pm 1.9725e-06$	$\mathbf{1.3536e+01 \pm 3.3132e-02}$	$\mathbf{6.4540e-01 \pm 7.4947e-02}$
	Proposed	$2.8014e-02 \pm 1.5111e-03$	$\mathbf{1.0638e-05 \pm 8.4747e-07}$	$1.4831e+01 \pm 4.8988e-02$	$1.3456e+00 \pm 1.4674e-01$
	SPEA2	$\mathbf{1.2919e-02 \pm 6.9397e-04}$	$8.1446e-05 \pm 1.0243e-05$	$1.3095e+02 \pm 7.7438e-02$	$8.7535e+01 \pm 1.5309e-01$
DTLZ4	Random	$4.6069e-02 \pm 1.0088e-02$	$1.7806e-03 \pm 4.7662e-04$	$\mathbf{1.3135e+01 \pm 4.1494e-01}$	$\mathbf{6.3520e-01 \pm 7.2995e-02}$
	Original	$2.8452e-02 \pm 1.2664e-03$	$2.2104e-05 \pm 2.0978e-06$	$1.3813e+01 \pm 5.8934e-02$	$6.4060e-01 \pm 6.9614e-02$
	Proposed	$2.7879e-02 \pm 1.3367e-03$	$\mathbf{1.0864e-05 \pm 7.9148e-07}$	$1.5101e+01 \pm 6.2203e-02$	$1.3591e+00 \pm 1.2607e-01$
	SPEA2	$\mathbf{1.3022e-02 \pm 7.2771e-04}$	$9.2581e-05 \pm 1.1494e-05$	$1.3124e+02 \pm 1.0552e-01$	$8.7665e+01 \pm 1.7151e-01$
DTLZ5	Random	$7.1387e-03 \pm 2.6948e-03$	$5.1590e-04 \pm 2.9150e-04$	$1.3403e+01 \pm 1.3308e-01$	$6.2730e-01 \pm 7.1941e-02$
	Original	$3.5364e-03 \pm 1.9787e-04$	$3.8692e-05 \pm 4.6751e-06$	$\mathbf{1.2149e+01 \pm 8.9120e-02}$	$\mathbf{5.7600e-01 \pm 6.6999e-02}$
	Proposed	$\mathbf{1.6077e-03 \pm 1.5687e-04}$	$\mathbf{5.8746e-06 \pm 9.1744e-07}$	$1.3575e+01 \pm 7.2202e-02$	$1.1437e+00 \pm 7.8130e-02$
	SPEA2	$2.3039e-03 \pm 1.1166e-04$	$2.4633e-05 \pm 2.5317e-06$	$1.2363e+02 \pm 8.3786e-02$	$7.7715e+01 \pm 1.2489e-01$
DTLZ7	Random	$4.1365e-02 \pm 1.8823e-02$	$4.6935e-03 \pm 1.3886e-03$	$1.3011e+01 \pm 1.3106e-01$	$6.3580e-01 \pm 7.4633e-02$
	Original	$2.3658e-02 \pm 2.6283e-03$	$\mathbf{1.5495e-03 \pm 2.1474e-04}$	$\mathbf{1.2066e+01 \pm 2.2497e-02}$	$\mathbf{5.5920e-01 \pm 6.4834e-02}$
	Proposed	$2.3932e-02 \pm 2.3022e-03$	$1.5639e-03 \pm 1.9189e-04$	$1.3029e+01 \pm 4.1459e-02$	$1.0860e+00 \pm 8.3545e-02$
	SPEA2	$\mathbf{1.5140e-02 \pm 8.2394e-04}$	$1.9248e-03 \pm 1.0965e-04$	$1.3342e+02 \pm 6.7649e-02$	$8.8697e+01 \pm 1.0649e-01$

(*e.g.*, those mentioned in Section I) applying *CD* or its modification and using neighbors along each objectives to estimate crowdedness, do not provide good distribution when the number of objectives is larger than two. Use of conventional clustering techniques lead to high time complexity such as in SPEA2.

This leads to conclusion that a new pruning method should be developed, particularly for many (> 2) objective problems. One way to do this would be to use an efficient nearest neighbor method to compute a distance metric with a few (probably two) nearest solutions (in the Euclidean sense) of every member of a non-dominated set in the normalized objective space. Thereafter, solutions having a larger distance metric can be preferred to maintain diversity in the non-dominated set. The extreme solutions can be preserved in the same manner as in the pruning algorithms based on *CD*, and the pruning technique should take care of updating crowdedness values for remaining members of the non-

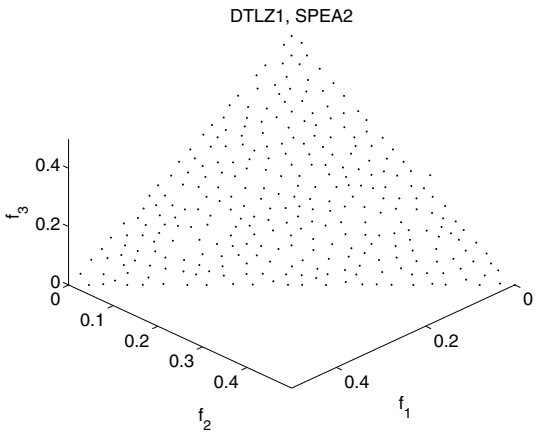


Fig. 12. A set of solutions for DTLZ1 using SPEA2.

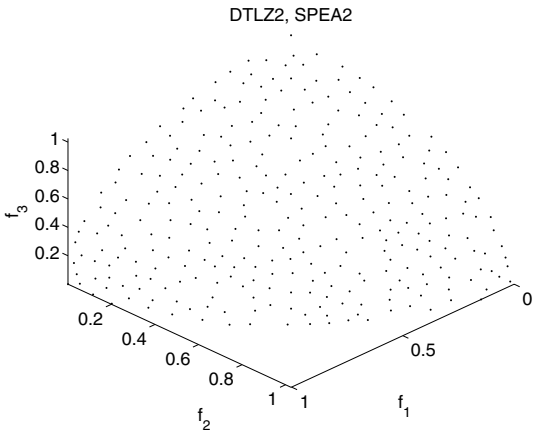


Fig. 15. A set of solutions for DTLZ2 using SPEA2.

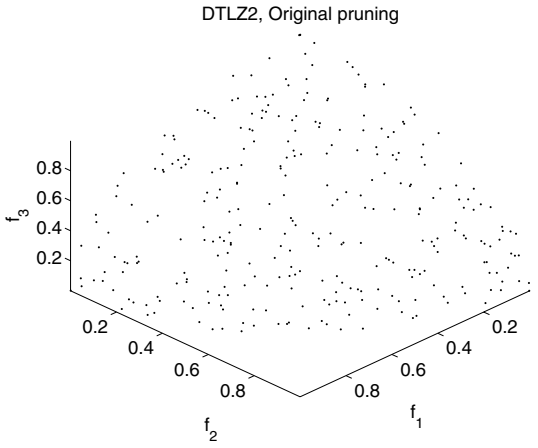


Fig. 13. A set of solutions for DTLZ2 using the original pruning method.

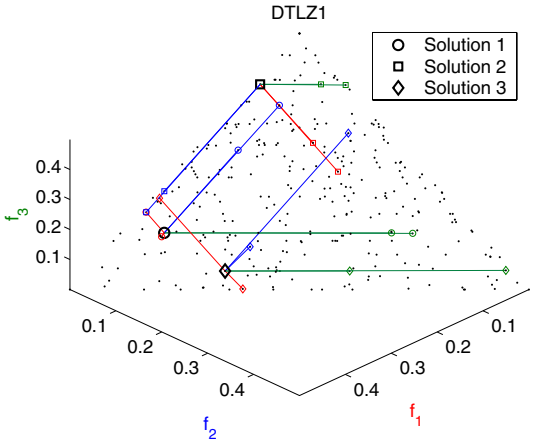


Fig. 16. A set of solutions to be pruned for DTLZ1 and three solutions with their neighbors according to individual objectives.

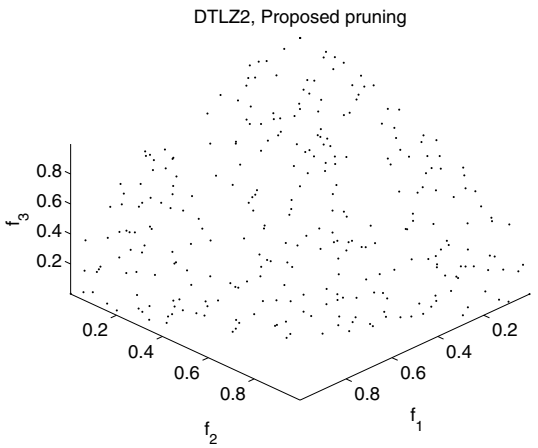


Fig. 14. A set of solutions for DTLZ2 using the proposed pruning method.

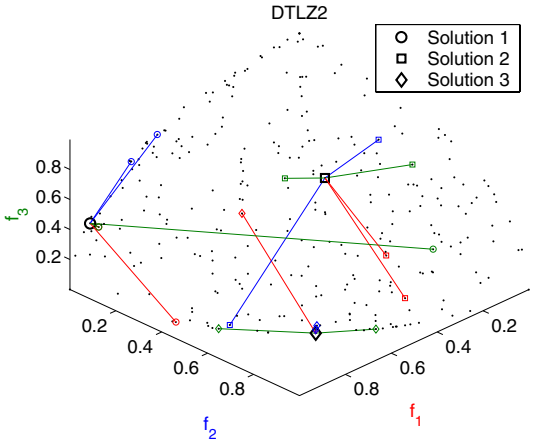


Fig. 17. A set of solutions to be pruned for DTLZ2 and three solutions with their neighbors according to individual objectives.

dominated set after each extraction of the most crowded member. Other ideas are also possible, but this paper has indicated the need for such studies, particularly for large number of objectives.

Better pruning method is not only likely to provide a better distribution, but may also help in a faster convergence because better distribution helps to observe the characteristics of the objective function space better.

VII. Conclusions

In this paper an algorithm improving pruning of non-dominated set of solutions is proposed. The algorithm is based on crowding distance and it is demonstrated to provide better distribution for the pruned set in bi-objective cases than the original algorithm used in NSGA-II.

Time complexity for the new algorithm is estimated and it is bounded by $O\left(MN\log N\right)$ or $O\left(M^2N\right)$, whichever is greater. In practice N is often much larger than M leading to the same complexity class $O\left(MN\log N\right)$ as the original pruning algorithm.

The original and proposed pruning methods were implemented and their time complexity classes were observed to be the same. The methods were compared using test problems and diversity metrics. According to the results, the improvement for the obtained diversity is considerable in bi-objective cases. However, there is no improvement in most tri-objective cases because the crowding distance metric does not estimate crowdedness properly in these cases.

The development of an efficient pruning method for problems with more than two objectives will be a subject of future research.

Acknowledgments

Authors wish to thank Dr. M. Laumanns and Mr. S. Bleuler for their help with SPEA2. First author gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), Technological Foundation of Finland, Finnish Cultural Foundation, and Centre for International Mobility (CIMO). This work was done during the visit of first author to KanGAL under the Indo-Finnish exchange student programme. First author also wishes to thank KanGAL researchers for their help and fruitful discussions.

References

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of the Third Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2001)*, Athens, Greece, Sept 2002, pp. 95–100.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[3] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, Oct 2003.

[4] N. K. Madavan, "Multiobjective optimization using a Pareto Differential Evolution approach," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, May 2002, pp. 1145–1150.

[5] S. Kukkonen and J. Lampinen, "An extension of Generalized Differential Evolution for multi-objective optimization with constraints," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 752–761.

[6] K. Maneeratana, K. Boonlong, and N. Chaiyaratana, "Multi-objective optimisation by co-operative co-evolution," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 772–781.

[7] X. Zou, M. Liu, L. Kang, and J. He, "A high performance multi-objective evolutionary algorithm based on the principles of thermodynamics," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 922–931.

[8] M. R. Sierra and C. A. Coello Coello, "Improving pso-based multi-objective optimization using crowding, mutation and e-dominance," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Guanajuato, Mexico, March 2005, pp. 505–519.

[9] T. Robič and B. Filipič, "DEMO: Differential Evolution for multi-objective optimization," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Guanajuato, Mexico, March 2005, pp. 520–533.

[10] H. E. Aguirre and K. Tanaka, "Selection, drift, recombination, and mutation in multiobjective evolutionary algorithms on scalable mnk-landscapes," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Guanajuato, Mexico, March 2005, pp. 355–369.

[11] C. R. Raquel and P. C. Naval Jr., "An effective use of crowding distance in multiobjective particle swarm optimization," in *Proceedings of the Genetic and Evolutionary Computation (GECCO 2005)*, Washington, DC, USA, June 2005, pp. 257–264.

[12] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of Generalized Differential Evolution," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 443–450.

[13] K. Sastry, M. Pelikan, and D. E. Goldberg, "Limits of scalability of multiobjective estimation of distribution algorithms," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 2217–2224.

[14] G. K. M. Pedersen and D. E. Goldberg, "Dynamic uniform scaling for multiobjective genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation (GECCO 2004)*, Seattle, WA, USA, June 2004, pp. 11–23.

[15] H. Sato, H. Aguirre, and K. Tanaka, "On the locality of dominance and recombination in multiobjective evolutionary algorithms," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 451–466.

[16] B. de la Iglesia, A. Reynolds, and V. J. Rayward-Smith, "Developments on a multi-objective metaheuristic (MOMH) algorithm for finding interesting sets of classification rules," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Guanajuato, Mexico, March 2005, pp. 826–840.

[17] N. Hallam, P. Blanchfield, and G. Kendall, "Handling diversity in evolutionary multiobjective optimisation," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 2233–2240.

[18] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2001.

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Prentice-Hall, 1990.

[20] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer-Verlag, 2005.

[21] A. Inorio and X. Li, "Solving rotated multi-objective optimization problems using Differential Evolution," in *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004)*, Cairns, Australia, Dec 2004, pp. 861–872.

[22] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Evolutionary Multi-objective Optimization*. London: Springer-Verlag, 2005, ch. Scalable Test Problems for Evolutionary Multiobjective Optimization, pp. 105–145.

Publication VI

Kukkonen, S., Lampinen, J.,
An Empirical Study of Control Parameters for The Third Version of Generalized Differential Evolution (GDE3)

IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada, July, 2006, pages 7355–7362.

2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

An Empirical Study of Control Parameters for The Third Version of Generalized Differential Evolution (GDE3)

Saku Kukkonen and Jouni Lampinen

Abstract— **In this paper the influence of control parameters to the search process of previously introduced Generalized Differential Evolution 3 (GDE3) is empirically studied. GDE3 is an extension of Differential Evolution (DE) for constrained multi-objective optimization. Besides the number of generations and the size of the population, GDE3 has two control parameters, CR and F, which have to be set before applying the method for solving a problem. The effect of these two control parameters is studied with a set of bi-objective test problems and performance metrics.**

The objectives of this study are to investigate the sensitivity of the control parameters according to different performance metrics, to understand better the effect and tuning of the control parameters on the search process, and to compare results with the corresponding results of the first version of GDE.

A similar non-linear relationship between the control parameters and performance metrics was observed as in earlier studies with the first version of GDE. According to the relationship a larger F value can be used with a small CR value than with a large CR value.

I. Introduction

Many practical problems have multiple objectives and several aspects cause constraints to problems. Generalized Differential Evolution (GDE) [1]–[3] is an extension of Differential Evolution (DE) [4], [5] for constrained multi-objective (Pareto-)optimization. DE is a relatively new real-coded Evolutionary Algorithm (EA), which has been demonstrated to be efficient for solving many real-world problems.

In the earlier study [6], the effect of control parameters has been studied using the first version of GDE and a set of bi-objective test problems. The first version of GDE [1] has no explicit non-dominated sorting or diversity maintenance mechanism. This paper continues the earlier study by repeating the same tests for the latest version of GDE, known as GDE3 [3], which includes non-dominated sorting and diversity maintenance. Results are presented the same way as earlier and a comparison of results between the two GDE versions is also included.

This paper continues with the following parts: Section II describes the Differential Evolution algorithm and Section III describes its extension, Generalized Differential Evolution. Section IV describes experiments and finally conclusions are given in Section V.

II. Differential Evolution

The DE algorithm [4], [5] was introduced by Storn and Price in 1995. The design principles of DE are simplicity, efficiency, and the use of floating-point encoding instead of binary numbers. As a typical EA, DE has a random initial population that is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides an appropriate stopping condition. Other control parameters for DE are the crossover control parameter CR, the mutation factor F, and the population size NP.

In each generation G, DE goes through each D dimensional decision vector $\vec{x}_{i,G}$ of the population and creates the corresponding trial vector $\vec{u}_{i,G}$ as follows [7]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, \text{(randomly selected,}\\
&\quad \text{except mutually different and different from } i)\\
&j_{rand} = \text{floor}\left(rand_i[0,1) \cdot D\right) + 1\\
&\text{for}(j = 1; j \leq D; j = j + 1)\\
&\{\\
&\quad \text{if}(rand_j[0,1) < CR \vee j = j_{rand})\\
&\quad\quad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G})\\
&\quad \text{else}\\
&\quad\quad u_{j,i,G} = x_{j,i,G}\\
&\}
\end{aligned}
$$

This is the most common DE version, DE/rand/1/bin. Both CR and F remain fixed during the entire execution of the algorithm. Parameter $CR \in [0, 1]$, which controls the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors and not from the old vector $\vec{x}_{i,G}$. The condition "$j = j_{rand}$" is to make sure that at least one element is different compared to the elements of the old vector. The parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$. In practice, CR controls the rotational invariance of the search, and its small value (*e.g.*, 0.1) is practicable with separable problems while larger values (*e.g.*, 0.9) are for non-separable problems. The control parameter F controls the speed and robustness of the search, *i.e.*, a lower value for F increases the convergence rate but also the risk of getting stuck into a local optimum. Parameters CR and NP have the same kind of effect on the convergence rate as F has.

After the mutation and crossover operations, the trial vector $\vec{u}_{i,G}$ is compared to the old vector $\vec{x}_{i,G}$. If the trial vector has an equal or better objective value, then it replaces the old vector in the next generation. DE is an elitist method since the best population member is always preserved and the average objective value of the population will never get worse.

The authors are with the Department of Information Technology, Lappeenranta University of Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland; email: saku.kukkonen@lut.fi.

0-7803-9487-9/06/$20.00/©2006 IEEE

III. Generalized Differential Evolution

The first version of a Generalized Differential Evolution (GDE) extended DE for constrained multi-objective optimization, and it modified only the selection rule of the basic DE [1]. The basic idea in the selection rule of GDE is that the trial vector is selected to replace the old vector in the next generation if it weakly constraint-dominates the old vector. This means that the trial vector is required to dominate[1] compared old population member in the constraint violation space or in the objective function space, or at least provide an equally good solution as the old population member. There was no sorting of non-dominated solutions during the optimization process or any mechanism for maintaining the distribution and extent of solutions. Also, there was no extra repository for non-dominated solutions.

The second version, GDE2, made a decision based on the crowdedness when the trial and old vector were feasible and non-dominating each other in the objective function space [2]. This improved the extent and distribution of the obtained set of solutions but slowed down the convergence of the overall population because it favored isolated solutions far from the Pareto-front until all the solutions were converged near the Pareto-front.

The third and latest version is GDE3 [3]. Besides the selection, another part of the basic DE has also been modified. Now, in the case of feasible and non-dominating solutions, both vectors are saved for the population of next generation. Before starting the next generation, the size of the population is reduced using non-dominated sorting and pruning based on diversity preservation.

In the earlier study [3], GDE3 was noted to obtain better distribution compared to earlier GDE versions. However, the performance was not thoroughly tested with commonly adopted metrics for MOEAs by applying different control parameter combinations.

IV. Experiments

GDE3 was tested with the same set of bi-objective benchmark problems as in [6]. These problems are known as SCH1, SCH2, FON, KUR, POL, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [8, pp. 338–360] and they all have two objectives to be minimized. The first two problems have one decision variable whereas the rest of the problems have multiple variables. A more detailed characterization of the problems can be found in [6] and [8, pp. 338–360]. Since almost all the aforementioned problems with multiple variables are variable-wise separable, new non-separable modifications of ZDT problems were also added to the test set. These problems are described in [9] and are called as L_3-ZDT1, L_3-ZDT2, L_3-ZDT3, L_3-ZDT4, and L_3-ZDT6.

In order to bound the number of results to fit into one paper, many-objective problems (*i.e.*, problems with more than two objectives) were not included to test set. Also, the diversity maintenance technique of GDE3 has been observed to perform badly for many-objective problems, and improving diversity maintenance is currently under research [10].

A population size of 100 was used in all tests. The number of generations for each problem was kept the same as in [6], and it was 30 for SCH1, 5 for SCH2, 150 for KUR, 30 for POL, and 250 for the FON, ZDT & L_3-ZDT problems. From the final population unique non-dominated solutions were extracted to form an approximation of the Pareto-front.

Different control parameter value combinations in the range $CR \in [0, 1]$ and $F \in [0, 3]$ with a resolution of 0.05 were tested for all the test problems. Tests were repeated 100 times with each control parameter combination and the results were evaluated using convergence and diversity metrics. Closeness to the Pareto-front was measured with a generational distance (GD) [8, pp. 326–327], which measures the average distance of the solutions from the Pareto-front. Diversity of the obtained set of solutions was measured using a spacing (S) metric [8, pp. 327–328], which measures the standard deviation of the distances from each solution to the nearest solution in the obtained non-dominated set. Smaller values for both metrics are preferable, and the optimal values are zero. The number of unique non-dominated solutions ($\aleph$) in the final population was also registered.

Results for the different problems are shown in Figs 1–15 as surfaces in the control parameter space. Similarly as for GDE in [6], the results for all the problems show that with a small F value the final population has the greatest amount of unique non-dominated solutions, and also the values for performance metrics GD and S are the best. For the multidimensional (*i.e.*, $D > 1$) test problems, values for the performance metrics are best with low CR values, whereas for the one-dimensional test problems, CR does not seem to have a notable impact on the result. Results with SCH1 show clearly how a large F value slows down convergence speed.

The value range for F in the tests is larger than the value usually used for single-objective optimization. The results confirm the earlier observation in [6] that a larger F value does not give any extra benefit in the case of multi-objective optimization, and therefore its value can be chosen from the same value range as in the case of single-objective optimization.

From the results of the multidimensional test problems, one can observe a non-linear relationship between CR and F values, *i.e.*, a larger F value can be used with a small CR value than with a large CR value, and this relationship is non-linear. This same phenomena was observed earlier also with GDE [6]. An explanation for this was found from a theory for the single-objective DE algorithm. A formula for the relationship between the control parameters of DE and the evolution/development of the population variance has been conducted in [11]. The change of the population variance between successive generations due to the crossover and mutation operations is denoted with c and its value is

[1] A solution (vector) x weakly dominates solution y in the function space F, if x has at least as good value for all the functions of F as y has. If x weakly dominates y and x has better value for at least one function of F than y has, then x dominates y. If x and y do not dominate each other, they belong to the same non-dominating set.

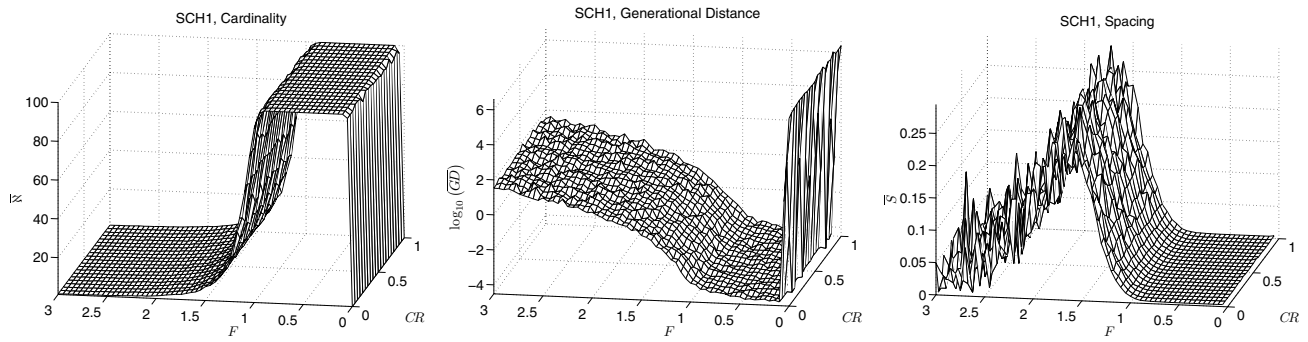


Fig. 1. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD, note logarithmic scale), and spacing (S) for SCH1 shown as surfaces in the control parameter space. Strange shape of the spacing surface is due to only a few non-dominated solutions when F is large.

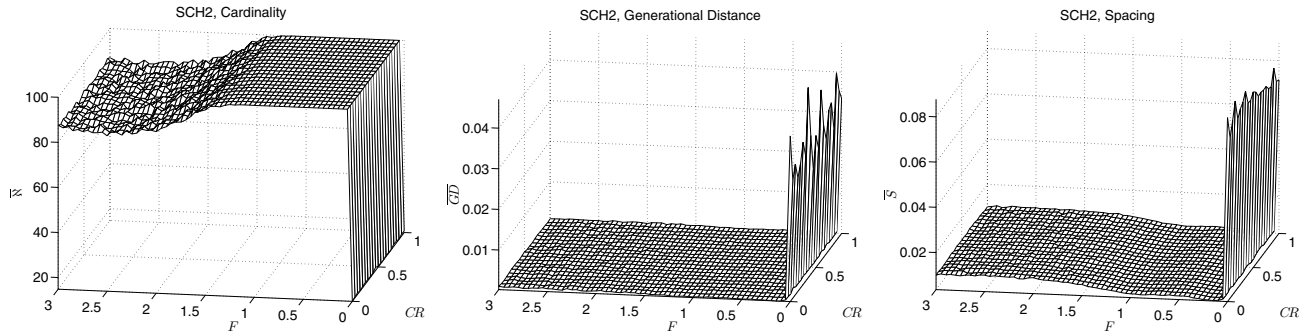


Fig. 2. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for SCH2 shown as surfaces in the control parameter space.

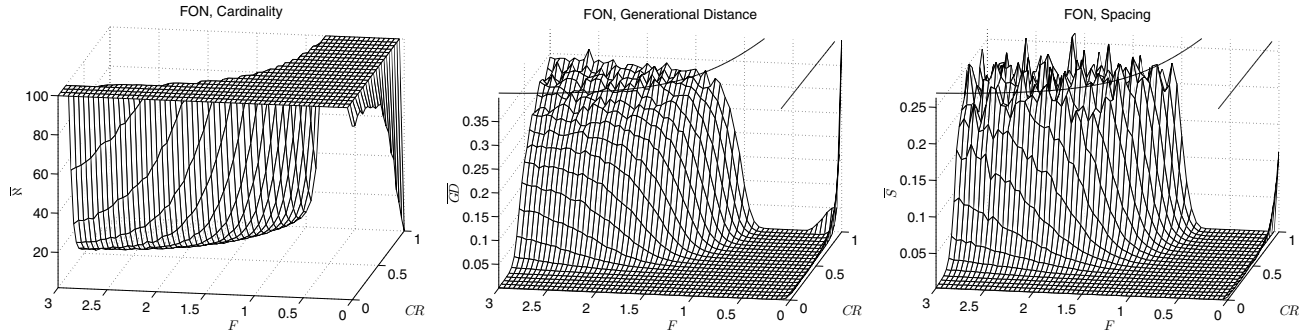


Fig. 3. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for FON shown as surfaces in the control parameter space.

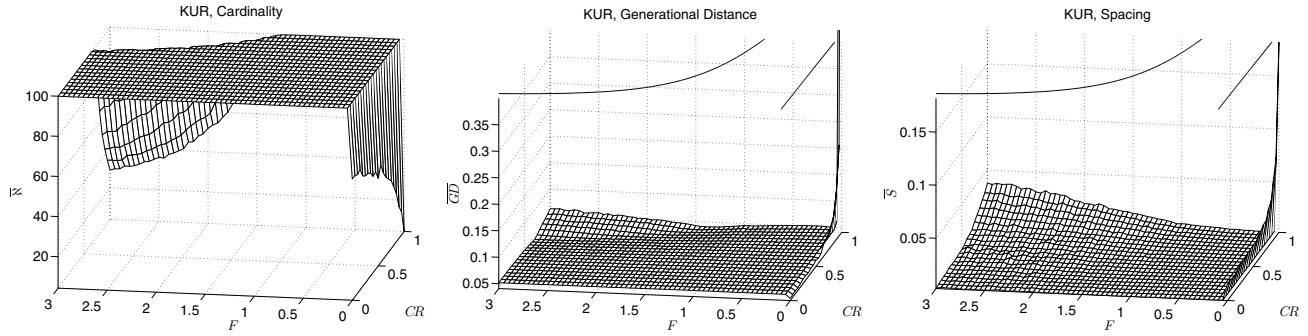


Fig. 4. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for KUR shown as surfaces in the control parameter space.

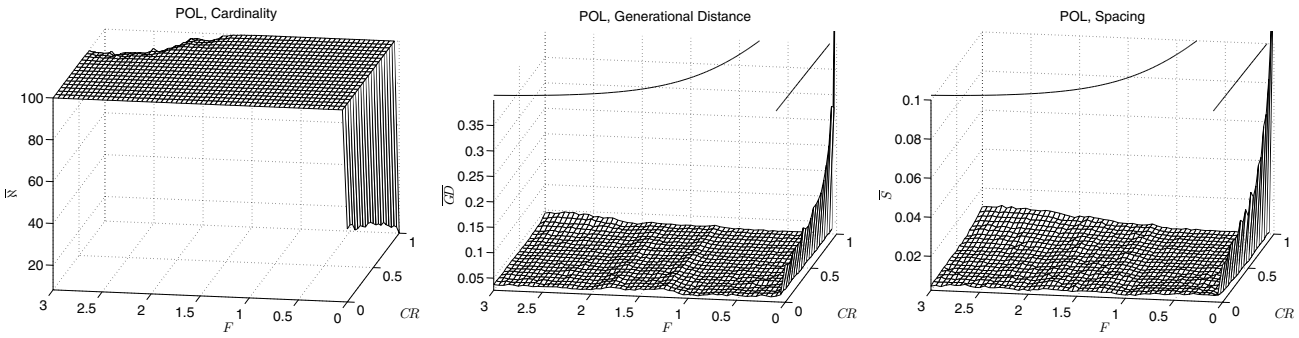


Fig. 5. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for POL shown as surfaces in the control parameter space.

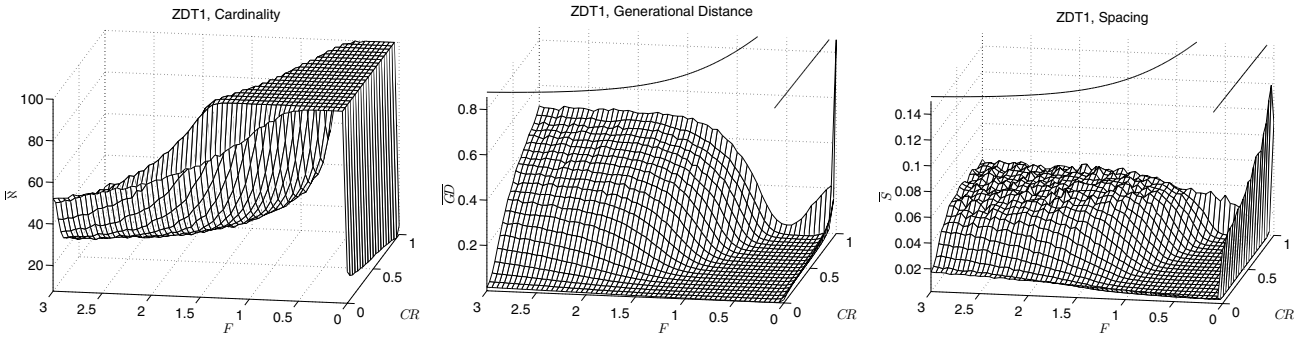


Fig. 6. Mean values of the number of unique non-dominated solutions ($\aleph$) in the the final population, generational distance (GD), and spacing (S) for ZDT1 shown as surfaces in the control parameter space.

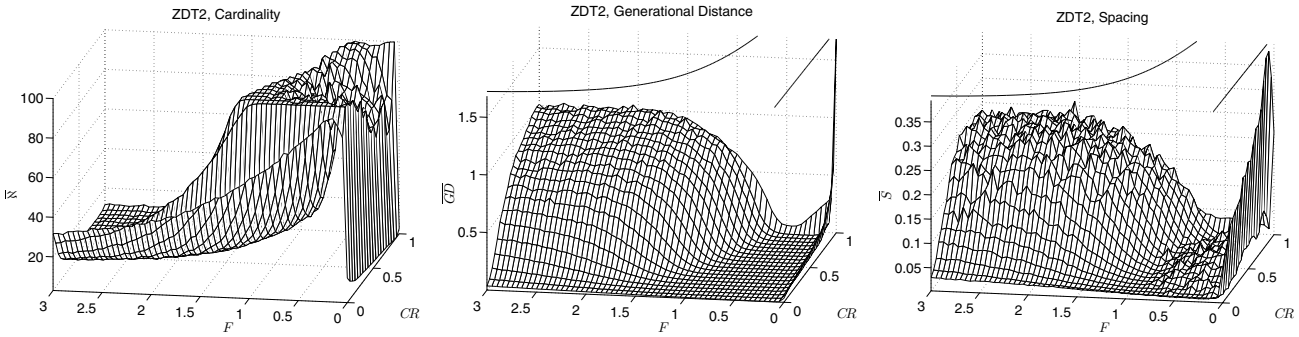


Fig. 7. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for ZDT2 shown as surfaces in the control parameter space.

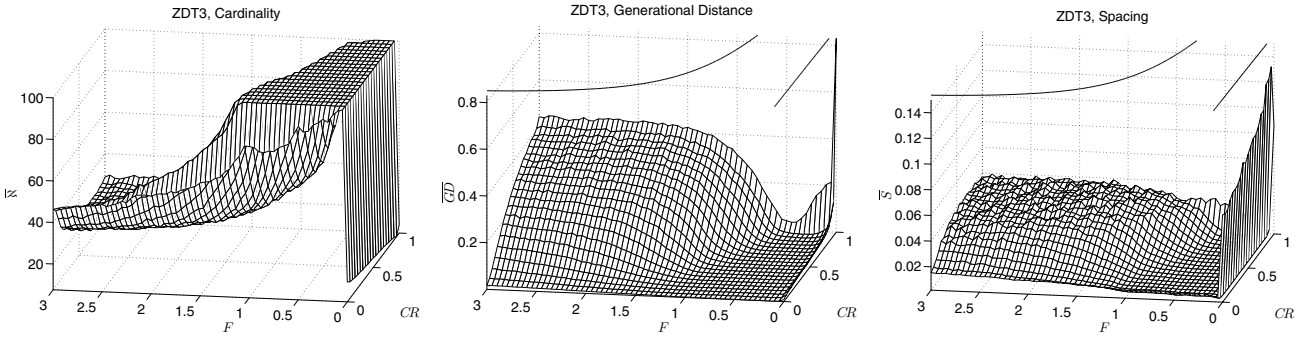


Fig. 8. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for ZDT3 shown as surfaces in the control parameter space.

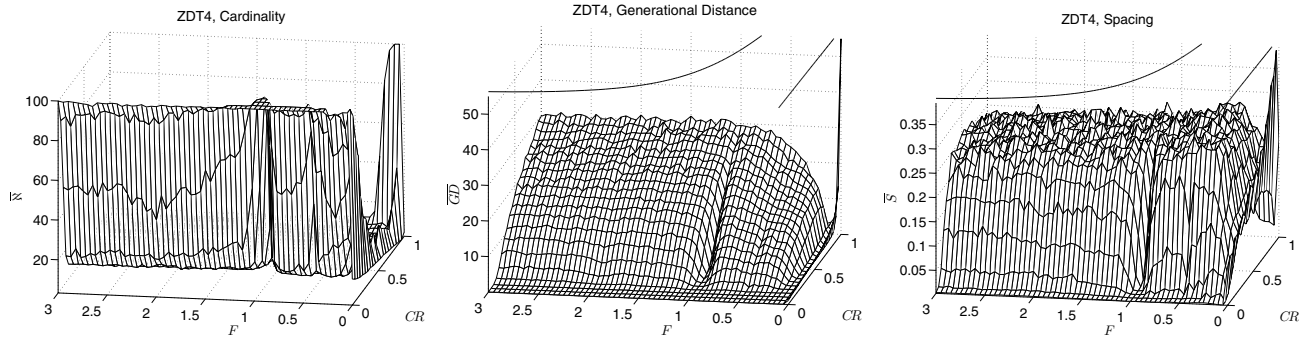


Fig. 9. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for ZDT4 shown as surfaces in the control parameter space.

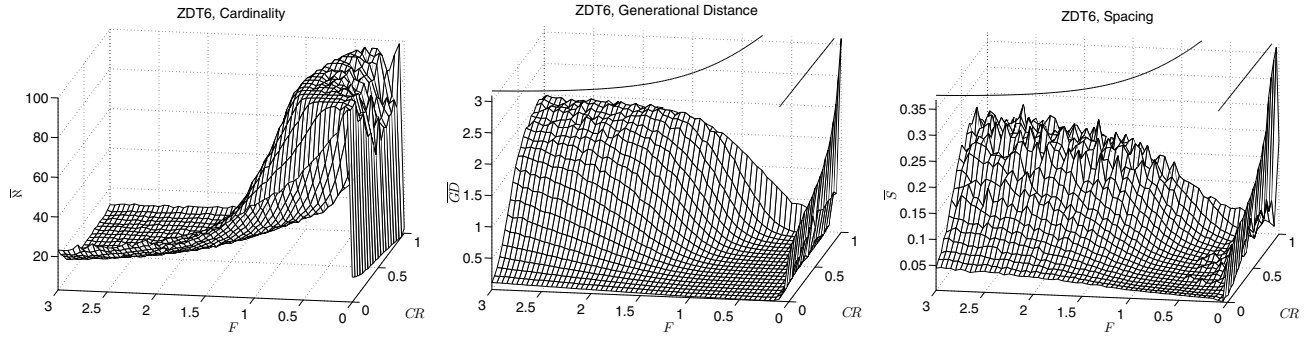


Fig. 10. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for ZDT6 shown as surfaces in the control parameter space.

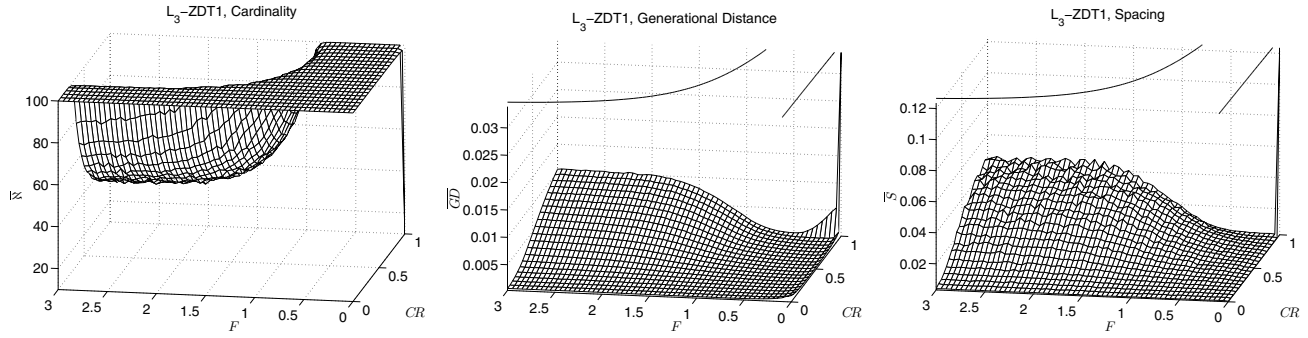


Fig. 11. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for L3-ZDT1 shown as surfaces in the control parameter space.

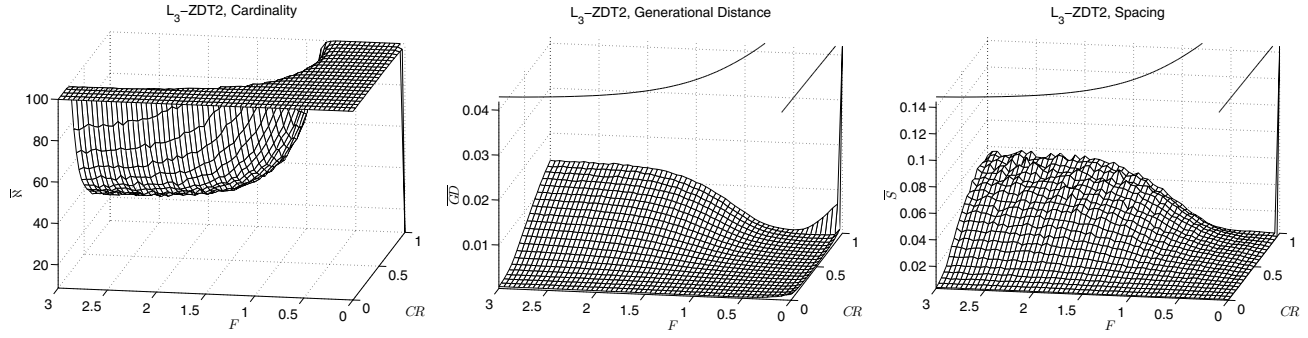


Fig. 12. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for L3-ZDT2 shown as surfaces in the control parameter space.

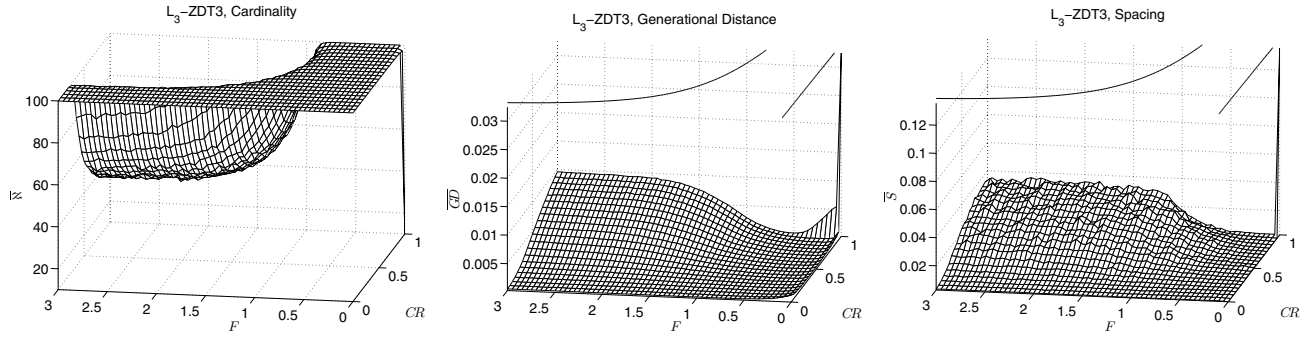


Fig. 13. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for L3-ZDT3 shown as surfaces in the control parameter space.

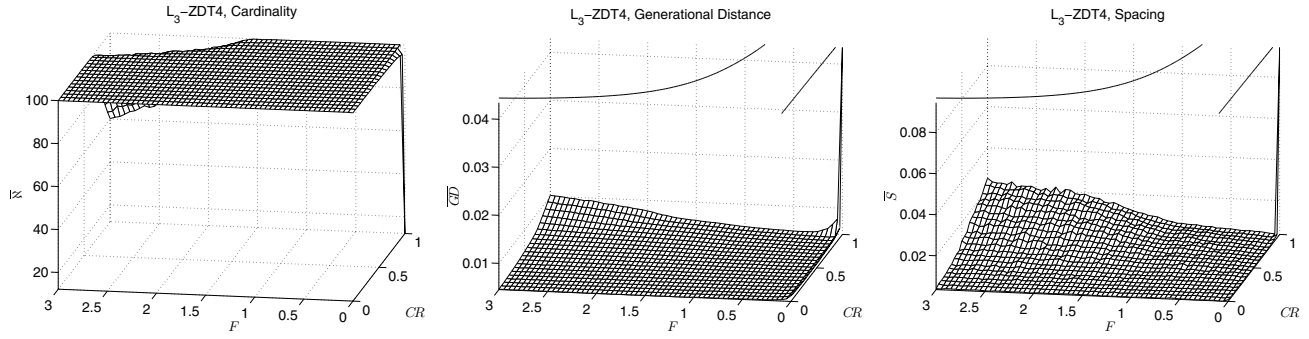


Fig. 14. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for L3-ZDT4 shown as surfaces in the control parameter space.

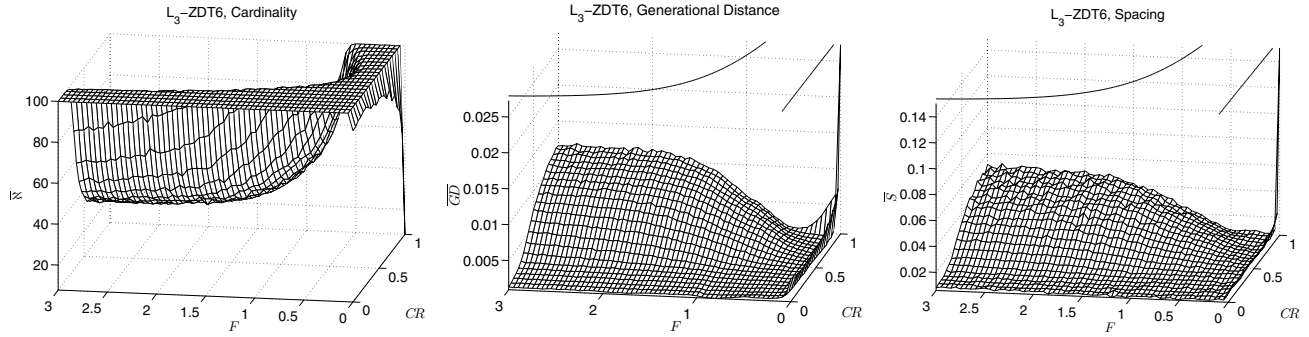


Fig. 15. Mean values of the number of unique non-dominated solutions ($\aleph$) in the final population, generational distance (GD), and spacing (S) for L3-ZDT6 shown as surfaces in the control parameter space.

calculated as $c = \sqrt{2F^2CR - 2CR/NP + CR^2/NP + 1}$. When $c < 1$, the crossover and mutation operations decrease the population variance. When $c = 1$, the variance do not change, and when $c > 1$, the variance increases. Since a selection operation of an EA usually decreases the population variance, $c > 1$ is recommended to prevent too high convergence rate with a poor search coverage, which typically results in a premature convergence. On the other hand, if c is too large, the search process proceeds reliably, but too slowly. In practice it has been observed that $c < 1.5$ is suitable upper limitation for most of the cases [5].

When the size of the population is relatively large (*e.g.*, $NP > 50$), the value of c depends mainly on the values of CR and F. Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$ are shown in Fig. 16. These curves are also shown with the GD and S surfaces in Figs 3–15, where it can be observed that the curves define quite well the area of good control parameter combinations according to the performance metrics. Some variation exists probably due to the different characteristics of the problems. Also, correspondence between the curve $c = 1.5$ and NP surfaces can be observed clearly. Especially, the NP surface for FON in Fig. 3 matches well with the curve $c = 1.5$ with large CR values. This is a coincidence because location of the NP surface "edge" depends on selected number of generations. This connection between c and NP was not observed so well earlier with GDE in [6].

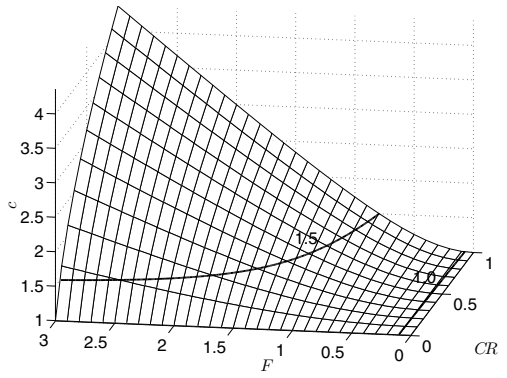


Fig. 16. Curves $c = 1.0$ and $c = 1.5$ for $NP = 100$.

These observations propose that the theory for the population variance between successive generations is applicable also in the case of multi-objective DE and provides a way to select good control parameter combinations. In order to show the correspondence between favorable c values and the performance metric values numerically, a correlation between performance metrics and absolute deflection of c from a value 1.25 was calculated. The hypothesis is that the performance declines when the corresponding c value differs from the mean of the given range $1.0 < c < 1.5$. This is illustrated in Fig. 17, where the c values and the corresponding GD values for ZDT1 are shown. This figure also justifies the recommended c range. The correlation coefficient values with corresponding significance levels for the test problems are presented in Table I. For FON and most of the ZDT & L3-ZDT problems a clear negative correlation between c and NP, and positive correlation between c and the other metrics can be observed. With the other problems there is less correlation. One should note that the recorded metric values depend on the number of generations used, and for some problems the number might not have been the best possible for observing the effect of the control variables.

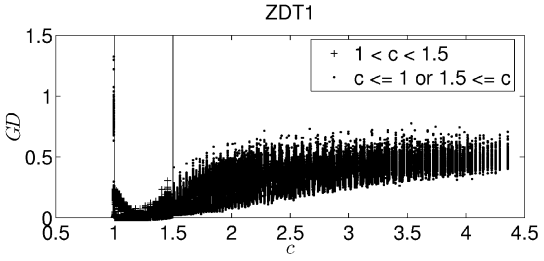


Fig. 17. Relation between the development of the population variance (c) and generational distance (GD) for ZDT1.

In general, the results are good with small CR and F values, which also means a low c value (close to one). One reason for the good performance with small control parameter values is that the test problems have conflicting objectives, which reduce overall selection pressure and prevent the premature convergence. Another reasons is that most of the problems have relatively easy objective functions to solve, leading to faster convergence with a small F, while a bigger F might be needed for harder functions with several local optima and/or if NP is smaller. With the L3-ZDT problems, larger control parameter values have been observed to provide a wider spread (see a web site `www.it.lut.fi/ip/evo` for additional results) and good results according to the hypervolume performance metric [9].

TABLE I

CORRELATION COEFFICIENTS (r) AND SIGNIFICANCE LEVELS (p) BETWEEN THE PERFORMANCE METRICS AND ABSOLUTE DEFLECTION OF c FROM A VALUE 1.25 FOR THE USED TEST PROBLEMS

$\lvert\mathbf{c} - \mathbf{1.25}\rvert$	$\aleph$		GD		S	
Problem	r	p	r	p	r	p
SCH1	-0.5797	0.00	-0.0333	0.00	0.1589	0.00
SCH2	-0.2796	0.00	-0.0269	0.00	0.0829	0.00
FON	-0.8272	0.00	0.8432	0.00	0.5807	0.00
KUR	-0.5483	0.00	-0.0368	0.00	0.4963	0.00
POL	-0.1781	0.00	-0.0034	0.22	0.0831	0.00
ZDT1	-0.6762	0.00	0.8757	0.00	0.6578	0.00
ZDT2	-0.5738	0.00	0.8583	0.00	0.6637	0.00
ZDT3	-0.6360	0.00	0.8635	0.00	0.5453	0.00
ZDT4	-0.2968	0.00	0.6044	0.00	0.2494	0.00
ZDT6	-0.5508	0.00	0.8733	0.00	0.6111	0.00
L3-ZDT1	-0.8690	0.00	0.8634	0.00	0.7693	0.00
L3-ZDT2	-0.8598	0.00	0.8702	0.00	0.7707	0.00
L3-ZDT3	-0.8648	0.00	0.8569	0.00	0.6782	0.00
L3-ZDT4	-0.5529	0.00	0.5564	0.00	0.6873	0.00
L3-ZDT6	-0.7542	0.00	0.8227	0.00	0.7181	0.00

The results for most of the ZDT problems show more variation in the value of $\aleph$. A large amount of non-dominated solutions is found for ZDT1 and ZDT3 with several different control parameter combinations. Problems ZDT2 and ZDT6 have concave Pareto-fronts, and from the results it can be seen that with some control parameter values the cardinality of the obtained non-dominated set is lower. The probable reason for this is that sometimes the resulting set for these problems has converged to a single solution point even though there is a diversity preservation mechanism in use. This is due to the fact that in the ZDT problems, the first objective depends on one decision variable while the second objective depends on the rest of the decision variables. This easily leads to a situation where the first objective gets optimized faster than the second one, and the population might lose its variability along this decision variable, leading to the convergence of the front to a single point. A small CR value is preferable, because then the search proceeds along the variable axes and convergence to a single point of the Pareto-optimal front does not occur so easily. Especially with ZDT4, a small CR value is preferable as it can be seen from the $\aleph$ surface in Fig. 9. From the GD and S surfaces of ZDT4, a better performance or "valleys" can be observed when F is 0.5, 1.0, and 2.0. An explanation for this is that ZDT4 has multiple local fronts with equal spacing. With the aforementioned F values and three decision vectors having solutions on local fronts, mutation of DE creates a vector, which again has a solution on a local front (except if vectors

in the difference term correspond adjoining local fronts and $F = 0.5$). This causes the search to proceed faster.

When the performance metric surfaces between GDE in [6] and GDE3 were compared, GD and S surfaces were found quite similar except that GDE3 provided a better distribution than GDE according to S. The biggest differences were observed with the $\aleph$ surfaces. GDE3 is able to produce Pareto-front approximation with a larger amount of points than GDE, when control parameter values were varied, and therefore GDE3 seems to be less sensitive to the selection of control parameter values compared to GDE. To illustrate the difference of results between GDE and GDE3 with the same setups, the mean values of performance metrics over the control parameter ranges are shown in Table II. GDE3 provides better overall results compared to GDE according to the mean value of the metrics in all the cases except one: with KUR the obtained front is on an average slightly closer to the Pareto-front with GDE than with GDE3.

TABLE II

AVERAGES OF PERFORMANCE METRIC VALUES OVER THE CONTROL PARAMETER RANGES $CR \in [0, 1]$ AND $F \in [0, 3]$ FOR GDE AND GDE3

	$\overline{\overline{\aleph}}$		$\overline{\overline{GD}}$		$\overline{\overline{S}}$	
	GDE	GDE3	GDE	GDE3	GDE	GDE3
SCH1	27.6092	**38.6723**	47505	**41974**	0.0872	**0.0767**
SCH2	41.1027	**92.4175**	0.0044	**0.0012**	0.0276	**0.0090**
FON	17.7244	**60.2501**	0.1269	**0.0851**	0.1203	**0.0697**
KUR	29.6221	**95.0428**	**0.0515**	0.0517	0.0525	**0.0074**
POL	25.7151	**97.7713**	0.1217	**0.0358**	0.0331	**0.0057**
ZDT1	32.5491	**54.2051**	0.2447	**0.1627**	0.0521	**0.0284**
ZDT2	9.8706	**39.0638**	0.5764	**0.3764**	0.2195	**0.1095**
ZDT3	32.0746	**53.8696**	0.1998	**0.1466**	0.0407	**0.0225**
ZDT4	7.2944	**23.0973**	15.8174	**13.3424**	0.2284	**0.2028**
ZDT6	8.5628	**35.6411**	1.2279	**0.6626**	0.2003	**0.0915**

Based on the results, the same control parameter ranges $CR \in [0, 1]$ and $F \in (0, 1+]$ are usable for both single- and multi-objective optimization, and it is safer to use smaller values for CR. Overall, it is wise to select values for control parameters CR and F satisfying the condition $1.0 < c < 1.5$.

V. CONCLUSIONS

Different control parameter values for Generalized Differential Evolution 3 (GDE3) were tested using bi-objective test problems and metrics measuring the convergence and diversity of the solutions. Results with GDE3 were compared to the first version of GDE, and it can be concluded that GDE3 is more robust in terms of selection of control variables, and provides in general better results as inferred from the used metrics. Based on the empirical results, suitable control parameter ranges for multi-objective optimization are the same as for single-objective optimization, *i.e.*, $CR \in [0, 1]$ and $F \in (0, 1+]$. The results also propose that a larger F value than usually used in the case of single-objective optimization, does not give any extra benefit in the case of multi-objective optimization. It also seems that in some cases it is better to use a smaller CR value to prevent the solution from converging to a single point of the Pareto-front.

Also in the case of multi-objective optimization, the non-linear relationship between CR and F was observed according to the theory of the basic single-objective DE, about the relationship between the control parameters and the development of the population variance c. It is advisable to select values for CR and F satisfying the condition $1.0 < c < 1.5$.

In the future, a further investigation of the theory about the development of the population variance and suitability for multi-objective problems is necessary. Also, extending studies for problems with more than two objectives remains to be studied. Special cases, as multi-objective problems without conflicting objectives and single-objective problems transformed into multi-objective forms, might be interesting to investigate.

ACKNOWLEDGMENTS

First author gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), Technological Foundation of Finland, Finnish Cultural Foundation, and Centre for International Mobility (CIMO). He also wishes to thank all the KanGAL people for their help and fruitful discussions.

REFERENCES

[1] J. Lampinen, "DE's selection rule for multiobjective optimization," Lappeenranta University of Technology, Department of Information Technology, Tech. Rep., 2001, [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 15.2.2006.

[2] S. Kukkonen and J. Lampinen, "An extension of Generalized Differential Evolution for multi-objective optimization with constraints," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 752–761.

[3] ——, "GDE3: The third evolution step of Generalized Differential Evolution," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 443–450.

[4] R. Storn and K. V. Price, "Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[5] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer-Verlag, 2005.

[6] S. Kukkonen and J. Lampinen, "An empirical study of control parameters for generalized differential evolution," in *Proceedings of the Sixth Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005)*, Munich, Germany, Sept 2005, p. 12 pages.

[7] K. V. Price, *New Ideas in Optimization*. London: McGraw-Hill, 1999, ch. An Introduction to Differential Evolution, pp. 79–108.

[8] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2001.

[9] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages, and evolutionary methodologies," in *Proceedings of the Genetic and Evolutioanary Computation Conference (GECCO 2006)*, Seattle, WA, USA, July 2006, accepted for publication.

[10] S. Kukkonen and K. Deb, "Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006, accepted for publication.

[11] D. Zaharie, "Critical values for the control parameters of Differential Evolution algorithms," in *Proceedings of Mendel 2004, 8th International Conference on Soft Computing*, Brno, Czech Republic, June 2002, pp. 62–67.

Publication VII

Kukkonen, S., Deb, K.,
A Fast and Effective Method for Pruning of Non-Dominated Solutions in Many-Objective Problems

Reprinted, with kind permission of Springer Science and Business Media, from *The 9th International Conference on Parallel Problem Solving from Nature (PPSN IX), Lecture Notes in Computer Science (LNCS), Vol. 4193, Reykjavik, Iceland, September, 2006, pages 553–562.*

A more comprehensive version of the article is available as a technical report from *KanGAL Report No. 2007004, Indian Institute of Technology (IIT) Kanpur, India, March, 2007, [online] http://www.iitk.ac.in/kangal/papers/k2007004.pdf, 15.4.2012.*

A Fast and Effective Method for Pruning of Non-dominated Solutions in Many-Objective Problems

Saku Kukkonen[1] and Kalyanmoy Deb[2]

[1] Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20, FIN-53851 Lappeenranta, Finland
saku.kukkonen@lut.fi
[2] Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
deb@iitk.ac.in

Abstract. Diversity maintenance of solutions is an essential part in multi-objective optimization. Existing techniques are suboptimal either in the sense of obtained distribution or execution time. This paper proposes an effective and relatively fast method for pruning a set of non-dominated solutions. The proposed method is based on a crowding estimation technique using nearest neighbors of solutions in Euclidean sense, and a technique for finding these nearest neighbors quickly. The method is experimentally evaluated, and results indicate a good trade-off between the obtained distribution and execution time. Distribution is good also in many-objective problems, when number of objectives is more than two.

1 Introduction

Pruning a set of non-dominated solutions is a common and essential part of multi-objective evolutionary algorithms (MOEAs) such as the strength Pareto evolutionary algorithm (SPEA2) [1] and the elitist non-dominated sorting genetic algorithm (NSGA-II) [2]. An idea is to prune a non-dominated set to have a desired number of solutions in such a way that the remaining solutions have as good diversity as possible, meaning that the spread of extreme solutions is as high as possible, and the relative distance between solutions is as equal as possible. The best way to obtain a good distribution would be using some clustering algorithm. However, this is computationally expensive, since clustering algorithms usually take time $O\left(MN^2\right)$ to prune a set of size N with M objectives [3]. The complexity makes clustering techniques inapplicable to large population sizes, especially as pruning is usually done after each generation. The pruning technique of SPEA2 is based on finding the kth nearest neighbor of solutions, and has complexity of clustering because of a naive implementation.

In NSGA-II, the pruning of non-dominated solutions takes time $O\left(MN \log N\right)$ based on the *crowding distance*. The pruning method of NSGA-II provides good

T.P. Runarsson et al. (Eds.): PPSN IX, LNCS 4193, pp. 553–562, 2006.

diversity in the case of two objectives, but when the number of objectives is more than two, the obtained diversity declines drastically [4]. The reason for the bad performance is the fact that the crowding distance fails to approximate the crowding of the solutions when the number of the objectives is more than two [5].

Pruning of non-dominated solutions can be seen as a multi-objective optimization problem in itself: the method should provide as good diversity as possible and be as fast as possible. It has been considered that these objectives are conflicting. However, this paper proposes a new algorithm with two different crowding estimation techniques for pruning non-dominated solutions in such a way that the obtained diversity is good also in the case of more than two objectives, and the consumed time is considerably less than in clustering.

2 Proposed Pruning Method

The basic idea of the proposed pruning method is to eliminate the most crowded members of a non-dominated set one by one, and update the crowding information of the remaining members after each removal. This idea is trivial but it contains two problems: how to efficiently determine the crowding of members and how to efficiently update the crowding information of remaining members after removal. Straightforward approaches for these computations are in time complexity class $O(MN^2)$, which makes them inapplicable to large population sizes. Therefore two approaches for crowding estimation based on the nearest neighbors of solution candidates are introduced, and then a technique for finding these nearest neighbors *quickly* is introduced.

2.1 Crowding Estimation Based on Nearest Neighbors

2-NN. Probably the simplest crowding estimation technique is to measure the distance between a solution and its nearest neighbor solution, and use this distance to estimate crowding. The solution having the smallest distance is considered as the most crowded. When the Euclidean (L_2) distance metric is used for distance calculations, there will always be two solutions having the same smallest distance to the nearest neighbor due to the symmetry property of the metric. Instead of selecting randomly one of the two solutions, the solutions can be ordered according to the distance to the second nearest neighbor. The solution having smaller distance to second nearest neighbor is more crowded. If the distance to the nearest and second nearest neighbors is marked with $L_2^{\mathrm{NN}_1}$ and $L_2^{\mathrm{NN}_2}$, respectively, then a distance vector $d_{2-\mathrm{NN}} = [L_2^{\mathrm{NN}_1}, L_2^{\mathrm{NN}_2}]$ is attached to each solution. In the case of real-coded variables and continuous objective functions, this provides a crowding measure, which usually establishes unambiguous ordering of solutions having the same smallest distance to the nearest neighbor.

***M*-NN.** A bit more developed idea is to use the k nearest neighbors for crowding estimation in such a way that distances to the k nearest neighbors are *multiplied*

together, and the solution having the smallest product is considered the most crowded. The product of distances, which is here called the *vicinity distance*, is a simple measure, but it already manages to contain information about vicinity and location. The idea is extended in such a way that the number of nearest neighbors for crowding estimation calculations is kept the same as the number of objectives, i.e., $k = M$. More formally vicinity distance is defined $d_{M-\mathrm{NN}} = \prod_{i=1}^{M} L_2^{\mathrm{NN}_i}$, where $L_2^{\mathrm{NN}_i}$ is distance to the ith nearest neighbor according to L_2 distance metric.

2.2 Efficient k Nearest Neighbor Search

Finding the k nearest neighbors (k-NN) is a widely used operation in vector quantization (VQ), and many efficient algorithms have been proposed during the last three decades. The exact k-NN search technique used here is known as the *equal-average nearest neighbor search (ENNS) algorithm* [6, 7], and it uses the following theorem for the Euclidean (L_2) distance measure to reduce the amount of distance calculations:

$$\left| \frac{1}{M} \sum_{i=1}^{M} x_i - \frac{1}{M} \sum_{i=1}^{M} y_i \right| \leq \frac{L_2(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{M}} \Leftrightarrow \left(\sum_{i=1}^{M} x_i - \sum_{i=1}^{M} y_i \right)^2 \leq M L_2(\boldsymbol{x}, \boldsymbol{y})^2 \ . \quad (1)$$

Thus, if the sums of elements of given vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are known, an upper bound for the Euclidean distance between them can be calculated. It is more convenient to use the squared Euclidean distance, since the actual distance values are not needed for finding neighbors, and calculating the square root is an expensive operation computationally. The geometrical interpretation of (1) is that vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are projected to a central axis of a hypercube (a projection vector starting from origin and going through point $(1, 1, \ldots, 1)$) and then the difference of projection values is at most equal to the Euclidean distance between vectors divided by the square root of the number of elements in the vectors. It has been proved that (1) holds also for any other projection vector $\boldsymbol{p}$ than the central axis of a hypercube, and (1) transforms into form [8]:

$$(p_x - p_y)^2 \leq L_2(\boldsymbol{x}, \boldsymbol{y})^2 \ , \quad \text{where} \quad p_x = \frac{\boldsymbol{p} \cdot \boldsymbol{x}}{|\boldsymbol{p}|} \quad \text{and} \quad p_y = \frac{\boldsymbol{p} \cdot \boldsymbol{y}}{|\boldsymbol{p}|} \ . \quad (2)$$

It is useful to select the projection axis to represent the direction in which the vectors have the largest variance, and such axis can be obtained using, e.g., principal component analysis (PCA) [8]. However, data analysis is not necessarily needed in the case of a set of non-dominated solutions, since there already exists information on how the vectors/solutions are distributed. When all the objectives are to be minimized (or maximized), there exists such kind of monotonicity between objective values that when values of $M-1$ objectives increase, then the objective value of the remaining objective decreases. Thus, there is a negative correlation between objective values, and the projection axis can be chosen to go through points $(0, 0, \ldots, 0, 1)$ and $(1, 1, \ldots, 1, 0)$ if the objective values have

been normalized to the value range $[0, 1]$. The projection axis chosen in this way in the case of two and three objectives has been illustrated in Fig. 1 for sets of non-dominated solution points. Projections of the points are also illustrated.

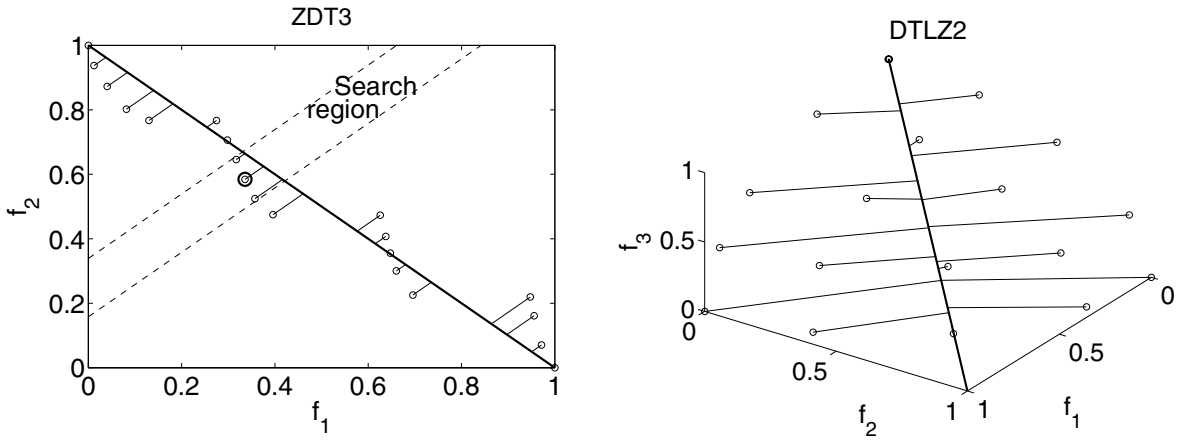


Fig. 1. Selected projection axis in the case of two and three objectives when k-NN search based on (2) is used

The ENNS technique can be used to find the nearest neighbors for solutions in such a way that first the projected values of the vectors are calculated, and these values are sorted. Then, for each solution vector, the distance to the solution having the nearest projected value is set as the best minimum distance found so far. The distances to other neighbor solutions according to the projected value are also calculated, and the minimum distance value is updated accordingly, as long as there exist neighbors for which (2) holds. This technique can be extended easily for finding the k nearest neighbors using k-th smallest distance instead of the minimum distance to reject distant solutions with (2) [8].

Besides ENNS, a technique called partial distortion search (PDS) algorithm [9] is used to speed up the execution further. It interrupts the distance calculation if a partial sum of elements exceeds a known minimum distance value. This technique adds one extra comparison but decreases the number of mathematical calculations speeding up the execution.

2.3 The Proposed Algorithm

The proposed pruning algorithm is based on the crowding estimation and k-NN search techniques presented above, and minimization of all objectives is assumed. For efficient maintenance of crowding information of remaining solutions after removal of the most crowded solution, a priority queue such as a heap [10, pp. 140–152] is used. The proposed algorithm is:

Pruning of non-dominated set
input: a non-dominated set $\mathcal{F}$ (objectives are minimized),
the size n of a desired pruned set
output: elements of a heap $\mathcal{H}$

1 find minimum ($f_i^{\min}$) and maximum ($f_i^{\max}$) values of each objective i of the members of $\mathcal{F}$ and normalize members according to the formula $f_i = (f_i - f_i^{\min})/(f_i^{\max} - f_i^{\min})$
2 for each member of $\mathcal{F}$, change Mth objective f_M to value $1 - f_M$ and calculate a sum of objective values, m
3 for each member of $\mathcal{F}$, find nearest neighbors (cf. Sect. 2.2) and calculate distance measure d (cf. Sect. 2.1) for crowding estimation
4 for members of $\mathcal{F}$ having a minimum or maximum objective value, assign $d = \infty$
5 create an ascending heap $\mathcal{H}$ from the members of $\mathcal{F}$
6 while $|\mathcal{H}| > n$
7 remove an element (root node) with a minimum d value from $\mathcal{H}$ and update $\mathcal{H}$
8 for the neighbors of the removed element
9 calculate a new d value
10 replace old d value in $\mathcal{H}$ with the new one and update $\mathcal{H}$

The first operation of the algorithm in line 1 is the normalization of objective values of the obtained non-dominated set. Different objectives might have very different value ranges, and using unnormalized values would distort the obtained distribution. The time complexity of the normalization is $O(MN)$. The normalization can lead to an infinite value if minimum and maximum values for some objective are same. Then this objective can be discarded from calculations. The second line of the algorithm re-maps the M-th objective value f_M to value $1 - f_M$ so that the original form (1) of ENNS can be used directly. The complexity of the operations in line 2 is $O(MN)$. Finding the M nearest neighbors (in the case of M-NN crowding measure) for all the members of given set in line 3 is known as the *all-M-nearest-neighbor problem*, and it can be done in time $O(MN \log N)$ [11]. Line 4 takes time $O(M)$ and makes sure that members having extreme objective values are removed last. Creating a heap in line 5 takes time $O(N \log N)$. The while-loop in lines 6–10 is executed at most N times (on average $N/2$ times). Removing a minimum element from the heap and updating the heap to have correct structure in line 7 takes time $O(\log N)$. If the M-NN crowding measure is used, each member of a non-dominated set has on average M neighbors, whose crowding values are affected if the member is removed (these neighbors can be found easily if neighborhood information is also stored when the nearest neighbors have been searched in line 3). Therefore, the for-loop in lines 8–10 is executed M times on average. The calculation of a new crowding value in line 9 means finding the M nearest neighbors. This can be done in time $O(M \log N)$ for a static set of vectors. Now, the set of vectors is changing (reducing) and the actual time complexity depends on how the vectors are distributed. Finally, replacing the crowding value to the heap, and updating the heap to have the correct structure in line 10 takes time $O(\log N)$.

This analysis leads to a complexity class estimate $O(M^2 N \log N)$ for the whole pruning algorithm. However, when the k-NN method presented earlier in this

section is used in lines 3 and 9, the actual time complexity depends on how members of the non-dominated set are distributed on the selected projection axis. As the pruning method is used extensively, the most interesting thing is to know the expected complexity in practice. This, as well as the performance of the method according to the obtained diversity, is evaluated experimentally in the following.

3 Experiments

The proposed pruning method with the two introduced crowding estimation techniques was implemented in the Generalized Differential Evolution 3 (GDE3) [12], which was then used to solve test problems. Also, pruning based on the crowding distance as in NSGA-II was implemented in GDE3. GDE3 is an extension of Differential Evolution (DE) [13] for constrained multi-objective optimization. Roughly speaking, the evolutionary part of the algorithm is DE and the multi-objective part is from NSGA-II [2]. This combination has been shown to give benefit over NSGA-II with rotated problems [14]. Furthermore, GDE3 has other improvements over NSGA-II, and an interested reader is advised to see references [12, 5].

NSGA-II and SPEA2 were used for comparison[1], and the diversity of the obtained results was measured using spacing, maximum spread, and hypervolume [15, pp. 327–333]. The spacing (S) measures the standard deviation of distances from each vector to the nearest vector in the obtained non-dominated set. A smaller value for S is better, and for an ideal distribution $S = 0$. The maximum spread (D) measures the length of the diagonal of a minimal hyperbox, which encloses the obtained non-dominated set, and a larger value tells about a larger spread between extreme solutions. The hypervolume (HV) calculates the volume of the objective space between the solutions and a reference (nadir) point, and a larger value is better. The hypervolume measures both diversity and convergence, but reflects more of convergence in its value.

Due to space limitations, only a small part of results are shown. The complete set of results (including figures and more test problems) can be found in [16], where also the code for the proposed pruning method is provided.

3.1 Bi-objective Problems

Bi-objective test problems, ZDT1, ZDT3, and ZDT6 [15, pp. 356–360] were solved using population size 100 and 1000 generations. Tests were repeated 100 times and mean & standard deviation values are reported in Table 1. According to the spacing value, the proposed pruning method provides the best diversity, and the M-NN crowding estimation technique is slightly better. Differences in the maximal spread and hypervolume values are minimal compared to the differences in the spacing values. The proposed pruning method takes about two times longer to

[1] Code for NSGA-II was obtained from the web site www.iitk.ac.in/kangal and for SPEA2 from the web site www.tik.ee.ethz.ch/pisa.

Table 1. Mean and standard deviation values of spacing (S), maximum spread (D), hypervolume (HV), and CPU times for ZDT test problems measured from 100 independent runs. GDE3 is implemented with pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN).

Problem	Method	S	D	HV	Total time (s)	Pruning time (s)
ZDT1	NSGA-II	$6.9175e-03 \pm 5.6118e-04$	$\mathbf{1.4146e+00} \pm 2.2506e-03$	$3.6604e+00 \pm 2.3269e-04$	$2.4076e+00 \pm 1.0456e-02$	$1.5600e-01 \pm 3.8006e-02$
	GDE3, CD	$6.4240e-03 \pm 5.5946e-04$	$1.4113e+00 \pm 1.4634e-03$	$3.6610e+00 \pm 1.2739e-03$	$1.4877e+00 \pm 1.3246e-02$	$\mathbf{1.0530e-01} \pm 3.1669e-02$
	GDE3, 2-NN	$2.8501e-03 \pm 2.8597e-04$	$1.4114e+00 \pm 1.5524e-03$	$3.6615e+00 \pm 1.7195e-03$	$\mathbf{1.3560e+00} \pm 6.5134e-03$	$2.3340e-01 \pm 3.6048e-02$
	GDE3, M-NN	$\mathbf{2.6305e-03} \pm 2.8921e-04$	$1.4115e+00 \pm 1.5376e-03$	$3.6616e+00 \pm 1.1364e-03$	$1.3570e+00 \pm 7.4536e-03$	$2.3260e-01 \pm 4.1013e-02$
	SPEA2	$3.2063e-03 \pm 2.9122e-04$	$1.4142e+00 \pm 7.2339e-05$	$\mathbf{3.6619e+00} \pm 3.6100e-05$	$1.1756e+01 \pm 7.0249e-02$	$7.6469e+00 \pm 1.9474e-01$
ZDT3	NSGA-II	$4.9342e-03 \pm 4.5888e-04$	$\mathbf{1.9676e+00} \pm 1.0790e-03$	$4.8146e+00 \pm 1.3933e-04$	$2.3584e+00 \pm 9.3980e-03$	$1.5770e-01 \pm 3.7196e-02$
	GDE3, CD	$4.3573e-03 \pm 4.1398e-04$	$1.9639e+00 \pm 1.7943e-03$	$4.8151e+00 \pm 1.0630e-04$	$1.3838e+00 \pm 5.4643e-03$	$\mathbf{8.5400e-02} \pm 2.9074e-02$
	GDE3, 2-NN	$1.7614e-03 \pm 1.9583e-04$	$1.9600e+00 \pm 3.6983e-02$	$4.8117e+00 \pm 3.6699e-02$	$1.2235e+00 \pm 5.9246e-03$	$1.6420e-01 \pm 4.5797e-02$
	GDE3, M-NN	$\mathbf{1.6415e-03} \pm 1.6563e-04$	$1.9639e+00 \pm 1.5796e-03$	$\mathbf{4.8154e+00} \pm 2.5151e-05$	$\mathbf{1.2194e+00} \pm 5.4717e-03$	$1.6510e-01 \pm 4.3797e-02$
	SPEA2	$3.0892e-03 \pm 3.4177e-04$	$1.9673e+00 \pm 1.0494e-04$	$2.8151e+00 \pm 6.1957e-05$	$1.1500e+01 \pm 7.2117e-02$	$7.3613e+00 \pm 2.2574e-01$
ZDT6	NSGA-II	$8.3199e-03 \pm 6.6005e-04$	$1.0463e+00 \pm 1.2942e-04$	$2.9204e+00 \pm 2.8917e-04$	$1.3882e+00 \pm 8.2118e-03$	$1.4790e-01 \pm 3.3584e-02$
	GDE3, CD	$6.6047e-03 \pm 6.5536e-04$	$1.1648e+00 \pm 2.5260e-02$	$3.0209e+00 \pm 1.3448e-01$	$\mathbf{1.2147e+00} \pm 5.7753e-03$	$\mathbf{9.6869e-02} \pm 3.4866e-02$
	GDE3, 2-NN	$2.7362e-03 \pm 3.0119e-04$	$1.1656e+00 \pm 2.0350e-02$	$3.0265e+00 \pm 1.0448e-01$	$1.2347e+00 \pm 5.7656e-03$	$2.3490e-01 \pm 3.5971e-02$
	GDE3, M-NN	$\mathbf{2.6386e-03} \pm 2.8531e-04$	$1.1632e+00 \pm 3.1709e-02$	$3.0114e+00 \pm 1.8180e-01$	$1.2309e+00 \pm 6.2109e-03$	$2.3240e-01 \pm 3.6985e-02$
	SPEA2	$3.1010e-03 \pm 3.0937e-04$	$\mathbf{1.1685e+00} \pm 3.6318e-05$	$\mathbf{3.0412e+00} \pm 1.5286e-04$	$1.1662e+01 \pm 5.7654e-02$	$7.5451e+00 \pm 2.0404e-01$

execute than the pruning method based on the crowding distance. However, the time needed for pruning is less than 20% from total CPU time needed[2].

3.2 Tri-objective Problems

The proposed pruning method was also tested on a set of tri-objective test problems, DTLZ1, DTLZ4, and DTLZ7 [4], using population size 300 and 1000 generations. Results after one run are shown in [16], and numerical results for the problems from 100 repetition runs are shown in Table 2. The improvement over the pruning method based on the crowding distance is clearly visible in [16], and, visually, the proposed pruning method provides similar results compared to SPEA2. Also the spacing metric in Table 2 indicates the same, this time the 2-NN crowding estimation technique being slightly better in most cases compared to M-NN. As in the case of bi-objective problems, the maximal spread and hypervolume values have only small differences. The proposed pruning method is now at worst about eight times slower than the pruning method based on the crowding distance. The pruning time is intelligibly less for the 2-NN than for the M-NN crowding estimation technique, and it is also less for DTLZ7, which does not have continuous Pareto-front.

[2] The total CPU time for the proposed pruning method is smaller compared to GDE3 with the crowding distance because of overall speedups in the program code. GDE3 uses a naive $O(MN^2)$ non-dominated sorting implementation instead of faster $O(N \log^{M-1} N)$ implementation [3].

Table 2. Mean and standard deviation values of spacing (S), maximum spread (D), hypervolume (HV), and CPU times for DTLZ test problems measured from 100 independent runs. GDE3 is implemented with pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN).

Problem	Method	S	D	HV	Total time (s)	Pruning time (s)
DTLZ1	NSGA-II	$2.7301e-02 \pm 3.6882e-03$	$8.0296e-01 \pm 4.8071e-02$	$9.6937e-01 \pm 2.0890e-03$	$\mathbf{9.3375e+00} \pm 6.6284e-02$	$6.3347e-01 \pm 1.2919e-01$
	GDE3, CD	$2.3927e-02 \pm 1.1410e-03$	$9.0238e-01 \pm 1.7048e-01$	$9.6748e-01 \pm 3.4917e-02$	$1.3648e+01 \pm 1.1168e-01$	$\mathbf{6.2460e-01} \pm 6.3444e-02$
	GDE3, 2-NN	$1.0591e-02 \pm 7.4208e-04$	$8.8532e-01 \pm 1.2251e-01$	$9.7272e-01 \pm 2.3476e-02$	$1.6237e+01 \pm 1.0560e-01$	$3.5605e+00 \pm 8.7404e-02$
	GDE3, M-NN	$1.1260e-02 \pm 6.8360e-04$	$\mathbf{9.1129e-01} \pm 1.9040e-01$	$9.6780e-01 \pm 3.6437e-02$	$1.7627e+01 \pm 1.6803e-01$	$5.1749e+00 \pm 1.0265e-01$
	SPEA2	$\mathbf{8.7750e-03} \pm 2.0557e-03$	$8.6981e-01 \pm 2.1762e-02$	$\mathbf{9.7622e-01} \pm 3.9209e-05$	$1.2211e+02 \pm 1.2821e-01$	$7.6808e+01 \pm 1.7632e-01$
DTLZ4	NSGA-II	$3.1285e-02 \pm 1.5006e-03$	$1.7385e+00 \pm 1.0485e-02$	$7.3842e+00 \pm 1.5108e-02$	$1.5752e+01 \pm 8.6729e-02$	$7.7610e-01 \pm 6.4790e-02$
	GDE3, CD	$2.8450e-02 \pm 1.2669e-03$	$1.7321e+00 \pm 1.0347e-06$	$7.4252e+00 \pm 2.9245e-03$	$\mathbf{1.3813e+01} \pm 5.8934e-02$	$\mathbf{6.4060e-01} \pm 6.9614e-02$
	GDE3, 2-NN	$1.4207e-02 \pm 1.6561e-03$	$1.7289e+00 \pm 3.1785e-02$	$7.4330e+00 \pm 1.0168e-01$	$1.6918e+01 \pm 4.9857e+00$	$4.1224e+00 \pm 4.9245e+00$
	GDE3, M-NN	$1.5221e-02 \pm 1.1413e-03$	$1.7321e+00 \pm 1.6773e-07$	$\mathbf{7.4437e+00} \pm 2.2990e-04$	$1.8109e+01 \pm 1.1213e-01$	$5.1986e+00 \pm 1.7774e-01$
	SPEA2	$\mathbf{1.2543e-02} \pm 2.4636e-03$	$\mathbf{1.7465e+00} \pm 7.0831e-02$	$7.3918e+00 \pm 1.9807e-01$	$1.3105e+02 \pm 1.0151e+00$	$8.7409e+01 \pm 1.2951e+00$
DTLZ7	NSGA-II	$2.3073e-02 \pm 1.6451e-03$	$3.5925e+00 \pm 4.8045e-02$	$1.3493e+01 \pm 2.4877e-02$	$1.4823e+01 \pm 1.4667e-01$	$7.7070e-01 \pm 1.0215e-01$
	GDE3, CD	$2.3658e-02 \pm 2.6283e-03$	$\mathbf{3.6179e+00} \pm 7.8436e-03$	$1.3545e+01 \pm 1.9480e-02$	$\mathbf{1.2066e+01} \pm 2.2497e-02$	$\mathbf{5.5920e-01} \pm 6.4834e-02$
	GDE3, 2-NN	$\mathbf{9.1687e-03} \pm 1.1766e-03$	$3.6153e+00 \pm 3.0744e-03$	$1.3586e+01 \pm 5.3207e-03$	$1.2809e+01 \pm 2.9450e-01$	$2.2352e+00 \pm 1.0391e-01$
	GDE3, M-NN	$9.7859e-03 \pm 8.8586e-04$	$3.6155e+00 \pm 2.6746e-03$	$\mathbf{1.3588e+01} \pm 3.7791e-03$	$1.3540e+01 \pm 3.3633e-02$	$3.0734e+00 \pm 6.1664e-02$
	SPEA2	$1.5140e-02 \pm 8.2394e-04$	$3.6067e+00 \pm 7.1954e-03$	$1.3577e+01 \pm 1.1132e-02$	$1.3342e+02 \pm 6.7649e-02$	$8.8697e+01 \pm 1.0649e-01$

3.3 Measured Pruning Time Complexity

The time complexity of the proposed pruning method was verified experimentally in the case of two and three objectives using ZDT1 and DTLZ1. The measured pruning times for various population sizes are shown in Fig. 2.

In the bi-objective case (ZDT1), the proposed pruning method takes about twice the time of the pruning method based on the crowding distance, and complexity classes of the methods are same. Intelligibly, there is no notable difference between the observed pruning times with the 2-NN and M-NN crowding estimation techniques.

In the tri-objective case (DTLZ1), the execution time of the proposed pruning method with the M-NN crowding estimation technique is at worst ten times more than the time with the pruning method based on the crowding distance, and the pruning time with the 2-NN crowding estimation technique is less than with the M-NN crowding estimation technique. It seems that the proposed pruning method scales well with the population size.

The difference between the measured pruning times in bi- and tri-objective cases is larger than the estimated complexity class $O(M^2 N \log N)$ predicts. The reason for this is probably the fact that the search of the nearest neighbors in the case of two objectives is relatively easy because of the monotonic relation between solutions in the objective space. On the other hand, the difference between the 2-NN and M-NN crowding estimation techniques in the tri-objective case is relatively small, although the difference should be in proportion to M^2.

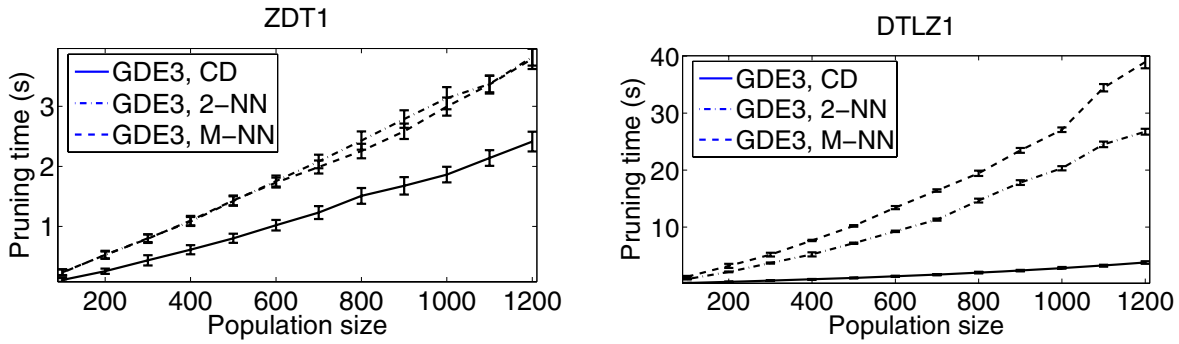


Fig. 2. Average and standard error of pruning times measured from 100 runs of solving ZDT1 and DTLZ1 by GDE3 with pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

A probable reason for this is that there are constant calculations, which decrease the proportional difference of measured times.

Estimation of the actual complexity class would require more problems with different Pareto-fronts and a larger number of objectives. However, the measured pruning times appear logarithmic and considerably smaller than that of the pruning method in SPEA2 [5].

4 Conclusions

A pruning method with two different crowding estimation techniques for pruning a set of non-dominated solutions has been proposed. The method is based on crowding estimation using nearest neighbors of solutions and a fast technique for finding the nearest neighbors.

According to the experimental results, the proposed pruning method provides a better distribution than the pruning method based on the crowding distance. Especially in the case of tri-objective problems, the obtained diversity is significantly better. The obtained distribution is observed to be similar to the distribution obtained with SPEA2. The execution time needed for the proposed pruning method is more than for the pruning method based on the crowding distance but significantly less than for the pruning method in SPEA2.

Based on the results, the proposed method provides near optimal distribution, which does not need improvement. The execution time might be still reduced even thought it is currently reasonable. Evaluating the performance in the case of more than three objectives, remain as future work.

References

1. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Proceedings of the Third Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2001), Athens, Greece (2002) 95–100

2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
3. Jensen, M.T.: Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. IEEE Transactions on Evolutionary Computation **7**(5) (2003) 503–515
4. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Evolutionary Multiobjective Optimization. Springer-Verlag, London (2005) 105–145
5. Kukkonen, S., Deb, K.: Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In: Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada (2006) Accepted for publication.
6. Guan, L., Kamel, M.: Equal-average hyperplane partioning method for vector quantization of image data. Pattern Recognition Letters **13**(10) (1992) 693–699
7. Ra, S.W., Kim, J.K.: A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. IEEE Transactions on Circuits and Systems-II **40**(9) (1993) 576–579
8. Baek, S., Sung, K.M.: Fast K-nearest-neighbour search algorithm for nonparametric classification. IEE Electronics Letters **36**(21) (2000) 1821–1822
9. Bei, C.D., Gray, R.M.: An improvement of the minimum distortion encoding algorithm for vector quantization. IEEE Transactions on Communications **COM-33**(10) (1985) 1132–1132
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. Prentice-Hall (1990)
11. Vaidya, P.M.: An $O(n \log n)$ algorithm for the all-nearest-neigbors problem. Discrete & Computational Geometry **4** (1989) 101–115
12. Kukkonen, S., Lampinen, J.: GDE3: The third evolution step of Generalized Differential Evolution. In: Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005), Edinburgh, Scotland (2005) 443–450
13. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, Berlin (2005)
14. Inorio, A., Li, X.: Solving rotated multi-objective optimization problems using Differential Evolution. In: Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004), Cairns, Australia (2004) 861–872
15. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, England (2001)
16. Kukkonen, S.: A fast and effective method for pruning of non-dominated solutions in many-objective problems, results (2006) [Online] Available: http://www.it.lut.fi/ip/evo/pruning , 15.6.2006.