Diss. ETH No. 15251

Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
doctor of sciences

presented by
MARCO LAUMANNS
Dipl.-Inform., University of Dortmund, Germany
born September 30, 1973
citizen of Germany

accepted on the recommendation of
Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Kalyanmoy Deb, co-examiner

2003

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 55

Marco Laumanns

Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to the
Swiss Federal Institute of Technology Zurich
for the degree of Doctor of Sciences

Diss. ETH No. 15251

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Kalyanmoy Deb, co-examiner

Examination date: August 28, 2003

Abstract

This thesis deals with the analysis and application of evolutionary algorithms for optimization problems with multiple objectives. Many application problems involve (i) a system model that is not given in closed analytical form and (ii) multiple, often conflicting optimization criteria. Both traits hamper the application of classical optimization techniques, which require a certain structure of the problem and are mostly designed to handle only a single objective. For this problem domain, the class of randomized search heuristics, to which evolutionary algorithms also belong, have become popular. Due to their population concept, evolutionary algorithms can process multiple solutions in parallel and can therefore cope with different objectives more naturally.

Like most randomized search algorithms, evolutionary algorithms are easy to describe and implement, but hard to analyze theoretically. Despite much empirical knowledge and successful application, only few theoretical results concerning their effectiveness and efficiency are available. This holds especially in the multiobjective case where these questions have not been investigated yet. However, even from a practical point of view is is important to distinguish

- whether a given algorithm is capable of solving a given problem (effectiveness); and
- the computational complexity (measured in computation time and memory requirements) of an algorithm to solve a given problem (efficiency).

The aim of this work is to contribute to the understanding of evolutionary algorithms for multiobjective optimization problems with respect to these questions. Specifically, the following topics are covered:

- Based on known concepts from decision theory, the topic of quality measurement is addressed, with respect to single solutions (via fitness functions) and sets of solutions (via quality indicators). The common mathematical framework allows us to compactly describe existing fitness functions and quality indicators as well as to analyze them theoretically.
- Convergence properties are investigated for the limit case of infinite running time, but finite memory resources. Based on the concept of ε-appproximations, new selection operators are proposed that guarantee the convergence of randomized search strategies to a well-defined discrete solution set with simultaneous consideration of diversity.
- In order to facilitate a running time analysis, simple model algorithms and problems are proposed and suitable proof techniques developed and applied. The results achieved concerning the expected running time show that through special selection operators, population-based approaches can be advantageous over multistart-strategies.

- Three case studies from the field of automotive engineering demonstrate how evolutionary algorithms can systematically be exploited in the design process. The applications underline the practical relevance of some results from the previous theoretical investigations.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Analyse und der Anwendung von evolutionären Algorithmen für Optimierungsprobleme mit mehrfacher Zielsetzung. Viele praktische Optimierungsprobleme basieren auf System-Modellen, die (i) nicht in geschlossener analytischer Form darstellbar sind und (ii) mehrere, konkurrierende Optimierungsziele beinhalten. Beide Eigenschaften erschweren die Anwendung klassischer Optimierungsverfahren, welche eine gewisse Struktur des Modells voraussetzen und in der Regel nur für Probleme mit einem einzigen Optimierungsziel entworfen wurden. Für diesen Anwendungsfall haben sogenannte randomisierte Suchverfahren eine gewisse Popularität erreicht, zu denen auch die Klasse der evolutionären Algorithmen gehört. Durch ihr Populationskonzept sind evolutionäre Algorithmen in der Lage, mehrere potentielle Lösungsalternativen des Optimierungsproblems gleichzeitig zu verarbeiten und so den verschiedenen Optimierungszielen besser gerecht zu werden.

Evolutionäre Algorithmen sind, wie die meisten randomisierte Suchverfahren, einfach zu beschreiben und zu implementieren. Ihr Verhalten ist jedoch oft sehr komplex und einer Analyse nur schwer zugänlich. Trotz mittlerweile viel Erfahrungswissens und vieler erfolgreicher Anwendungen existieren bisher nur wenige theoretische Resultate bezüglich ihrer Effektivität und ihrer Effizienz. Dies gilt insbesondere für den Fall mehrfacher Zielsetzung, für den diese fundamentalen Fragen bisher noch nie untersucht wurden. Dennoch ist es auch aus Anwendungssicht wichtig zu wissen

- ob ein gegebener Algorithmus zur Lösung eines gegebenen Problems im Stande ist (Effektivität) und
- mit welchem Aufwand, gemessen in Rechenzeit oder Speicherplatzbedarf, der Algorithmus ein gegebenes Problem löst (Effizienz).

Das Ziel dieser Arbeit ist es, zum Verständnis evolutionärer Algorithmen zur Lösung von Mehrziel-Optimierungsproblemen im Hinblick auf obige Fragen beizutragen. Speziell werden dazu folgende Themen behandelt:

- Basierend auf bekannten Konzepten der Entscheidungstheorie wird der Themenbereich der Qualitätsmessung behandelt, sowohl in Bezug auf Einzellösungen (mittels Fitnessfunktionen) als auch auf Mengen von Lösungen (mittels Qualitätsindikatoren). Die mathematisch vereinheitlichte Beschreibungsweise erlaubt zum einen, die existierenden Fitnessfunktionen und Qualitätsindikatoren kompakt zu beschreiben, und zum anderen, ihre Eigenschaften zu analysieren.
- Das Konvergenzeigenschaften werden für den Grenzfall unendlicher Laufzeit, aber endlichen Speicherplatzes untersucht. Basierend auf einem neuartigen Lösungskonzept werden Selektionsoperatoren vorgeschlagen, die die Konvergenz randomisierter Suchverfahren gegen eine wohldefinierte diskrete Lösungmenge unter Berücksichtigung von Diversität garantiert.

- Zum Zweck einer ersten Laufzeitanalyse werden einfache Modellalgorithmen und einfache diskrete Modellprobleme vorgeschlagen sowie geeignete Beweistechniken entwickelt. Die damit erzielten Resultate über die erwartete Laufzeit in Abhängigkeit der Problemgrösse zeigen, dass populationsbasierte Ansätze durch geschickte Seletionsverfahren Vorteile gegenüber Multistart-Strategien haben können.

- Anhand dreier Fallstudien aus der Automobiltechnik wird demonstriert, wie evolutionäre Algorithmen in einem Entwurfswurfsprozess gezielt eingesetzt werden können. Die Anwendungsbeispiele verdeutlichen die praktische Relevanz einzelner Aspekte aus den vorgängigen theoretischen Untersuchungen.

I would like to thank

- Prof. Lothar Thiele for supervising and guiding my research work,
- Prof. Kalyanmoy Deb for his willingness to be the co-examiner of my thesis,
- Prof. Eckart Zitzler for plenty of advice, help, and support.

Contents

List of Symbols and Abbreviations **1**

1 Introduction **3**
1.1 Background 3
1.2 Illustrative Example 5
1.3 Problem Statement 6
1.4 Thesis Contributions and Overview 8

2 Multiobjective Optimization with Evolutionary Algorithms **11**
2.1 Multiobjective Optimization 11
2.2 Evolutionary Algorithms 15
2.3 Selection under Multiple Objectives 18
2.3.1 Selection based on Fitness Functions 18
2.3.2 Properties of Fitness Functions 20
2.4 Performance Assessment 24
2.4.1 Approximation Sets and Quality Indicators 26
2.4.2 Absolute Quality Indicators 27
2.4.3 Relative Quality Indicators 34
2.5 Summary 36

3 Limit Behavior and Global Convergence **37**
3.1 Related Work 38
3.1.1 Algorithms for Guaranteed Convergence 39
3.1.2 Elitist Selection with Focus on Distribution Quality . . . 40
3.1.3 Limitations of Existing Selection Strategies 41
3.2 Algorithms for Convergence and Diversity 41
3.2.1 Concept of Pareto Set Approximation 41
3.2.2 Algorithm to Maintain an ε-approximate Pareto Set . . . 44
3.2.3 Algorithm to Maintain an ε-Pareto Set 46
3.3 Simulations 48
3.3.1 Convergence Behavior 48
3.3.2 Distribution Behavior 50
3.3.3 Results 52
3.4 Possible Extensions 52
3.4.1 Other Definitions of ε-Dominance 52

3.4.2 Guaranteeing Minimum Distances 52
3.4.3 Steering Search by Defining Ranges of Non-acceptance 53
3.4.4 Fixed Archive Size by Dynamic Adaptation of ε 53
3.5 Summary 55

4 Running Time Analysis 57
4.1 Methodology and Related Work 58
4.2 Two Example Problems 59
4.3 A Simple Evolutionary Multiobjective Optimizer 62
4.3.1 SEMO 62
4.3.2 Analysis of SEMO on LOTZ 63
4.3.3 Analysis of SEMO on COCZ and a General Upper Bound Technique 64
4.3.4 Comparing SEMO to a (1+1)-EA using Multistarts . . . 66
4.4 Two Improved Evolutionary Multiobjective Optimizers 69
4.4.1 FEMO and the Fair Sampling Strategy 69
4.4.2 GEMO and the Greedy Selection Mechanism 72
4.5 Higher-dimensional Objective Spaces 74
4.5.1 Multiobjective Leading Ones (mLOTZ) Problem 74
4.5.2 Multiobjective Count Ones Problem (mCOCZ) Problem 77
4.6 From One-bit to Independent-bit Mutations 79
4.7 Summary 80

5 Applications in Automotive Engineering 83
5.1 An Evolutionary Multiobjective Design Framework 84
5.2 Design Space Exploration of Road Trains 86
5.2.1 Optimization Problem 87
5.2.2 Algorithms 89
5.2.3 Results 90
5.3 Parameter Optimization of Adaptive Cruise Control Systems . . 93
5.3.1 Optimization Problem 93
5.3.2 Algorithms 94
5.3.3 Results 96
5.4 Model Fitting for a Vehicle Dynamics Simulation 100
5.4.1 Optimization Problem 100
5.4.2 Algorithms 101
5.4.3 Results 102
5.5 Summary 102

Bibliography 107

List of Symbols and Abbreviations

$\mathbb{N}$	set of natural numbers $\{1, 2, \ldots\}$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	set of n-dimensional real vectors
$\mathsf{P}\{A\}$	probability of event A
$\mathsf{E}[T]$	expectation of random variable T
Φ	fitness function
I	quality indicator
X	decision space
Y	objective space
f	objective function
m	number of objective functions
n	number of decision variables
x	decision alternative
y	objective vector
EA	evolutionary algorithm
MOEA	multiobjective evolutionary algorithm
SEMO	Simple Evolutionary Multiobjective Optimizer
FEMO	Fair Evolutionary Multiobjective Optimizer
GEMO	Greedy Evolutionary Multiobjective Optimizer

1

Introduction

1.1 Background

All areas of human interaction with its environment involve decision situations Decision making should ideally be based on complete knowledge of the alternatives at hand as well as their consequences. As the complex nature of the system under concern often renders exact predictions impossible, one usually has to rely on models, which provide tractable approximations to reality. Here, systems analysis plays an important role (Bell et al. 1977), since only a well-informed decision maker is in a position to take well-founded decisions. In systems analysis, three interrelated activities can be distinguished based on different points of interest:

Modeling: What are the mechanisms that produce a certain behavior or output on a given input, and how can they be described?

Simulation: What output is produced by the system for a given input?

Optimization: What input needs to be provided to the system in order to receive a desired or optimal output?

All three areas are represented by established scientific disciplines with their own methodologies and approaches.

This thesis focuses on optimization, i.e., the search for optimal solutions among a set of alternatives. The inputs to the system can be represented by decision variables of arbitrary domains, and all feasible combinations of these variables form the set of *decision alternatives*. The criteria to judge the different decision alternatives relate to the output they produce. Quantitative criteria are usually referred to as *objectives* (Kaliszewski 1994).

Decision situations often involve multiple criteria or objectives. In many cases, objectives are *incommensurable*, meaning they are not comparable with respect to magnitude and value, and *conflicting*, meaning that the different objectives cannot be arbitrarily improved without decreasing the value of another. This results in trade-offs between the objectives. Insight into such trade-offs (e.g. risk vs. profit, labor cost vs. social security, greenhouse gas emissions vs. nuclear waste) is often of crucial importance for decision making.

Due to the impossibility to achieve optimal values in all objectives simultaneously, multiple criteria decision making (MCDM) always involves a *choice problem*. The final solution represents a compromise between the different objectives depending on the *preferences* of the decision maker. The scientific area concerned with modeling and analyzing preference structures to formalize the choice process from usually small, explicit list of alternatives is called *multiattribute decision analysis* (Keeney and Raiffa 1976; Fandel and Spronk 1985).

Many decision problems, though, contain a large, possibly infinite number of decision alternatives. In such cases, it is impossible to explicitly compare all alternatives, and therefore the choice problem is accompanied by a *search problem* to filter promising (optimal) from unpromising (non-optimal) alternatives. Problems of this type are treated in the area called *multiobjective decision making* or *multiobjective optimization*. A typical classification of methods for multiobjective decision making is given by Hwang and Masud (1979), who distinguish four classes according to when the decision maker's preferences enter the formal decision making process:

1. No articulation of preference information (only search),

2. A priori articulation of preference information (choice before search),

3. Progressive articulation of preference information (integration of search and choice), or

4. A posteriori articulation of preference information (search before choice).

The first class assumes that a global criterion is available to guide the search without making use of any decision maker preferences. In the second class, the different objectives are aggregated into one meta-objective so that traditional single-objective optimization methods can be applied. Here, preferences come into play at the beginning to define the aggregation via weights, aspiration levels, etc. The progressive methods allow the decision maker to interactively specify and modify preferences during the search. The methods in the fourth class assume that the search is conducted first, and a set of promising alternatives is generated before the decision maker can make the choice, possibly by resorting to appropriate methods from multiattribute decision analysis.

This thesis concentrates on the fourth class of methods. The aim is to approximate the set of optimal decision alternatives algorithmically. After this search process, the generated set of alternatives can serve as an input to the

decision maker, who can select a final, single alternative according to her preferences. The latter part is subject of the field of multiattribute decision analysis and is outside the scope of this thesis.

1.2 Illustrative Example

The aim of this work can be illustrated by the example of a simple decision problem with two conflicting objectives: A set of four different items is given, each of which has a certain profit and a certain weight associated with it (see Figure 1).

Item 1	
Profit:	4
Weight:	1

Item 2	
Profit:	1
Weight:	1

Item 3	
Profit:	4
Weight:	2

Item 4	
Profit:	2
Weight:	2

Fig. 1: Collection of four items, each of which has a profit and a weight associated with it.

The task is to decide for each item whether we select it or not, and our objectives are to maximize the total profit and to minimize the total weight of our selection of items. Any of the 16 possible combinations of the four items represents a decision alternative and therefore a potential *solution* to the decision problem.

It can immediately be seen that our two objectives are conflicting, as we cannot find any collection that has minimal weight and maximal profit at the same time. We also note the trivial cases of choosing all items, which maximizes the profit, but also the weight, and choosing no item, which minimizes the weight and also the profit. But what about the solutions in between, the compromise solutions? Alternative $\{2, 3\}$, e.g., is certainly preferable to $\{1, 4\}$ as it yields both a higher profit and a lower weight (we can also say that $\{2, 3\}$ dominates $\{1, 4\}$), but how does it compare to $\{1, 2, 4\}$? Is there an unambiguous and objective way to tell which solutions are good and which are bad in the whole solution set, and if so, how can we find good or optimal solutions?

Figure 2 shows a diagram of the objective values of all possible combinations of items, i.e., of all decision alternatives. All these solutions can be classified according to whether or not there exists another solution that is superior to it with respect to one objective and not worse regarding the other. If this is not the case, the respective solution is termed *Pareto-optimal* (Pareto 1896). It is clear that no reasonable decision maker would opt for a non Pareto-optimal solution, regardless of her preferences. Which specific Pareto-optimal solution will be chosen is subjective and depends on the preferences of the decision maker.

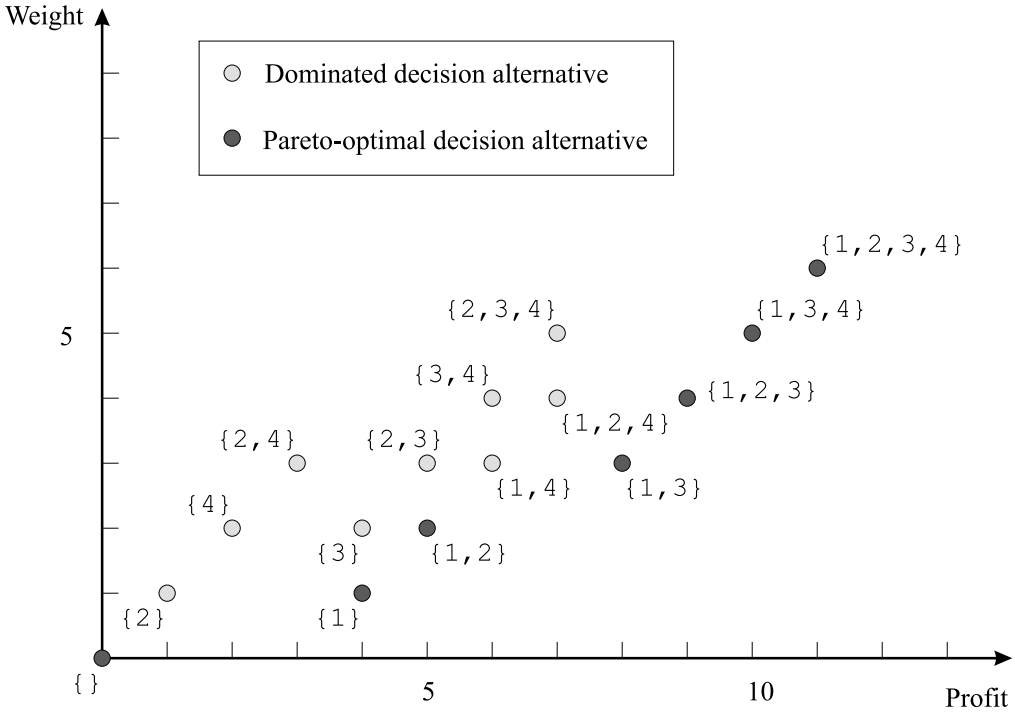


Fig. 2: Objective values of all decision alternatives.

The above example is a multiobjective formulation of the 0/1 knapsack problem, a well-known and well-studied combinatorial optimization problem (Martello and Toth 1990). The set of Pareto-optimal solutions in the example can easily be determined by enumerating all possible decision alternatives. However, this approach is usually not feasible for larger instances as the decision space grows exponentially with the number of decision variables. The task of generating the Pareto-optimal set therefore becomes a challenging algorithmic problem. This especially holds for problems where analytical methods are not applicable, because the underlying mapping of decision alternatives to objective values is too complex or even unknown.

1.3 Problem Statement

The general problem considered in this work is the following. Given is a definition of the *decision space*, which is the set of all possible decision alternatives, and vector valued *objective function*, which maps each decision alternative to a vector of objective values (see Figure 3). In our example, the decision space X is the set of all subsets of items, and the objective function f assigns each element x out of X a value pair, the sum of the profits and the sum of the weights of the items included in x. In many practical problems, however, we do not have access to the definition or mathematical description of the objective func-

tion. Therefore, the objective function has to be regarded as a "black box". It is only allowed to evaluate the objective function for a finite number of arbitrarily chosen decision alternatives. This is of course a more restrictive setting than the above example of the knapsack problem, where both, the structure (a simple sum over the items) and the parameters (the individual weights and profit values) of the objective function is known. Nevertheless, it is a relevant scenario arising in many applications, especially when objective values are determined by simulation models, or even by experiments in the physical world.

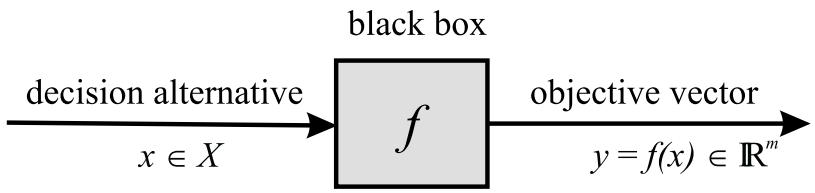


Fig. 3: Schematic view of black box optimization, where elements of the decision space X need to be determined such that the components of the corresponding objective vector are optimal under the mapping $f : X \mapsto \mathbb{R}^m$, $m \in \mathbb{N}$.

Our task in the above problem is to determine the set of Pareto-optimal decision alternatives. In many cases, especially when the decision space is very large or the objective function is very complex, this aim might be difficult or even impossible to achieve in reasonable time. In such cases, the aim is rephrased to finding at least a good *approximation* of the Pareto-optimal set, i.e., a set of solutions that are in some sense close to optimal and represent the true Pareto-optimal set well. The question of what constitutes a good approximation of the Pareto set forms a central part of this thesis and will be discussed at a later stage. In the knapsack example one could think of a possible approximation using four solutions {},{1}, {1, 3}, and {1, 2, 3, 4}.

Not many algorithms are applicable in such an information restricted scenario of black box optimization. One option, which has become very popular in recent years, is to use evolutionary algorithms, because these algorithms are formulated independent of the objective function. Evolutionary algorithms (EAs) are randomized search algorithms inspired by principles of natural evolution (Bäck et al. 1997). Decision alternatives coded as *individuals* undergo cycles of variation and selection in order to be steadily improved, so that optimal or near optimal solutions are eventually found. Typically, many solutions collected in *populations* are processed simultaneously so that different optimal solutions can be found in parallel. This makes evolutionary algorithms an attractive candidate for solving multiobjective optimization problems, where different Pareto-optimal solutions are sought.

The class of randomized search algorithms is very broad. It ranges from pure random search or Monte Carlo methods to stochastic local search methods. In between these extremes, and more recently developed, are methods such as simulated annealing (Kirkpatrick et al. 1983), tabu search (Glover and Laguna

1997) and many other heuristics and meta-heuristics (Ehrgott and Gandibleux 2000). Common features of randomized search algorithms are that they work iteratively, which means that new search points are generated and evaluated in discrete time steps, and they explicitly use randomness during some of their operations. The algorithms considered in this thesis are problem-independent algorithms in the sense that they do not make any assumptions about the objective function, such as linearity, differentiability, etc. They work with any kind of decision space, provided that appropriate search operators are defined. In contrast to many other, traditional and more specialized methods, they only require to evaluate the objective function at arbitrarily chosen search points and are therefore well suited for our black box optimization scenario. From a conceptual point of view, a problem-independent randomized search algorithm can be seen as a general recipe of how a search space is scanned, and searched through, in order to find optimal solutions.

Randomized search algorithms typically use mechanisms that are simple to describe and implement, but exhibit a complex behavior, which is hard to analyze theoretically. Consequently, a lot of empirical knowledge and successful applications are available, but much less rigorous theoretical results about their efficiency exist. Nevertheless, theoretical work is important for making more precise statements about their performance, and to understand the dynamics of these algorithms. In particular, a theoretical analysis addresses following essential questions:

- Is a particular algorithm able to find a set of different solutions to a given multiobjective optimization problem?
- How good are these solution sets?
- How much time is necessary to find the solution sets?

1.4 Thesis Contributions and Overview

This thesis contributes towards the understanding of evolutionary algorithms for multiobjective optimization problems with respect to the above questions. The work is structured into four chapters.

Fundamentals of evolutionary multiobjective optimization algorithms.
The vast variety of different instances and implementations of randomized multiobjective optimization algorithms makes it increasingly difficult to derive general results about their performance, similarities and differences. The aim of this chapter is to present the basic concepts of multiobjective optimization and to expose the essential differences between multiobjective evolutionary algorithms and their single-objective counterparts. These differences are mainly related to the question how

single decision alternatives as well as sets of decision alternatives can be compared under the presence of multiple objectives. Comparing and grading single decision alternatives plays a crucial role in the selection of search points, and its algorithmic realization in evolutionary algorithms is through various so-called *fitness assignment* schemes. Comparing and grading sets of decision alternatives is important for comparing the final outcome of different algorithms, which is facilitated by *quality indicators* and therefore plays a major role in empirical *performance assessment* of multiobjective optimization algorithms. The analysis is based on the decision-theoretic concepts of ordered sets. This novel approach allows (i) to investigate both issues using the same mathematical framework, (ii) to define and prove properties of several commonly used fitness assignment schemes as well as various popular quality indicators, and (iii) to derive new classification schemes.

Limit behavior and global convergence. The limit behavior of a randomized search algorithm means its dynamics under the assumption that unlimited time resources are available, i.e., the algorithm is allowed to run for an infinite amount of time. The analysis of the limit behavior allows to make statements about what a randomized search algorithm can maximally achieve. In case of multiobjective optimization, an algorithm should be able to find a set of (approximate) Pareto-optimal solutions at least in the limit of infinite time resources. Though this is almost trivial in the case of infinite memory (simply by keeping track of all non-dominated solutions found so far), the case of finite memory has not been solved yet. Consider, for example, an algorithm with a maximum memory size of two applied to the above knapsack problem. How would such an algorithm decide which solution to keep and which to discard in each iteration? To approach this question, we propose a notion of representative and approximate Pareto sets and investigate the convergence properties of different algorithms with respect to these solution sets. It is shown how and why many common multiobjective evolutionary algorithms fail to maintain a representative set of Pareto-optimal solutions due to the decisions they make during the selection steps, a deficit that has not been realized before. To overcome this fundamental limitation, we propose first selection operators with the desired properties of (i) working with a finite memory and (ii) guaranteeing the convergence of the generated sequence of solution sets. The new selection operators are problem independent and can be used within any randomized search algorithm.

Running time analysis. In addition to global convergence in the limit, we are also interested in a quantitative analysis, specifically the expected running time for a given class of problems and the success probability for a given optimization time. For the knapsack problem, such an analysis would, for instance, consider the number of time steps a certain algorithm needs on average to find a specific collection of Pareto-optimal solutions. To judge

the scalability of the algorithm, one would then investigate different problem sizes by varying the number of items, and try to express the expected running time as a function of the number of items. So far, such results were only available for the single objective case. Focusing on the optimization of pseudo-Boolean functions, this work presents running time results for different multiobjective evolutionary algorithms for different problem scenarios. In particular,

- two pseudo-Boolean model problems are introduced, which are scalable in the number of decision variables and number of objectives,
- simple individual-based and population-based multiobjective EAs are defined, and
- methods to analyze population-based evolutionary algorithms in a multiobjective framework are presented, which lead to
- complexity results in terms of bounds of the expected running time of the different algorithms.

These fundamental contributions represent the first running time results in the multiobjective case. A further motivation for this analysis is to investigate the benefit of the use of a population in solving multiobjective problems: is a population-based algorithm searching concurrently for all optimal solutions in a single run more efficient, or are multiple, separate runs of a single-objective optimizer searching for different optimal solutions a better strategy?

Applications. A main motivation for the design and analysis of randomized search algorithms is to apply them to real-world applications, and multiobjective optimization problems surfaced in many areas. Typically, such problems contain complex search spaces, where the quality of decision alternatives is evaluated through simulation using sophisticated computer models. This usually prevents conventional optimization techniques from being applicable, and randomized search algorithms are used because they are independent of the problem representation. Using on a generic multiobjective design procedure based on evolutionary algorithms, this chapter presents three case studies from automotive engineering: the design space exploration of road trains, parameter optimization of adaptive cruise control systems for trucks, and model fitting for a vehicle dynamics simulation. The aim of this chapter is show the successful application of this procedure for real-world examples and to exemplify some of the theoretical concepts devised in the thesis.

2

Multiobjective Optimization with Evolutionary Algorithms

This chapter addresses the fundamentals of randomized search algorithms, especially evolutionary algorithms, for multiobjective optimization. The investigations are based on decision theory. Decision theory is typically used as a mathematical framework to formally define the multiobjective optimization problem. Moreover, it also proves to be a powerful tool to derive new results regarding important aspects of evolutionary multiobjective optimization, namely the selection and the performance assessment problem. After introducing the decision-theoretic foundations of multiobjective optimization, a general model for randomized search algorithms is presented. The model serves as a baseline, and as a common framework for the investigations of multiobjective evolutionary algorithms throughout this thesis. Then, two specific issues are investigated, where the peculiarities of multiobjective optimization make the algorithms differ essentially from their single-objective counterparts: the ranking and selection of solutions within the algorithms and the performance assessment of algorithms. Both topics will be discussed from the decision-theoretic concept of preference orderings, which allows for a mathematically rigorous analysis and categorization of the existing techniques.

2.1 Multiobjective Optimization

The task in multiobjective optimization is to find solutions to problems of the form

$$\begin{array}{ll} \text{maximize} & f(x) = (f_1(x),\ f_2(x), \ldots,\ f_m(x)) \\ \text{subject to} & x \in X, \end{array} \tag{2.1}$$

where x is called the *decision space* or search space and its elements $x \in X$ *decision alternatives* or potential *solutions*. There is generally no further restriction on the domain of the decision alternatives.

The functional relationship between the decision alternatives and the decision criteria is established through the *objective function* $f : X \mapsto \mathbb{R}^m$, $m \in \mathbb{N}$. The *objective space* $Y = f(X)$ is the image of the decision space X under the objective function f. An element $y \in Y$ is called *objective vector* and its components *objective values*.

In a decision making and optimization context, the notion of optimality depends on how decision alternatives are compared and ordered, i.e., on the *preference structure* of the decision maker. Often, preferences are assumed tacitly and not made explicit because they are obvious. Consider for example the special case of $m = 1$, the well-known case of single-objective optimization. Here, the meaning of maximization is unambiguous and intuitively clear. The decision alternatives can be totally ordered according to their (single) objective values and the optimal solution is the one with the largest objective value. For $m \geq 2$, however, the situation changes and the notion of maximality is not so apparent. From both a practical and a theoretical standpoint, it is therefore necessary to define the notion of optimality by means of preference structures. Besides formally clarifying what is actually understood by multiobjective optimization, it provides the decision-theoretic foundation on which the investigations in the subsequent sections rely.

The starting point is the basic assumption that a decision maker, who is confronted with any pair of decision alternatives, either

- clearly prefers one over the other,
- feels indifferent about them, or
- considers the two alternatives as incomparable.

These and other preference relations will then be formally defined and their properties derived. This allows to characterize optimal solutions as the maximal elements of decision space under the respective preference ordering. Finally, these concepts are related back to the multiobjective optimization problem to clearly specify the meaning of maximization. The following definition summarizes basic properties of binary relations used in this chapter.

Def. 1: **(Properties of binary relations)** *Let A be a set. A* binary relation $R \subset A \times A$ *is called*

reflexive	*if* $\forall a \in A : (a, a) \in R$
irreflexive	*if* $\forall a \in A : (a, a) \notin R$
symmetric	*if* $\forall a, b \in A : (a, b) \in R \Rightarrow (b, a) \in R$
asymmetric	*if* $\forall a, b \in A : (a, b) \in R \Rightarrow (b, a) \notin R$
antisymmetric	*if* $\forall a, b \in A : ((a, b) \in R \wedge (b, a) \in R) \Rightarrow a = b$
transitive	*if* $\forall a, b, c \in A : ((a, b) \in R \wedge (b, c) \in R) \Rightarrow (a, c) \in R$
negatively transitive	*if* $\forall a, b, c \in A : ((a, b) \notin R \wedge (b, c) \notin R) \Rightarrow (a, c) \notin R$
connected	*if* $\forall a, b \in A : ((a, b) \in R \vee (b, a) \in R)$

For brevity, the notation aRb will be used to express $(a, b) \in R$. Now, the above comparison relations can be formalized to construct a preference structure.

Def. 2: **(Preference structure)** *A preference structure on a set A is a partition of $A \times A$ into the three binary relations $\succ$, $\sim$, $\|$ such that for $a, b \in A$*

- $a \succ b \Leftrightarrow$ *a is preferable to b (a dominates b),*
- $a \sim b \Leftrightarrow$ *a and b are equally preferable,*
- $a \| b \Leftrightarrow$ *a and b are incomparable,*

where the preference relation $\succ$ is asymmetric, the indifference relation $\sim$ is reflexive and symmetric and the incomparability relation $\|$ is irreflexive and symmetric.

A preference structure can uniquely be represented by the weak preference relation $\succeq := \succ \cup \sim$, since

- $a \succ b \Leftrightarrow (a \succeq b \wedge b \not\succeq a)$,
- $a \sim b \Leftrightarrow (a \succeq b \wedge b \succeq a)$,
- $a \| b \Leftrightarrow (a \not\succeq b \wedge b \not\succeq a)$.

Though a given preference structure allows to describe the relation of any two decision alternatives, it is still too general for our purpose as it does not preclude cycles of the preference relation. For example, a situation where $a \succ b$ and $b \succ c$, but also $c \succ a$ holds, is not useful in an optimization context. Thus, the weak preference relation is additionally required to be transitive. A transitive preference relation is called a *preorder* or quasi-order. For convenience, the following definition also summarizes the other types of order relations used in the course of this work.

Def. 3: **(Order relations)** *A binary relation is called*

- *a* preorder *if it is reflexive and transitive,*
- *a* total preorder *if it is reflexive, transitive and connected,*
- *a* total order *if it is an antisymmetric total preorder*

- *a* partial order *if it is reflexive, transitive and antisymmetric,*
- *a* strict partial order *if it is asymmetric and transitive.*
- *a* strict weak order *if it is asymmetric and negatively transitive.*

In the following, the decision space, together with the weak preference relation of the decision maker, is regarded as a preordered set. In a preordered set, every element might have elements that it precedes and elements that it succeeds according to the applied order relation. These elements can be summarized in special sets called the upper and lower section.

Def. 4: **(Upper and lower sections)** *Let* $(A, \succeq)$ *be a preordered set. With each* $a \in A$ *we associate the following sets:*

- *the* upper section $[a, \rightarrow) \equiv \{x \in A : x \succeq a\}$,
- *the* strict upper section $(a, \rightarrow) \equiv \{x \in A : x \succ a\}$,
- *the* lower section $(\leftarrow, a] \equiv \{x \in A : a \succeq x\}$,
- *the* strict lower section $(\leftarrow, a) \equiv \{x \in A : a \succ x\}$.

With this semantic, the maximal elements of the preordered set $(A, \succeq)$ are those which have an empty strict upper section:

Def. 5: **(Maximal set)** *Let* $(A, \succeq)$ *be a preordered set. The* maximal set A^* *of* $(A, \succeq)$ *is defined as*

$$A^* := \{a \in A : (a, \rightarrow) = \emptyset\}$$

The elements of A^* *are called* maximal elements.

The maximal elements are of special interest for the decision maker because no other decision alternative is preferable to it. In this respect, they represent *optimal* decision alternatives. For a multiobjective optimization problem (2.1), we are now in a position to define the meaning of maximization by specifying the preference ordering.

Def. 6: **(Order relation for decision alternatives and objective vectors)** *Let a multiobjective optimization problem be given by the decision space* X *and the objective function* $f : X \mapsto Y \subseteq \mathbb{R}^m, m \in \mathbb{N}$. *For decision alternatives* $a, b \in X$ *the order relation* $\succeq_f$ *is defined as*

$$a \succeq_f b :\Leftrightarrow f(a) \succeq f(b),$$

where $\succeq \subseteq Y \times Y$ *is defined as*

$$f(a) \succeq f(b) :\Leftrightarrow (f_i(a) \geq f_i(b)\ \forall i \in \{1, \ldots, m\}).$$

In this special case, the maximal set X^* is called *Pareto-optimal set*, or Pareto set in short, and its elements Pareto-optimal decision alternatives or Pareto-optimal solutions. Its image in objective space, $Y^* := f(X^*)$, is called Pareto front and its elements Pareto-optimal objective vectors.

This thesis treats multiobjective optimization problems under the concept of Pareto optimality. Nevertheless, other preference orderings give rise to different notions of optimality; for an overview and discussion the reader is referred to the literature (Ehrgott 2000). In the following, it makes use of the preference ordering in the above sense unless otherwise stated and skip the index f of $\succeq_f$ for brevity. The resulting relations $\succeq$ and $\succ$ will also be termed "weak dominance" and "dominance" relations such that a is said to dominate b if and only if $a \succ b$. From their definition the following equivalences are immediate:

Cor. 1: *Let X be a decision space and $f : X \mapsto Y \subseteq \mathbb{R}^m, m \in \mathbb{N}$. Then for all decision alternatives $a, b \in X$*

$$a \sim b \quad \Leftrightarrow \quad (f_i(a) = f_i(b)\ \forall i \in \{1, \ldots, m\})$$
$$a \succ b \quad \Leftrightarrow \quad (f_i(a) \geq f_i(b)\ \forall i \in \{1, \ldots, m\} \wedge \exists i \in \{1, \ldots, m\} : f_i(a) > f_i(b)).$$

Furthermore, $(X, \succeq)$ is a preordered set and $(Y, \succeq)$ is a partially ordered set, while $(X, \succ)$ and $(Y, \succ)$ are strictly partially ordered sets.

The aim of the multiobjective optimization algorithms treated in this thesis is thus to identify X^*, the maximal set of X, and Y^*, the image of X^* in objective space. In cases where it is very difficult the aim is to approximate X^* with respect to its image in objective space, i.e., to find a set $X' \subseteq X$ such that its image $f(X')$ is a good approximation of Y^*. In this respect, one goal of this thesis is to give a useful, formal definition of Pareto set approximation; this issue will be discussed in Chapter 3.

2.2 Evolutionary Algorithms

This section provides a model of evolutionary algorithms as a special instances of randomized search algorithms. The purpose of this exposition is two-fold. After formalization of the multiobjective optimization problem in the previous section, it is important to define the terminology of the optimization algorithms used in this thesis. A second aim is to highlight those components of evolutionary algorithm that are especially important in the context of multiobjective optimization, and to concretize and locate the open questions for the subsequent investigations. Without sacrificing preciseness, the model shall be as general as possible, such that the derived results are valid for a broad class of algorithms.

A randomized search algorithm starts with choosing an initial search point $x^{(0)} \in X$ and computes its objective vector $f(x^{(0)})$. It then proceeds in discrete time steps $t \in \mathbb{N}$ and produces new search points $x^{(t)}$ depending

on the previously generated ones $x^{(0)}, \ldots, x^{(t-1)}$ and their objective values $f(x^{(0)}), \ldots, f(x^{(t-1)})$, until a termination criterion is fulfilled. For efficiency reasons and because of limited resources of real computers, randomized search algorithms have to work with finite memory. In particular, it cannot be expected to be able to store all generated search points, but only a subset of bounded size. Besides a sample of the generated solutions, however, they are allowed to accumulate a bounded amount of additional state information. Such state information is often used as a compact representation of the knowledge gained during the search process to hopefully generate better solutions in the future. Evolutionary algorithms implement this memory constraint by the concept of *population*. A population is a collection of a finite number of decision alternatives represented as *individuals*.

Algorithm 1 A Conceptual Evolutionary Algorithm

1: $t \leftarrow 0$
2: $A^{(0)} := \emptyset$
3: **while** $\mathsf{terminate}(A^{(t)}, t) = \text{false}$ **do**
4: $\quad t \leftarrow t + 1$
5: $\quad B^{(t)} \leftarrow \mathsf{generate}(A^{(t-1)})$ {create new offspring population}
6: $\quad A^{(t)} \leftarrow \mathsf{select}(A^{(t-1)}, B^{(t)})$ {update parent population}
7: **end while**
8: $\mathsf{output}(A^{(t)*})$

A conceptual evolutionary algorithm is depicted above as Algorithm 1. Evolutionary algorithms typically summarize many little time steps into a larger one called *generation*. In each generation t, a whole new population B called *offspring population* or simply offspring is generated en bloc from the currently stored *parent population* A or parents. At the core of the algorithm are the two operators generate and select. The operator generate creates a new offspring population based on the parent population. The parent and offspring population is then passed to the select operator to decide which individuals to maintain for the parent population of the next iteration and which individuals to discard. In evolutionary algorithms, the generate operator is usually given by the repeated application of the variation operators called mutation and recombination. While mutation is a unary operator creating one offspring individual on the basis of a single parent individual, the recombination operator is able to create multiple offspring from multiple parents. The details for the generation process are not relevant here.

The question arises whether the reduction of the algorithm to a simple alternation the two operators generate and select is an appropriate model for the subsequent investigations. Alternatively, one could consider both directions of either a more detailed or a more compact representation. A more detailed model is for example used in Bäck (1996). It requires to specify the operators and their interplay. Thereby it necessarily restricts the number of possible instances, and the loss of generality can only partially be counteracted by introducing many

algorithmic parameters. More compact formulations were proposed in efforts to derive a mathematically analyzable, stochastic models of evolutionary algorithms. They typically combine the operators into an overall transition operator $\mathsf{T} = \mathsf{select} \circ \mathsf{generate}$, which maps the state of the algorithm, represented by the currently stored solutions $A^{(t)}$, to its state $A^{(t+1)}$ at time $t+1$. In randomized search algorithms, T is typically stochastic as both the select and the generate operators can contain randomized operations. The equation

$$A^{(t+1)} = \mathsf{T}(A^{(t)})$$

suggests that randomized search algorithm can be modeled as Markov chains. For the case of evolutionary algorithms see, e.g., Rudolph (1997a, 1998b), Vose (1999), or He and Yao (2002). The main difficulty in using the Markov chain approach is, however, to bound the state space and to derive a compact and tractable formulation of the transition matrix. This thesis makes use of the above algorithmic formulation of evolutionary algorithms, which is sufficient for the remainder of this chapter as well as the investigations of the limit behavior in Chapter 3. The details of the operators are only specified where necessary, that is in the running time analysis of concrete algorithms in Chapter 4 as well as in the application case studies of Chapter 5.

This section concludes with a short discussion of the role of the generate used in the above conceptual algorithm before the select operator is treated in more detail in a separate section. The generate operator is used to create new search points. In doing so, it normally makes use of the information gathered so far during the search process and accumulated in the population A. Initially, the set A is empty because no information has been collected yet. Therefore, the initial search points are usually drawn at random from the search space. The set A normally contains a sample of the better search points visited so far. Many algorithms work under the assumption that other promising solutions are in some respect similar to already known good ones and place the next search points in their vicinity. Examples for this strategy are the use of a neighborhood in randomized local search and simulated annealing or the mutation operator in evolutionary algorithms. Another idea is to exploit the similarities among multiple solutions as does the recombination operator in evolutionary algorithms. Other information about the search history than the actual found search points can be gathered by the algorithm as well, either in global variables or by augmenting the representation of the individuals using additional strategy parameters. Common strategies are, to record the successful search steps in order to proceed in the same direction, as in the covariance matrix adaptation evolution strategies, or to avoid cycling and returning to already visited areas, as in tabu search. A further example of such additional information will be presented within the FEMO and the GEMO algorithm in Chapter 4. The generate operator is generally objective function independent in the sense that objective function values are not explicitly used. In this respect, there is no difference between single objective and multiobjective optimization. The investigations conducted in this thesis abstract from the details of the generate operator and focus on issues

where randomized search algorithms for multiobjective optimization are different from their single-objective counterparts.

2.3 Selection under Multiple Objectives

The main role of the selection operator in evolutionary algorithms is to impose a direction on the search process, while the creation and variation of search points is usually undertaken in an undirected way. The search direction should be in accordance with the decision maker's preference structure so that better decision alternatives are favored over worse. Thus, it is important to investigate how selection operators and preference structures are interrelated. This allows for a more precise characterization and classification of different selection schemes.

2.3.1 Selection based on Fitness Functions

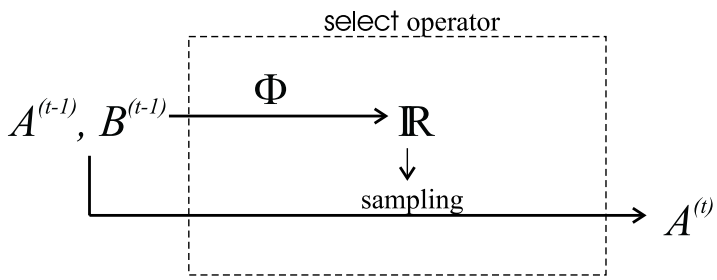


Fig. 4: Division of the selection operator into a fitness assignment process and a sampling process. The sampling process makes use of the scalar fitness values defined by the fitness function Φ.

The task of the selection operator is to discern better search points from worse. It usually works by first grading or ranking the different alternatives and then performing the actual selection operation to determine a new set $A^{(t)}$ out of elements from $A^{(t-1)}$ and $B^{(t-1)}$. The grading is achieved by assigning each element $a \in A \cup B$ a scalar *fitness* value $\Phi(a)$ as a measure of its quality or utility [1] within the set of currently stored solutions. The mapping $\Phi : A \mapsto \mathbb{R}$ is not restricted to being a pure function of the decision alternative, or its objective function values, but is a relative measure depending on the other search points in A. The sampling process of choosing elements from $A^{(t-1)}$ and $B^{(t-1)}$ to be included in $A^{(t)}$ is then performed using the previously computed Φ values. This two-step process is conceptually visualized in Figure 4.

[1] Though the word "utility" has been used here in an informal and intuitive way, the connection to utility theory is obvious and, as the subsequent considerations reveal, sensible: the function Φ can be easily associated with a local approximation of the generally unknown utility function of the decision maker.

The sampling itself, which can be done in different ways (deterministically or randomly, with or without replacement), is of no further relevance here as there are no fundamental differences from the single-objective case. Comprehensive investigations of different sampling schemes can be found, e.g., in Hancock (1997) and Blickle and Thiele (1996)[2].

The focus of this section is therefore on the fitness assignment. The central question is: to what extent do fitness functions respect the given preference ordering of the decision alternatives when they transform this preorder into the canonical order on the set of real numbers? It will be shown that information will necessarily be lost during this reduction, and it appears important to minimize these losses. The following example illustrates this issue.

Ex. 1: **(Weighted sum fitness function)** *Consider the fitness function*

$$\Phi_{\text{WeightedSum}} := \sum_{i=1}^{m} w_i \cdot f_i(x),\ w_i \in \mathbb{R},$$

applied to the following set A of decision alternatives from the knapsack problem example using weights $w_1 = 2$ and $w_2 = -1$. The resulting fitness values are

$$\begin{array}{lll} A = \{a, b, c, d\} & & \\ a = \{\} & f(a) = (0, 0) & \Phi_{\text{WeightedSum}}(a) = 0 \\ b = \{1\} & f(b) = (4, 1) & \Phi_{\text{WeightedSum}}(b) = 7 \\ c = \{3\} & f(c) = (4, 2) & \Phi_{\text{WeightedSum}}(a) = 6 \\ d = \{1, 2, 4\} & f(d) = (7, 4) & \Phi_{\text{WeightedSum}}(a) = 10 \end{array}$$

It appears reasonable that the fitness of b is better than the fitness of c as b dominates c. It might be unsatisfactory, though, that the fitness function evaluates d, a non Pareto-optimal solution, better than a and b, both Pareto-optimal solutions. Of course, at this stage the algorithm cannot know that another Pareto-optimal solution exists that dominates d. But this excuse fails in the case of ranking c better than a, because the algorithm knows with b already a solution that dominates c and could thereby clearly identify c as non-optimal.

We want to formalize the issues raised above. To start with, the notion of order morphisms is introduced as the key concept to analyze fitness functions mathematically.

Def. 7: **(Order morphisms)** *Let R be a binary relation on a set A and S a binary relation on a set B. A function $u : A \mapsto B$ is called*

- *an* order homomorphism *if* $\forall a, b \in A : aRb \Rightarrow u(a)Su(b)$

[2] Also the common distinction of rank-based and fitness-based selection schemes is not important here, as the elements can always be totally ordered once the scalar Φ values are derived. The reverse way is also possible by defining the Φ values as the ranks of the elements in the totally ordered set A.

- *an* order isomorphism *if* $\forall a, b \in A : aRb \Leftrightarrow u(a)Su(b)$

For the special case of B being the set of real numbers $\mathbb{R}$, an order isomorphism u is also called a *representation* of the order relation $\succeq$. Ideally, one would therefore look for fitness functions being a representation of the dominance relation such that S is a canonical order on the reals. This would mean that that there is a one-to-one correspondence between the dominance relation of the decision alternatives and, e.g., the greater-than relation on their fitness values. Unfortunately, such a representations does not exist as a consequence of the following theorem.

Th. 1: **(Bridges and Mehta 1995, pp. 8 - 9)** *Let A be a countable set and $\succeq$ and $\succ$ binary relations on X. A necessary and sufficient condition for the existence of a real-valued function u on A so that*

$$a \succeq b \Leftrightarrow u(a) \geq u(b)$$

is that $\succeq$ is a total preorder and for

$$a \succ b \Leftrightarrow u(a) > u(b)$$

is that $\succ$ is a strict weak order.

For our fitness function Φ it is therefore impossible to be an order representation in the above sense, because the weak dominance $\succeq$ is usually not connected and the dominance relation $\succ$ is not negatively transitive (see Definition 3).

To be consistent with the preference structure of the decision maker, a fitness function should nevertheless be at least an order homomorphism into $\mathbb{R}$. This means that a fitness assignment never *contradicts* the preference relation by assigning a solution a a better fitness value than b though b is preferred to a. Such order homomorphisms are easy to construct, for example by simply assigning each $a \in A$ the size of its lower section. Likewise, the weighted sum fitness function of Example 1 is an order homomorphism.

As a result of these considerations, it can be stated that being an order homomorphism appears to be a minimum requirement for any fitness function. On the other hand, the ideal case of an order isomorphism does not exist. For a more precise characterization of fitness functions, we will investigate next which additional properties are possible and desirable, and classify existing fitness assignment strategies accordingly. This is important as the choice of the fitness function determines the search direction and thereby strongly influences the search process and the success of the optimization algorithm.

2.3.2 Properties of Fitness Functions

The aim of this section is to propose a new mathematical classification scheme for fitness functions and describe their properties based on decision theory. After introducing additional concepts that are useful for a compact notation, we

briefly review the commonly used algorithmic classification of fitness assignment strategies. Then, two additional new mathematical properties are introduced, which extend the concept of order morphisms and allow for a more precise characterization.

To facilitate the description and analysis of fitness functions, we make use of the following graph-based coding of the order relation. An ordered set $(A, \succeq)$ can be regarded as a directed graph, where A is the set of nodes and $\succeq$ the set of edges. There exists an edge from a to b if and only if $a \succeq b$. If $\succeq$ is the dominance relation, the corresponding graph is called a dominance graph. In case of $\succ$ being a strict partial order, a more compact formulation can be achieved by retaining only edges between direct successors:

Def. 8: **(Cover relation and minimal dominance graph)** *Given a strict partial order $\succ$ on a set A, the cover relation $\succ_c$ is defined for elements $a, b \in A$ so that*

$$a \succ_c b \Leftrightarrow (a \succ b \wedge \nexists c \text{ so that } a \succ c \succ b)$$

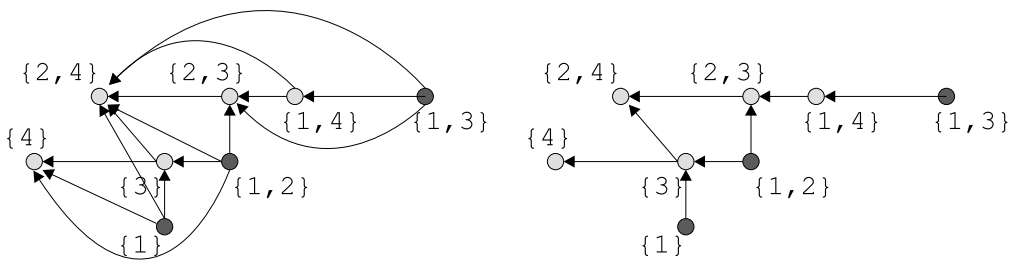


Fig. 5: Dominance graph of some of the decision alternatives of the knapsack problem example (left) and the resulting minimal dominance graph (right). The maximal elements of A are depicted by black circles, all other elements by grey circles.

The resulting graph $(A, \succ_c)$ is called a minimal dominance graph. It is the transitive reduction of $(A, \succ)$, and its pictorial representation is also known as a *Hasse diagram* (Skiena 1990, p.206f). Figure 5 shows an example of a dominance graph and the resulting minimal dominance graph. As addition notation, it is helpful to define the *dimension* and the *length* of preordered sets.

Def. 9: **(Dimension and length of preordered sets)** *Let $(A, \succeq)$ be a preordered set. The dimension* $\dim(A, \succeq)$ *is the smallest $k \in \mathbb{N}$ such that $\succeq$ is the intersection of k total orders. The length* $\mathrm{len}(A, \succeq)$ *is defined as the length of the longest path in the associated minimal dominance graph $(A, \succ_c)$, i.e., the number of nodes on that path.*

In the above example, the dimension is 2 as the dominance relation is the intersection of the $\geq$ relation on f_1 and the $\leq$ relation on f_2. The length is 4, which can easily be verified in the figure.

Traditionally, fitness assignment and selection schemes for multiobjective evolutionary algorithms are classified from an algorithmic viewpoint. An overview and discussion of different multiobjective fitness assignment and selection schemes is given by Fonseca and Fleming (1995, 1997) and Horn (1997), who distinguish

Aggregation selection where Φ is a function of the different objective values of a solution independent of the other solutions;

Criterion selection where the solutions are ranked for each objective function; and

Pareto selection where the solutions are ranked based on the dominance relation.

The algorithmic classification is primarily descriptive and aimed at an easy comprehension and implementation of the methods. However, the choice of fitness assignment schemes should not be based only on algorithmic aspects. For a better understanding of their effect on the algorithm, it is necessary to know their mathematical properties. From a practical viewpoint this is important, for both choosing an existing scheme as well as for designing a new method. As discussed above, a minimal requirement is that Φ is order preserving, i.e., an order homomorphism regarding the preference orderings $\succeq$ and $\succ$. To capture the other issues raised in Example 1, two further properties concerning the treatment of the maximal elements are introduced, which can be stated as follows.

Def. 10: (Properties of fitness functions) *Let* $(A, \succeq)$ *be a preordered set. A function* $\Phi : A \mapsto \mathbb{R}$ *is called*

- max-preferring *if* $[a, \rightarrow) = \emptyset \Leftrightarrow (\nexists b \in A \setminus A^* : \Phi(b) \geq \Phi(a))$ *and*
- max-indifferent *if* $\forall a, b \in A^* : \Phi(a) = \Phi(b)$.

A fitness function is called *max-preferring*, if all maximal elements are ranked not worse that all other individuals, and *max-indifferent*, if all maximal elements are graded equally. In Example 1, the fitness function was neither max-preferring nor max-indifferent.

Equipped with a compact notation and a formalization of desirable properties, we are now in a position to give a formal definition and classification of fitness functions on preordered sets. Table 1 gives a summary of several, frequently used fitness functions together with their properties of being

(1) an order homomorphism of $(A, \succeq)$ into $(\mathbb{R}, \geq)$,

(2) an order homomorphism of $(A, \succ)$ into $(\mathbb{R}, >)$,

(3) max-preferring, and

(4) max-indifferent.

Name	$\Phi(a)$	(1)	(2)	(3)	(4)
dominance grade	$-\lvert(a, \rightarrow)\rvert$	*	*	*	*
dominance level	$-\mathrm{len}((a, \rightarrow))$	*	*	*	*
inverse dominance grade	$\lvert(\leftarrow, a)\rvert$	*	*		
inverse dominance level	$\mathrm{len}((\leftarrow, a))$	*	*		
SPEA	$-\begin{cases} \frac{(\leftarrow,a]}{\lvert A\rvert+1}, & \text{if } (a, \rightarrow) = \emptyset \\ 1 + \sum_{b\in[a,\rightarrow)} \frac{(\leftarrow,b]}{\lvert A\rvert+1}, & \text{else} \end{cases}$	*		*	
SPEA2	$-\sum_{b\in(a,\rightarrow)} \lvert(\leftarrow, b]\rvert$	*	*	*	*
rank sum	$\sum_{i=1}^{m} \lvert\{b \in A : f_i(a) > f_i(b)\}\rvert$	*	*		
weighted sum	$\sum_{i=1}^{m} w_i \cdot f_i(a), w_i \in \mathbb{R}^+$	*	(*)		
goal vector	$-\lvert\lvert g - f(a)\rvert\rvert_\alpha, g \in \mathbb{R}^m$	*	(*)		

Tab. 1: Fitness functions and their properties: (1) order homomorphism regarding $\succeq$, (2) order homomorphism regarding $\succ$, (3) max-preferring, (4) max-indifferent. If the fitness function $\Phi : A \mapsto \mathbb{R}$ has a property, the corresponding column is marked with an asterisk. The symbol (*) for the weighted sum means that the property holds if all weights are nonnegative and for the goal vector that it holds for all norms L_α with a finite α. The table is arranged in three sections, which correspond to the classification of Pareto selection (top section), criterion selection (middle section) and aggregation selection (bottom section).

It is easy to verify that all four properties are mutually independent. Thus, none of the proposed criteria to evaluate fitness assignment schemes is redundant.

The first observation is that all of the analyzed fitness functions are order preserving in the weakest sense, i.e., they represent one of the many possible order homomorphisms regarding the weak dominance relation $\succeq$. The induced rankings essentially differ only in how incomparable decision alternatives are ranked relatively to each other and guarantee that a preferable alternative is never ranked worse than a less preferable one. However, not all fitness functions guarantee that preferable alternatives are always ranked strictly better than less preferable ones. This is revealed by the fact that some functions are not order homomorphisms regarding $\succ$. This was, for instance, recognized as a deficit in the SPEA fitness assignment scheme and led to a subsequent modification for the SPEA2 algorithm (Zitzler et al. 2002).

As to max-preference and max-indifference, it is noteworthy that the approaches of criterion selection and aggregation selection usually do not possess these properties. On the other hand, the approaches based on Pareto selection do not fulfill these properties automatically, such as the SPEA as well as the inverse dominance level and inverse dominance grade approaches. In general, max-indifference appears to be a desirable property, but the lack of discriminating power between the maximal elements can also have drawbacks when it comes to bounding the size of the set A; this matter, and its influence on the convergence property of the algorithm, is subject of a detailed discussion in Chapter 3; it will be explained why a pure selection based on fitness functions is usually not sufficient to guarantee convergence.

2.4 Performance Assessment

An important aspect of an algorithm is its performance, i.e., how "good" it is in carrying out a specific task. For such a characterization one has to evaluate the quality of the result in relation to the resources that were needed to achieve it. In our setting, the result of a multiobjective optimization algorithm is a set of solutions, which makes is necessary to assess the quality of *sets* of solutions. This is a similar, but more involved question compared to the one discussed in the previous section, where the focus was to evaluate and compare only single solutions.

The main resource of optimization algorithms is computation time, which can be measured experimentally or calculated theoretically by counting the number of operations. If the task is to find the optimum, the performance can be easily defined as the time needed until the optimum is found. If, however, the optimal solution cannot be found and only approximated within the given time resources, performance can be defined as the achievable approximation quality for a given amount of computation time. These are two complementary views of the same concept, namely investigating the relationship of achieved solution quality and computation time.

In the special case of randomized search algorithm applied to multiobjective optimization problems, the quality assessment is not straightforward due to the following features.

Multi-variateness: Solutions of a multiobjective optimization algorithm are multi-variate, meaning that they represent vectors in objective space.

Multiple solutions: The output of a multiobjective optimization algorithm is a set of solutions rather than a single solution.

Stochasticity: The output of a randomized multiobjective optimization algorithm is a random variable (or a random process, if considered over time).

The multi-variateness of solutions and the resulting difficulties to grade and compare them under multiple objectives has been discussed thoroughly in the previous section. Moreover, when multiple solutions are combined in a set, the question of how to grade and compare different sets of solutions is even harder to answer. This issue will be discussed next. The third topic, how to deal with the stochastic nature of randomized search when assessing its performance, is not addressed in this thesis.

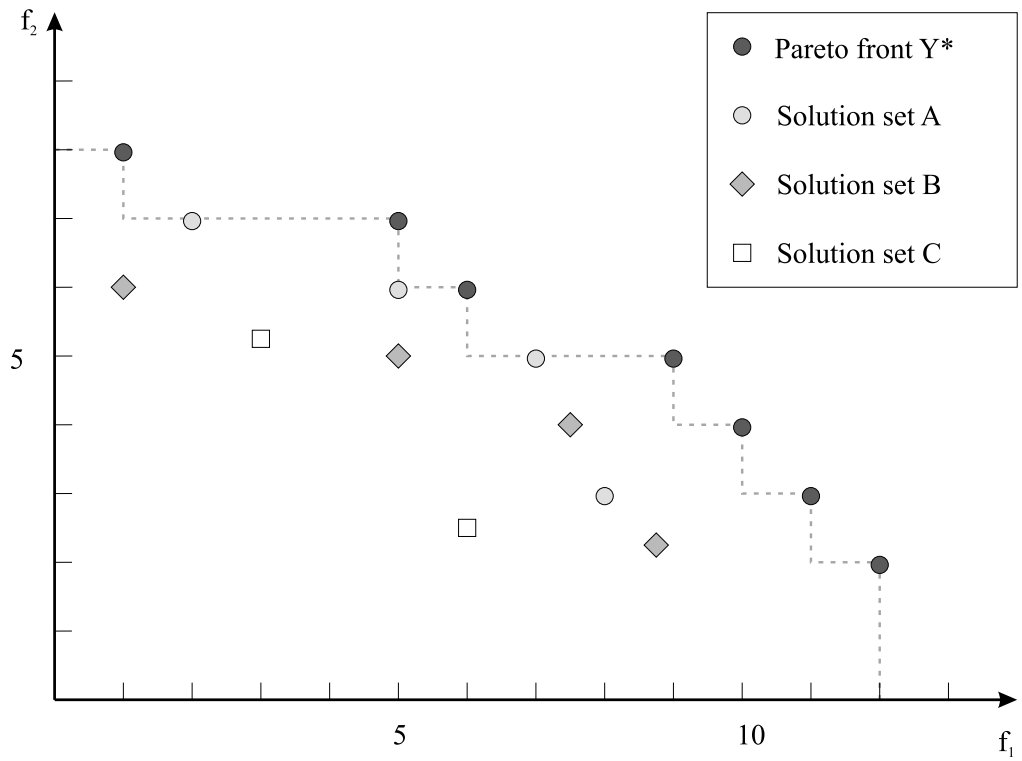


Fig. 6: Pareto front of a two-dimensional objective space and the images of three different solution sets.

The main question under concern in this section is: Given different sets of solutions produced by different algorithms, can we clearly say that one set is better than another? And if so, how much better is it? Figure 6 shows an example of a three solution sets A, B, C. It seems reasonable to say that A is better than C since all elements of C are dominated by some element of A. But how does A compare to B and B compare to C? And in addition, can we quantify, how much approximation set A is better than C?

Consequently, we are looking for indicators of the quality of solution sets. We proceed as follows. First, approximation sets are introduced to capture the output of a multiobjective optimization algorithm, and the notion of a quality indicator is formalized. Then, different types of absolute and relative quality indicators are investigated from a decision-theoretic point of view to answer the following questions:

- How well do absolute quality indicators reflect performance differences?
- To what extent are absolute quality indicators in accordance with the general preference relations on approximation sets?
- What additional information can be gained if several quality indicators are considered simultaneously?
- Are relative quality indicators a better choice?

2.4.1 Approximation Sets and Quality Indicators

The output of a randomized search algorithm at time t according to Algorithm 1 is the maximal set $A^{(t)*}$ of the currently stored search points $A^{(t)}$. The motivation behind concentrating on the maximal set is the assumption that dominated solutions are of no further interest for the decision maker anyway. The algorithm tries to approximate the Pareto set of the given multiobjective optimization problem as good as possible. The approximation quality, however, is measured in objective space. For notational convenience, the index t will be skipped and we carry out our investigations in objective space, i.e., we focus on $f(A^*)$. The output of an algorithm can hence be formalized as an approximation set as follow.

Def. 11: **(Approximation set)** *Let a multiobjective optimization problem be given by X and $f : X \mapsto \mathbb{R}^m$, $m \in \mathbb{N}$. A set $A \subseteq f(X)$ is called an approximation set if*

$$\forall a \in A \ \nexists b \in A : b \succ a$$

The set of all approximations sets is denoted as $\mathcal{A}$.

As with single decision alternatives or single objective vectors, order relations are needed to compare different approximation sets. The preference relation of single decision alternatives can be generalized to sets of decision alternatives.

Def. 12: *Let $\mathcal{A}$ be the set of all approximation sets and $A, B \in \mathcal{A}$. The binary relation $\succeq \subseteq \mathcal{A} \times \mathcal{A}$ is defined as*

$$A \succeq B :\Leftrightarrow \forall b \in B \ \exists a \in A : a \succeq b$$

Apparently, $(\mathcal{A}, \succeq)$ is a partially ordered set, since $\succeq$ is reflexive, transitive and antisymmetric. This weak dominance relation defines a preference structure according to Definition 2. The strict preference relation $\succ$, the indifference relation $\sim$, and the incomparability relation $\|$ can be derived analogously.

With these binary relations one can state qualitative differences between approximation sets, i.e., whether one approximation set is better, worse, or incomparable to another. The only maximal element of $(\mathcal{A}, \succeq)$ is $\mathcal{A}^* = f(X^*)$, the Pareto front itself. By using subsets of $\succeq$, even stronger preference relations can be defined on $\mathcal{A}$.

Def. 13: *Let $\mathcal{A}$ be the set of all approximation sets and $A, B \in \mathcal{A}$.*

- $A \dot{\succ} B \Leftrightarrow \forall b \in B\ \exists a \in A : a \succ b$ *(A pointwise dominates B)*
- $A \dot{>} B \Leftrightarrow \forall b \in B\ \exists a \in A : a > b$ *(A is pointwise larger than B)*

With $\dot{>} \subseteq \dot{\succ} \subseteq \succ \subseteq \succeq$, the implication hierarchy

$$A \dot{>} B \Rightarrow A \dot{\succ} B \Rightarrow A \succ B \Rightarrow A \succeq B$$

is obtained. In addition to this qualitative assessment, one would like to make more precise, quantitative statements when comparing approximation sets. Thus many authors have proposed to use functions that assign each approximation set a real number to reflect its quality, and to compare different algorithms based on these function values. This approach can be formalized by the notion of a quality indicator.

Def. 14: **(Quality indicator)** *An* absolute quality indicator *I is a function $I : \mathcal{A} \mapsto \mathbb{R}$, which assigns each approximation set a real value $I(A)$. A* relative quality indicator *I is a function $I : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}$ that assigns each pair of approximation sets a real value $I(A, B)$.*

2.4.2 Absolute Quality Indicators

The motivation of using quality indicators is to give more precise, quantitative statements about differences between approximation sets by applying common metrics (in the mathematical sense) to the scalar values resulting from the indicator. Ideally, a larger quality indicator value would mean that the corresponding approximation set is better, and vice versa:

$$A \succ B \Leftrightarrow I(A) > I(B)$$

Following the same argumentation as in the previous section regarding fitness functions, a numerical representation of $(\mathcal{A}, \succeq)$, i.e. an order isomorphism into the reals, cannot exist because $\succeq$ is not connected. Given this general limitation, the following questions:

- what can quality indicators achieve,
- what implications do they allow, and
- how useful are they?

The inferential power of quality indicators can be analyzed by investigating the type of order homomorphism they represent. An order homomorphism of a relation $R \subseteq \mathcal{A} \times \mathcal{A}$ into $S \subseteq \mathbb{R} \times \mathbb{R}$ allows to infer

$$ARB \Rightarrow I(A)SI(B).$$

By negating this logical implication, we can then arrive at conclusions that can be drawn from the indicator values:

$$\neg(ARB) \Leftarrow \neg(I(A)SI(B))$$

and, if S is asymmetric,

$$\neg(ARB) \Leftarrow I(B)SI(A)$$

Thus, we can only infer from the indicator values that A and B are not in relation R. This already shows that the inferential power regarding our preference relations on approximation sets is limited, since $\neg(ARB)$ does not imply BRA unless R is complete. This deficiency is illustrated in the following example.

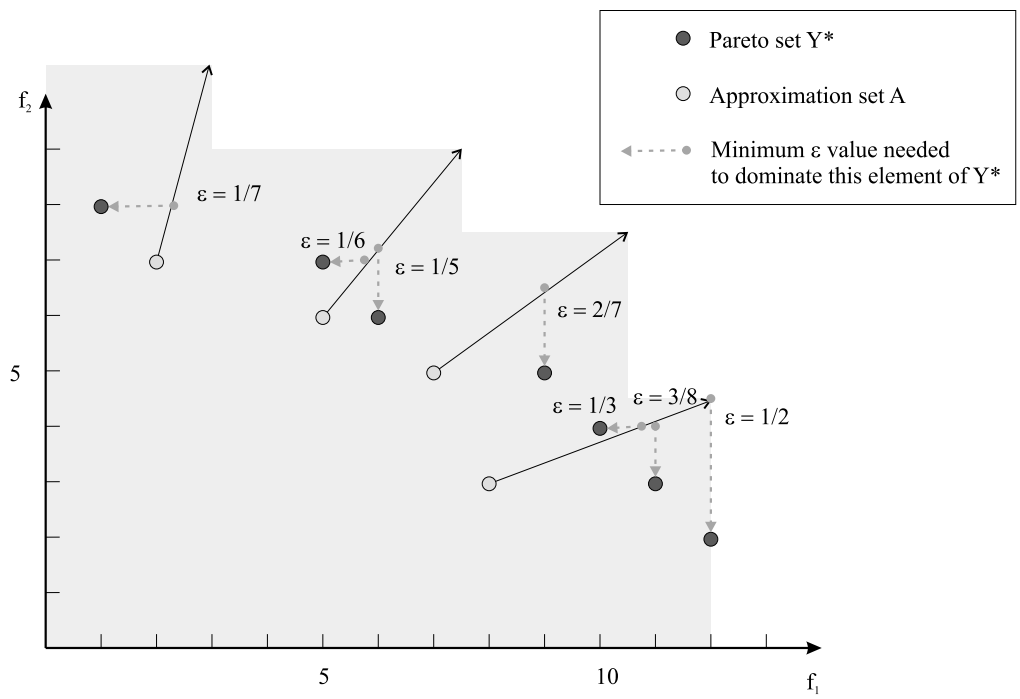


Fig. 7: Construction principle of the ε-indicator. For each Pareto-optimal objective vector $y \in Y^*$ (dark shaded points), the corresponding element from $a \in A$ (light shaded points) has to be identified with the minimum ε such that $(1+\varepsilon)a \succeq y$. The maximum over all these ε-values determines the indicator value. In this case, $I_\varepsilon(A) = 0.5$. Similarly, the indicator values for the other two sets from Figure 6 can be calculated as $I_\varepsilon(B) = 0.4$ and $I_\varepsilon(C) = 1$. The grey shaded area corresponds to all objective vectors that are within a factor of $(1+\varepsilon)$, $\varepsilon = 0.5$, of set A. It can be seen that this area is just large enough to touch the "critical" Pareto-optimal objective vector $(12, 2)$.

Ex. 2: **(ε-indicator[3])** *Let $f(X) \subseteq \mathbb{R}^{+m}$. The ε-indicator I_ε is defined as*

$$I_\varepsilon(A) := \inf_{\varepsilon \in \mathbb{R}} \{\forall y \in Y^* \; \exists a \in A : (1+\varepsilon) \cdot a \succeq y\}.$$

The ε-indicator assumes w.l.o.g. a positive objective space and requires the Pareto front to be known. It can then be calculated in $O(m \cdot |A| \cdot |Y^*|)$ arithmetic operations as

$$I_\varepsilon(A) = \max_{y \in Y^*} \min_{a \in A} \max_{1 \le i \le m} \frac{a_i}{y_i} - 1,$$

because for each Pareto-optimal objective vector, we want to find the solution with the best – the minimum – approximation factor (see also Figure 7. The overall indicator value is then the maximum over those factors for all Pareto-optimal objective vectors Y^*.

On the basis of the conventional order relations applied to the ε-indicator values and the example sets from Figure 6, it can be stated that

$$\begin{aligned} A \succeq C \quad &\Rightarrow \quad I_\varepsilon(A) \le I_\varepsilon(C) \\ A \not\succ B \Leftarrow A \not\succeq B \quad &\Leftarrow \quad I_\varepsilon(A) > I_\varepsilon(B) \end{aligned}$$

and, since only the pointwise greater relation is a sufficient condition for a strict inequality of the indicator values,

$$\begin{aligned} A \dot{\succ} C \quad &\Rightarrow \quad I_\varepsilon(A) < I_\varepsilon(C) \\ C \not\dot{\succ} B \quad &\Leftarrow \quad I_\varepsilon(C) \ge I_\varepsilon(B) \end{aligned}$$

Furthermore, $I_\varepsilon(Z) = 0$ for a hypothetical set Z would imply that $Z = Y^*$ so that the more complicated relation $D := \{(0, r) \in \mathbb{R}^2 : r > 0\}$ can be used to imply

$$Z \succ A \Leftarrow I_\varepsilon(Z) D I_\varepsilon(A) \Leftrightarrow (I_\varepsilon(Z) = 0 \wedge I_\varepsilon(A) > 0).$$

Here, the negation

$$Z \not\succ A \Rightarrow \neg(I_\varepsilon(Z) D I_\varepsilon(A)) \not\Rightarrow I_\varepsilon(A) D I_\varepsilon(Z)$$

does not lead to any useful statements. The reason is that the implication fails because D is not connected.

The last aspect of the example can even be generalized: Although relations serving as "detectors" of the dominance relation on approximation sets can be constructed, they have no practical use because they are not connected.

Lem. 1: *Let $I : \mathcal{A} \mapsto \mathbb{R}$ be a quality indicator, $S \subset \mathbb{R}^2$ and $R \subset \mathcal{A} \times \mathcal{A}$ binary relations, where R is not connected. If for all $A, B \in \mathcal{A}$*

$$I(A) S I(B) \Rightarrow A R B$$

then S is also not connected.

[3] A comprehensive treatment of ε-approximations of Pareto-optimal sets will be given in Chapter 3 in connection with investigation of the convergence behavior of multiobjective evolutionary algorithms

Proof: Suppose S is connected. Then

$$ARB \Rightarrow \neg(BRA) \Rightarrow \neg(I(B)SI(A)) \Rightarrow I(A)SI(B)$$

and therefore

$$ARB \Leftrightarrow I(A)SI(B)$$

which contradicts Theorem 1. □

The above consideration shows that connected relations of the indicator values, e.g., the conventional order relations, allow only limited conclusions regarding the approximation sets. Strong statements are only possible for smaller sets of indicator value pairs. Lemma 1 states, that even though sufficient conditions to detect whether an approximation set is better than another may exist, they cannot be expressed by a connected relation on $I(\mathcal{A})$. Therefore, such a relation holds for few pairs only and cannot be an order homomorphism, simply because it would be an order isomorphism, which is precluded by Theorem 1. As a result, absolute quality indicators can be classified according to their inferential direction:

1. Quality indicators that are order homomorphism from the set of approximation sets into the set of indicator values,

2. Quality indicators whose inverse is an order homomorphism from the set of indicator values to the set of approximation sets,

3. Quality indicators that have no inferential power at all.

Quality indicators of the first type are therefore most frequently used as they do not contradict the preference relation on approximation sets. Indicators of the second type are not useful, because they necessarily evaluate some approximation sets better than others though the given preference relation expresses exactly the opposite.

Table 2 gives an overview of several absolute quality indicators together with a characterization of their inferential power. In this context, the example of the hypervolume indicator proposed by Zitzler and Thiele (1998) allows the strongest statements. This quality indicator returns the hypervolume of the fraction of the objective space that is weakly dominated by an approximation set A.[4] From $A \succ B$ follows $I_H(A) > I_H(B)$. The reason is that A must contain at least one objective vector which is not weakly dominated by B, therefore a certain portion of the objective space is dominated by A but not by B.

Another benefit of being an order homomorphism is that the Pareto front corresponds to an extremum in indicator value space. If this extremum is unique, the quality indicator can be used to detect Pareto-optimality, as suggested by

[4]Note that $f(X)$ has to be bounded, i.e., there must exist a hypercube in $\mathbb{R}^n$ that encloses $f(X)$. If this requirement is not fulfilled, it can be easily achieved by an appropriate transformation.

Indicator	Name (Reference)	$\dot{>}$	$\dot{\succ}$	$\succ$	$>$	$\geq$
I_{H}	Hypervolume indicator (Zitzler and Thiele 1998)	$>$	$>$	$>$	$\not<$	$\not<$
I_{ε}	ε-indicator (Example 2)	$>$	$\geq$	$\geq$	$\not<$	$\dot{\not<}$
I_{D}	Distance from reference set (Czyzak and Jaskiewicz 1998)	$>$	$\geq$	$\geq$	$\not<$	$\dot{\not<}$
I_{PF}	Fraction of Pareto front covered (Ulungu et al. 1999)	$\geq$	$\geq$	$\geq$	$\not\leq$	$-$
I_{P}	Number of Pareto points contained (Zitzler et al. 2003)	$\geq$	$\geq$	$\geq$	$\not\leq$	$-$
I_{ER}	Error ratio (Van Veldhuizen 1999)	$\geq$	$\geq$	$-$	$\dot{\not<}$	$-$

Tab. 2: Overview of some absolute quality indicators and their properties. The third column lists the relation of indicator values that follow from the $\dot{>}$, $\dot{\succ}$ and $\succ$ relations of the approximation sets in the top line, while the forth column lists the relations of the approximation sets that follow from the $>$ and $\geq$ relations of the indicator values in the top line.

Fleischer (2003). Fleischer (2002) proved that the hypervolume indicator attains its maximum if and only if its argument is the Pareto front. With the framework developed in this chapter, it is possible to prove the following, more general theorem.

Th. 2: *Let A be an approximation set and $I : \mathcal{A} \mapsto \mathbb{R}$ a quality indicator. The condition*

$$I(A) = \max\{I(B) : B \in \mathcal{A}\}$$

is

- *necessary for A being the Pareto front Y^*, if I is an order homomorphism of $(\mathcal{A}, \succeq)$ into $(\mathbb{R}, \geq)$, and*
- *necessary and sufficient for A being the Pareto front Y^*, if I is an order homomorphism of $(\mathcal{A}, \succ)$ into $(\mathbb{R}, >)$.*[5]

Proof: For the necessary conditions assume that $A = Y^*$. Therefore, for all $B \in \mathcal{A}$, $B \neq A$: $A \succ B$. If I is an order homomorphism of $(\mathcal{A}, \succeq)$ into $(\mathbb{R}, \geq)$, then $I(A) \geq I(B)$ for all $B \in \mathcal{A}$. If I is an order homomorphism of $(\mathcal{A}, \succ)$ into $(\mathbb{R}, >)$, then $I(A) > I(B)$ for all $B \in \mathcal{A}$, $B \neq A$. In both cases, $I(A)$ is maximum. For the sufficient condition assume $A \neq P$. In this case $P \succ A$ and,

[5] Of course, the theorem also holds for quality indicators, where smaller values are considered better, i.e., for order homomorphisms of $(\mathcal{A}, \succeq)$ into $(\mathbb{R}, \leq)$ and $(\mathcal{A}, \succ)$ into $(\mathbb{R}, <)$.

since I is an order homomorphism of $(\mathcal{A}, \succ)$ into $(\mathbb{R}, >)$, $I(P) > I(A)$, which contradicts the precondition that $I(A)$ is maximum. □

Most of the quality indicators listed in Table 2 are order homomorphisms into $(\mathbb{R}, \leq)$, for which the sufficient condition could not be proven in this general manner. This does not preclude that the condition is sufficient for all these indicators. For the ε-indicator, e.g., only the Pareto front achieves the minimum value of $I_\varepsilon = 0$. This, however, can only be found through a case-by-case inspection of the quality indicators.

One class of indicators that belong to third type of the above enumeration and do not allow any conclusions to be drawn regarding the dominance relationship between approximation sets is represented by the various diversity indicators (Srinivas and Deb 1994; Schott 1995; Zitzler 1999; Sayin 2000; Deb 2001; Wu and Azarm 2001). If we consider a pair $A, B \in \mathcal{A}$ with $A \succ B$, in general the indicator value of A can be less, or greater than, or even equal to, the value assigned to B (for the diversity indicators mentioned above). Therefore, the orders induced by these indicators are neither sufficient nor necessary conditions for any of the preference relations defined on approximation sets, and no statements are possible.

Given these limitations, what further information can be gained by considering multiple quality indicators at the same time, i.e., quality vectors? Indeed, many authors have claimed that "the performance of multiobjective optimizers is a multiobjective problem itself". We already know that $(\mathcal{A}, \succeq)$ cannot isomorphically be mapped to $(\mathbb{R}, \geq)$ or to any other connected relation on the reals, but can it be mapped to $(\mathbb{R}^k, \succeq)$? An equivalent question is, what is the dimension of the partially ordered set $(\mathcal{A}, s)$? If $\dim(\mathcal{A}, \succeq) = k < \infty$, then there would exist a set of k linear orders whose intersection is equal to $\succeq$. The purpose of such a collection of k indicators would be to tell:

- if one set is better than the other, how much better it is, or
- if two sets A and B are incomparable to each other, are there certain aspects, expressed by the scalar indicators, in which A is better than B, and others, where B is superior to A.

The variety among the indicators proposed suggests that this goal is, at least, difficult to achieve. The following theorem shows it cannot be achieved in general.

Th. 3: **(Zitzler et al. 2003)** *Suppose an optimization problem with $m \geq 2$ objectives where the objective space is $Y = \mathbb{R}^m$. Then, there does not exist a finite combination of k quality indicators $I : \mathcal{A} \mapsto \mathbb{R}^k$, $I = (I_1, \ldots, I_k)$ such that I is an order isomorphism of $(\mathcal{A}, \succ)$ into $(\mathbb{R}^k, R)$ for any relation R, i.e.*

$$A \succ B \Leftrightarrow I(A) R I(B)$$

for any approximation sets $A, B \in \mathcal{A}$.

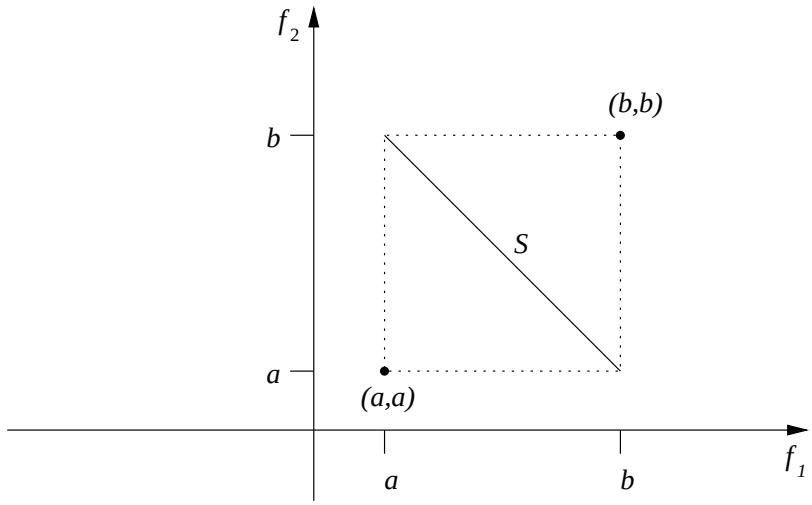


Fig. 8: Illustration of the construction used in Theorem 3 for a two-dimensional minimization problem. We consider an open rectangle $(a, b)^2$ and define an open line S within. For S holds that any two objective vectors contained are incomparable to each other, and therefore any subset $A \subseteq S$ is an approximation set.

Sketch of proof: The proof is based on the following fundamental results from set theory (Hrbacek and Jech 1999):

- $\mathbb{R}$, $\mathbb{R}^k$, and any open interval (a, b) in $\mathbb{R}$ resp. hypercube $(a, b)^k$ in $\mathbb{R}^k$ have the same cardinality, denoted as $2^{\aleph_0}$, i.e., there is a bijection from any of these sets to any other;
- If a set S has cardinality $2^{\aleph_0}$, then the cardinality of the power set $\mathcal{P}(S)$ of S is $2^{2^{\aleph_0}}$, i.e., there is no injection from $\mathcal{P}(S)$ to any set of cardinality $2^{\aleph_0}$.

As we consider the most general case where $Y = \mathbb{R}^m$, it is possible to construct a set S (see Figure 8) such that any two points contained are incomparable to each other. Accordingly, any subset A of S is an approximation set and the power set of S, the cardinality of which is $2^{2^{\aleph_0}}$, is exactly the set of all approximation sets $A \subseteq S$. We will then show that any two approximation sets $A, B \subseteq S$ with $A \neq B$ must differ in at least one of the k indicator values. Therefore, an injection from a set of cardinality $2^{2^{\aleph_0}}$ to $\mathbb{R}^k$ is required, which finally leads to a contradiction.

Note that Theorem 3 also holds (i) if we only assume that Y contains an open hypercube in $\mathbb{R}^m$ and (ii) if we consider any other preference relations on $\mathcal{A}$ (for $\parallel$ and $\succeq$ it follows directly from Theorem 3, for $\dot{\succ}$ and $\dot{>}$ the proof has to be slightly modified).

The construction of an order isomorphism is possible, however, if we restrict the cardinality of approximation sets and consider only $\mathcal{A}' := \{A \in \mathcal{A} : |A| \leq l\}$ for a fixed $l \in \mathbb{N}$. In this case one could define a bijective mapping $I : \mathcal{A}' \mapsto \mathbb{R}$ such that I is an order isomorphism of $(\mathcal{A}', \succeq)$ into $(\mathbb{R}, R)$, where R

is defined as $aRb :\Leftrightarrow I^{-1}(a) \succeq I^{-1}(b)$. The existence of such an indicator is only of theoretical use since we cannot apply any meaningful metrics to it. □

As a result we conclude that $\dim(\mathcal{A}, \succeq) = \infty$, i.e., we cannot describe the preference relation on approximations sets with a finite number of scalar absolute indicators.

2.4.3 Relative Quality Indicators

Relative quality indicators can be used to overcome the difficulties with absolute indicators. However, they also have a drawback: comparing t algorithms using a single relative indicator results in $t(t-1)$ distinct indicator values – in contrast to the t values in the case of an absolute indicator. This renders the analysis and the presentation of the results more difficult. Nevertheless, Theorem 3 suggests that this is in the nature of multiobjective optimization problems.

In principle, there are no such theoretical limitations of relative indicators as for absolute indicators. The reason is that the class of relative indicators contains the preference relations defined on $\mathcal{A}$, i.e., the preference relation is in fact a special relative indicator. While preference relations represent *binary* relations, the relative indicators are general *valued* relations. Consider for instance the indicator

$$I(A, B) := \begin{cases} 5 & \text{if } A \dot{\succ} B \\ 4 & \text{if } A \dot{\succ} B \wedge A \not\dot{\succ} B \\ 3 & \text{if } A \succ B \wedge A \not\dot{\succ} B \\ 2 & \text{if } A \succeq B \wedge A \not\succ B \\ 1 & \text{if } A = B \\ 0 & \text{else} \end{cases}$$

which is a compact representation of all preference relations we used so far. By using a finer scale of the values, quality differences can be expressed in more detail. A function $s : \mathbb{R} \mapsto \{0, 1\}$ can then be defined to map $I(A, B)$ to a preference relation R on $\mathcal{A}$, i.e. $s(I(A, B)) \Leftrightarrow ARB$. Consider the following example, which is a generalization of the ε-indicator from Example 2.

Ex. 3: **(Relative ε-indicator)** *Let $f(X) \subseteq \mathbb{R}^{+m}$. The ε-indicator I_ε is defined as*

$$I_\varepsilon(A, B) := \inf_{\varepsilon \in \mathbb{R}} \{\forall b \in B \; \exists a \in A : (1+\varepsilon) \cdot a \succeq b\}.$$

With this definition the following equivalences hold:

$$\begin{aligned} & A \succeq B \quad \Leftrightarrow \quad I_\varepsilon(A, B) \leq 0 \\ \textit{and} \quad & A \dot{\succ} B \quad \Leftrightarrow \quad I_\varepsilon(A, B) < 0. \end{aligned}$$

Furthermore,

$$\begin{aligned} A \succ B \quad &\Leftrightarrow \quad A \succeq B \wedge B \not\succeq A \quad \Leftrightarrow \quad I_\varepsilon(A, B) \leq 1 \wedge I_\varepsilon(B, A) > 1 \\ A = B \quad &\Leftrightarrow \quad A \succeq B \wedge B \succeq A \quad \Leftrightarrow \quad I_\varepsilon(A, B) = 1 \wedge I_\varepsilon(B, A) = 1 \\ A \parallel B \quad &\Leftrightarrow \quad A \not\succeq B \wedge B \not\succeq A \quad \Leftrightarrow \quad I_\varepsilon(A, B) > 1 \wedge I_\varepsilon(B, A) > 1. \end{aligned}$$

The example shows, how the valued relations given by the quality indicator correspond directly to the binary preference relations, while the magnitude of the quality indicator values provides additional information about how much one approximation set is better than another. It also demonstrates how logical combinations of the indicator values can be used to derive equivalences with further preference relations.

However, care has to be taken when defining a relative indicator, as not all existing relative indicators guarantee the existence of any logical combination to create equivalences with the binary order relations. An example are those indicators that are, as Knowles and Corne (2002) denote it, *symmetric*, i.e., $I(A, B) = c - I(B, A)$ for a constant $c \in \mathbb{R}$. Even though symmetric indicators are attractive as only half the number of indicator values have to be considered in comparison to a general relative indicator, their inferential power is restricted as shown by the following theorem.

Th. 4: **(Zitzler et al. 2003)** *Let I be a relative quality indicator with $I(A, B) = -I(B, A)$ for all $A, B \in \mathcal{A}$. If*

$$I(A, B) > I(B, A) \Leftrightarrow A \succ B$$

holds for all $A, B \in \mathcal{A}$ then also

$$I(A, B) = I(B, A) = 0 \Leftrightarrow (A = B \vee A || B)$$

for all $A, B \in \mathcal{A}$.

Table 3 gives an overview of several relative quality indicators from the literature. For each preference relation on approximation sets it depicts the equivalent conditions on the indicator values, if such conditions are possible at all.

2.5 Summary

In this chapter, the fundamentals of evolutionary multiobjective optimization was discussed from a decision-theoretic point of view. Decision theory, especially the theory of ordered sets, represents a natural way to treat the general multiobjective optimization scenario as well as multiobjective optimization algorithms in the same mathematical framework. It allows to precisely specify the limits and difficulties imposed on any algorithm under the presence of multiple objectives in contrast to the single-objective optimization scenario.

For evolutionary multiobjective optimization algorithms, the differences to their single objective counterparts are imminent in the *selection* operator as well as in the *performance assessment*. Both issues are necessarily connected to order relations: in the former for comparing single decision alternatives and in the latter for comparing sets of decision alternatives. Common approaches to

ind.	I_ε	I_{C}	I_{H2}	I_{R1}	I_{R2}	I_{R3}	I_{LI}
$\dot{\succ}$	$I_\varepsilon(A, B) < 0$	-	-	-	-	-	-
$\dot{\prec}$	-	$I_{\mathrm{C}}(A, B) = 1$ $I_{\mathrm{C}}(B, A) = 0$	-	-	-	-	-
$\succ$	$I_\varepsilon(A, B) \leq 0$ $I_\varepsilon(B, A) > 0$	$I_{\mathrm{C}}(A, B) = 1$ $I_{\mathrm{C}}(B, A) < 1$	$I_{\mathrm{H2}}(A, B) > 0$ $I_{\mathrm{H2}}(B, A) = 0$	-	-	-	-
$\succeq$	$I_\varepsilon(A, B) \leq 0$	$I_{\mathrm{C}}(A, B) = 1$	$I_{\mathrm{H2}}(A, B) \geq 0$ $I_{\mathrm{H2}}(B, A) = 0$	-	-	-	-
$=$	$I_\varepsilon(A, B) = 0$ $I_\varepsilon(B, A) = 0$	$I_{\mathrm{C}}(A, B) = 1$ $I_{\mathrm{C}}(B, A) = 1$	$I_{\mathrm{H2}}(A, B) = 0$ $I_{\mathrm{H2}}(B, A) = 0$	-	-	-	-
$\parallel$	$I_\varepsilon(A, B) > 0$ $I_\varepsilon(B, A) > 0$	$I_{\mathrm{C}}(A, B) < 1$ $I_{\mathrm{C}}(B, A) < 1$	$I_{\mathrm{H2}}(A, B) > 0$ $I_{\mathrm{H2}}(B, A) < 1$	-	-	-	-

I_{C}: Coverage indicator (Zitzler 1999)
I_{H2}: Binary hypervolume indicator (Zitzler 1999)
I_{R1}, I_{R2}, I_{R3}: Utility function indicators R1 to R3 (Hansen and Jaszkiewicz 1998)
I_{LI}: Lines of intersection (Knowles and Corne 2000)

Tab. 3: Overview of relative quality indicators. The entries in the table correspond to relations of the indicator values that are equivalent with the respective preference relation on the approximation sets.

both issues are to look for real-valued *representations* of the underlying order relations. For selection operators this is formalized by the notion of *fitness functions*, for performance assessment by the notion of *quality indicators*. As the main result in this chapter, it was proven that an order representation in the sense of an isomorphism cannot exist in both cases, and the consequences of this fact were discussed. We proposed further, weaker properties and classified existing fitness function and quality indicators accordingly.

3

Limit Behavior and Global Convergence

This chapter is devoted to the analysis of the limit behavior of multiobjective evolutionary algorithms, which is achieved by studying the random sequence of its solution set $(A^{(t)})$ and its maximal elements $(A^{(t)*})$ for $t \to \infty$. For a multiobjective optimization algorithm it is desirable that it converges to the Pareto set at least in the limit, i.e., when unlimited time resources are available.

For a randomized search algorithm, convergence needs to be defined in a probabilistic sense. A multiobjective optimization algorithm is said to converge to the Pareto set X^* with probability one[1], if

$$\mathsf{P}\left\{\lim_{t\to\infty} A^{(t)*} \subseteq X^*\right\} = 1.$$

Sequences of approximation sets that do not converge to a subset of the Pareto set occur with zero probability. If the equality $\lim_{t\to\infty} A^{(t)*} = X^*$ holds in the above equation, the algorithm is said to converge to the *whole* Pareto set.

Convergence to the whole Pareto set can easily be proven if we assume that (i) every decision alternative is generated with probability one during the run and (ii) non-dominated alternatives are never discarded from the internal memory A. As the Pareto set can contain a huge or even infinite number of solutions, this result is of little practical relevance.

One is actually interested in a *bounded approximation* of the Pareto set, i.e., a solution set of bounded size, which represents the Pareto set well. To fulfill the first requirement, an algorithm has to work with a solution storage of bounded size and can therefore in most cases only converge to a subset of the Pareto set. Nevertheless, this subset should be as diverse as possible in order to be a

[1]Convergence with probability one is also called almost sure convergence; for an overview and discussion of different modes of stochastic convergence in the area of evolutionary computation see Rudolph (1997b).

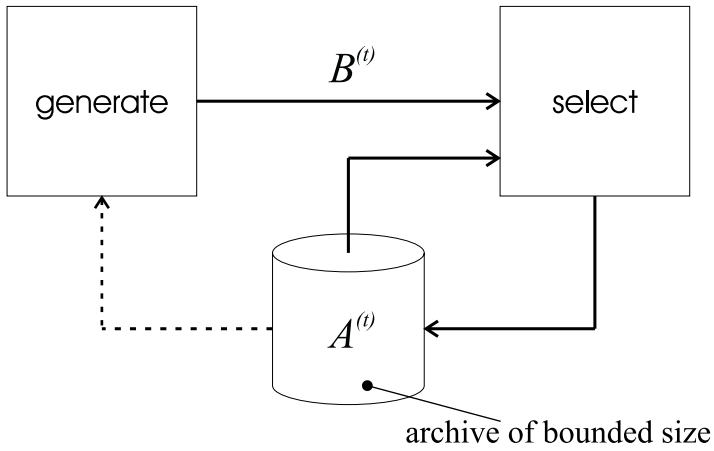


Fig. 9: Block diagram of Algorithm 1. The select operator has to work with an solution storage of bounded capacity.

good representation of the Pareto set. In summary, we are therefore looking for algorithms that

1. converge to the Pareto set,
2. maintain a diverse set of solutions, and
3. work with bounded memory.

The discussion of related work in the following section shows that such algorithms do not exist so far. The intention of this chapter is to fill this gap and answer the question, how do we guarantee that an algorithm converges to a subset of solutions that represents the Pareto set well under the assumption of unlimited time resources, but limited memory resources.

To achieve this goal, we start by specifying what we understand by a bounded approximation of the Pareto set. Based on the notion of ε-dominance, this definition combines the aspects of approximation quality and diversity. Building on this concept, we propose two new selection operators and prove that they fulfill all three properties listed above. Section 3.3 then presents some simulation results to demonstrate the behavior of the new algorithms and to visualize the important differences to the existing approaches. In Section 3.4 various practically relevant extensions to the new approach are outlined and discussed.

3.1 Related Work

The task of bounding the size of the stored solution set is performed by the selection operator. This is illustrated in Figure 9, which displays a block diagram of the conceptual evolutionary algorithm, Algorithm 1. The solution storage

will be referred to in the following as the *archive* of the algorithm; and the focus of the subsequent investigations is therefore on the *archiving strategy* of the selection operator. The archiving strategies proposed in the context of multiobjective evolutionary algorithms can be divided into two categories depending on whether their focus lies on convergence or distribution quality.

3.1.1 Algorithms for Guaranteed Convergence

Theoretic work on convergence in evolutionary multiobjective optimization is mainly due to Rudolph (1998a, 1998c, 2001), Rudolph and Agapie (2000), and Hanne (1999, 2001). The aim of the authors was to construct algorithms for which they can prove convergence to the Pareto set. Their concepts and corresponding algorithms are described in the following.

Efficiency Preservation and the Problem of Deterioration Hanne suggested (1999) and implemented (2001) a selection strategy for MOEAs based on the concept of "(negative) efficiency preservation" as a multiobjective generalization of the "plus" (elitist) selection, also denoted as a $(\mu + \lambda)$-strategy[2]. He defines efficiency preservation as the property of only accepting new solutions if they dominate at least one of the current solutions. Negative efficiency preservation is given when a solution is discarded only if a dominating solution is accepted in return. Both properties are mutually independent, and sufficient to preclude the problem of *deterioration*. Deterioration occurs, when elements of a solution set at a given time are dominated by a solution set the algorithm maintained some time before. This can happen using the standard Pareto-based selection schemes even under elitism, as well as with virtually all selection schemes used in the advanced state-of-the-art MOEAs, as will be described shortly.

In Hanne (1999) a convergence proof for a $(\mu + \lambda)$-MOEA with Gaussian mutation distributions over a compact real search space has been enabled by the application of a (negative) efficiency preservation selection scheme. A disadvantage of this approach is that no assumptions can be made as to the distribution of solutions, since with both efficiency and negative efficiency preservation arbitrary regions of the objective space – and hence of the Pareto front – can become unreachable.

Rudolph's and Agapie's Elitist MOEAs Similar to Hanne's idea is Rudolph's (1998a) concept of *elite preservation*. Based on this concept, Rudolph and Agapie (2000) suggested MOEAs with a fixed-size archive, where a sophisticated selection process precludes the problem of deterioration. The

[2]This notation refers to how the selection operator chooses individuals for the next generation. The old parent population of size μ is merged with the offspring population of size λ. Then, the best μ solutions of this union are deterministically chosen to form the parent population of the next generation. This selection strategy is called elitist, as the best individuals always survive. This view, though, holds only if a total order exists on the set of individuals, e.g., for single-objective optimization.

authors have shown that these algorithms converge to the Pareto set, provided that their variation operators guarantee to generate each point of the decision space with a probability not less than an arbitrarily small positive constant. However, when all archive members are Pareto-optimal, the algorithm does not allow any new Pareto-optimal solution to enter the archive. Thus, although the algorithms guarantee convergence to the Pareto set, they do not guarantee a good distribution of Pareto-optimal solutions.

3.1.2 Elitist Selection with Focus on Distribution Quality

Efforts to maintain well-distributed solution sets have resulted in a number of elitist MOEAs which especially address the diversity of the archived solutions by different mechanisms. We discuss some selected representatives in order to highlight the typical working principle of the respective selection operators and the potential deficiencies regarding their convergence properties.

Pareto-Archived Evolution Strategy (PAES) Knowles and Corne (2000) suggested a simple elitist MOEA using a single parent, single child $(1+1)$-EA called Pareto-Archived Evolution Strategy (PAES). If a new solution is not dominated by any archive member it is included in the archive, deleting in turn all members that it dominates. If the archive exceeds its maximum size, the acceptance of new solutions is decided by a histogram-like density measure over a hyper-grid division of the objective space. This archiving strategy is similar to the one proposed by Kursawe (1990, 1991), who already used an adaptive distance measure to maintain a good spread of non-dominated solutions in a fixed-size archive.

Strength Pareto Evolutionary Algorithm (SPEA) Zitzler and Thiele (1999) have suggested an elitist MOEA using the concept of non-domination and a secondary population of non-dominated points. After every generation, the secondary population is updated with the non-dominated offspring, while all dominated elements are discarded. If this archive exceeds its maximum size, a clustering mechanism groups all currently non-dominated solutions into a pre-defined number of clusters and picks a representative solution from each cluster, thereby ensuring diversity among the external population members.

Elitist Non-Dominated Sorting GA (NSGA-II) In NSGA-II (Deb et al. 2000), the parent and offspring population (each of size N) are combined and evaluated using (i) a fast non-dominated sorting approach, (ii) an elitist approach, and (iii) an efficient crowding approach. When more than N population members of the combined population belong to the non-dominated set, only those that are maximally apart from their neighbors according to the crowding measure are chosen. This way, like when using PAES and SPEA, an existing non-dominated solution may get replaced by

another, since selection is then based only on the specific diversity measure or on the clustering procedure. In a succession of these steps, deterioration may occur such that convergence can no longer be guaranteed for any of these algorithms.

3.1.3 Limitations of Existing Selection Strategies

It is clear from the above discussion that the existing elitist MOEAs cannot achieve both tasks simultaneously, either they enable convergence or they focus on a good distribution of solutions. The convergence criterion can easily be fulfilled by dominance preservation; however, a pure implementation of this approach leaves the distribution aspect unsolved. All algorithms focusing on a good distribution are on the other hand in danger of deterioration. The diversity-preservation operator used in each of the above algorithms are primarily geared to maintain spread among solutions. While doing so, the algorithms have no way of *knowing* which solutions are already Pareto-optimal and which are not. The diversity-preservation operators alway emphasize the less crowded regions of the non-dominated solutions.

3.2 Algorithms for Convergence and Diversity

The above discussion of existing selection operators reveals that none of them is able to combine convergence and diversity, as it is desirable for a multiobjective optimization algorithm. We want to overcome this fundamental problem and proceed as follows. First, we discuss why it is important to bound the size of the archived solution set. These considerations motivate the idea of discrete Pareto set approximations as an appropriate solution concept for multiobjective optimization problems. Using the notion of ε-dominance, we show that the corresponding ε-approximate Pareto sets also comply with the diversity aspect. What remains to be addressed is the convergence aspect. To this end, two selection algorithms are constructed that implement the proposed solution concept. The idea of the algorithms is to always maintain a ε-approximation of all solutions produced so far. From this invariance property, the convergence of the algorithm to a discrete, representative and well-distributed solution set is an immediate consequence.

3.2.1 Concept of Pareto Set Approximation

In many multiobjective optimization problems, the Pareto set X^* is of substantial size. Thus, the numerical determination of X^* is prohibitive, and X^* as a result of an optimization is questionable. Moreover, it is not clear at all what a decision maker can do with such a large result of an optimization run. What would be more desirable is an approximation of X^* which *approximately* dominates all elements of X and is of reasonable size. This set can then be used

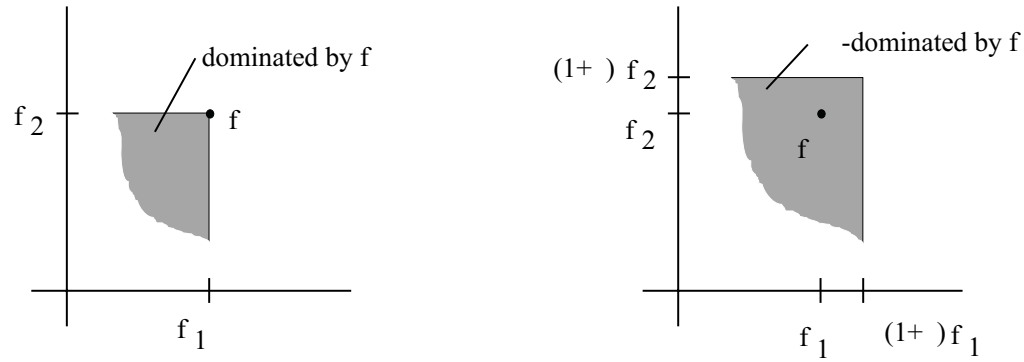


Fig. 10: Graphs visualizing the concepts of dominance (left) and ε-dominance (right).

by a decision maker to determine interesting regions of the decision and objective space, which can be explored in further optimization runs. Next, we define a generalization of the dominance relation as visualized in Figure 10 (right). Without loss of generality, a normalized and positive objective space as well as a maximization problem is assumed for notational convenience.

Def. 15: **(ε-dominance)** *Let $f : X \mapsto \mathbb{R}^{+m}$ and $a, b \in X$. Then a is said to* ε-dominate *b for some $\varepsilon > 0$, denoted as $a \succ_\varepsilon b$, if and only if for all $i \in \{1, \ldots, m\}$*

$$(1 + \varepsilon) \cdot f_i(a) \geq f_i(b). \tag{3.1}$$

Def. 16: **(ε-approximate Pareto set)** *Let X be a set of decision alternatives and $\varepsilon > 0$. Then a set X_ε is called an* ε-approximate Pareto set *of X, if any vector $a \in X$ is ε-dominated by at least one vector $b \in X_\varepsilon$, i.e.,*

$$\forall a \in X : \exists b \in X_\varepsilon \quad \textit{such that} \quad b \succ_\varepsilon a. \tag{3.2}$$

The set of all ε-approximate Pareto sets of X is denoted as $P_\varepsilon(X)$. The image of an ε-approximate Pareto set $F_\varepsilon = f(X_\varepsilon)$ is called an ε-approximate Pareto front.

Of course, the set X_ε is not unique. Many different concepts for ε-efficiency[3] and the corresponding Pareto set approximations exist in the operations research literature, a survey is given by Helbig and Pateva (1994). As most of the concepts deal with infinite sets, they are not practical for our purpose of producing and maintaining a representative subset. Nevertheless, they are of theoretical interest and have properties which can be used for instance in convergence proofs, see Hanne (1999) for an application in MOEAs.

Using discrete ε-approximations of the Pareto set was suggested simultaneously by Evtushenko and Potapov (1987), Reuter (1990), and Ruhe and Fruhwirt (1990). As in our approach, each Pareto-optimal point is approximately

[3]The terms "efficient" and "Pareto-optimal" can be used synonymously. While the former appears to be more frequent in operations research literature, we generally use the latter as it is more common in the field of evolutionary computation.

dominated by some point of the representative set. The first two studies use absolute deviation (additive ε, see below) and the third uses relative deviation (multiplicative ε as above), but they are not concerned with the size of the representative set in the general case.

Regarding the size of such Pareto set approximations, Papadimitriou and Yannakakis (2000) and Erlebach et al. (2001) have pointed out that under very mild assumptions, which also hold in our case, there is always an approximate Pareto set whose size is polynomial in the length of the encoded input, i.e., in the logarithm of the largest objective value. This can be achieved by placing a hyper-grid in the objective space using the coordinates 1, $(1+\varepsilon)$, $(1+\varepsilon)^2, \ldots$ for each objective. As it suffices to have one representative solution in each grid cell and to have only non-dominated cells occupied, it can be seen that for any finite ε and any set X with bounded image in objective space, i.e., $1 \leq f_i(x) \leq K$ for all $x \in X$, $i \in \{1, \ldots, m\}$, there exists a set X_ε containing

$$|X_\varepsilon| \leq \left(\frac{\log K}{\log(1+\varepsilon)} \right)^{m-1} \tag{3.3}$$

vectors. A proof will be given in connection with Algorithm 3 in Section 3.2.3. Note that the concept of approximation can also be used if other similar definitions of ε-dominance are used, e.g., the following additive approximation

$$\varepsilon_i + f(b) \geq f(a) \quad \forall i \in \{1, \ldots, m\} \tag{3.4}$$

where ε_i are constants, separately defined for each coordinate. In this case there exist ε-approximate Pareto sets whose size can be bounded as follows:

$$|X_\varepsilon| \leq \prod_{j=1}^{m-1} \frac{K-1}{\varepsilon_i} \tag{3.5}$$

where $1 \leq f_i \leq K$, $K \geq \varepsilon_i$ for all $i \in \{1, \ldots, m\}$. A further refinement of the concept of ε-approximate Pareto sets leads to the following definition.

Def. 17: **(ε-Pareto set)** *Let X be a set of decision alternatives and $\varepsilon > 0$. Then a set $X^*_\varepsilon \subseteq F$ is called an ε-Pareto set of X if*

1. *X^*_ε is an ε-approximate Pareto set of X, i.e., $X^*_\varepsilon \in P_\varepsilon(X)$, and*
2. *X^*_ε contains maximal elements of X only, i.e., $X^*_\varepsilon \subseteq X^*$.*

The set of all ε-Pareto sets of X is denoted as $P^*_\varepsilon(X)$. The image of an ε-Pareto set $F^*_\varepsilon = f(X^*_\varepsilon)$ is called an ε-Pareto front.

The above defined concepts are visualized in Figure 11. An ε-Pareto set X^*_ε not only ε-dominates all decision alternatives in X, but also consists of Pareto-optimal decision alternatives only, therefore we have $P^*_\varepsilon(X) \subseteq P_\varepsilon(X)$.

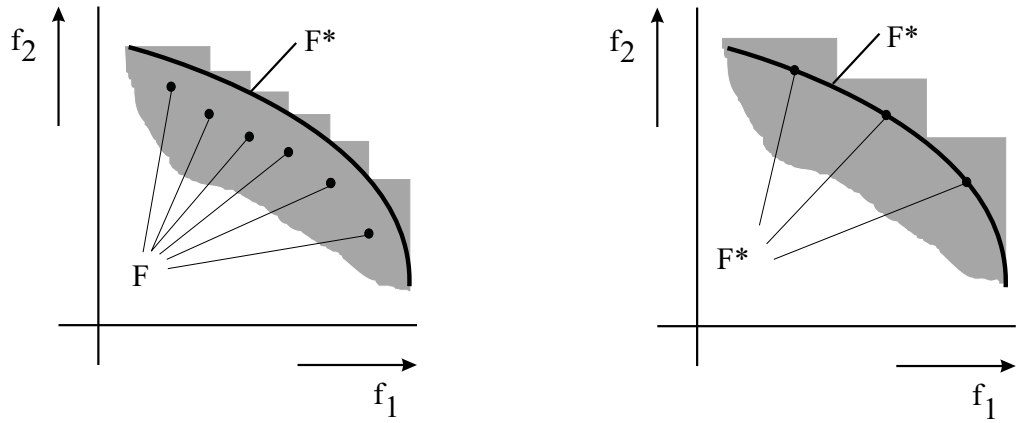


Fig. 11: Graphs visualizing the concepts of ε-approximate Pareto front (left) and ε-Pareto front (right). The Pareto front F^* is given by the black curve. The black dots represent an ε-approximate Pareto front F_ε and an ε-Pareto front F_ε^*. The shaded area contains those objective vectors that are ε-dominated by the elements of F_ε (left) and F_ε^* (right); this area must include the whole Pareto front. All elements of the ε-Pareto front F_ε^* are Pareto-optimal, which is not required for the elements of the ε-approximate Pareto front.

Since finding the Pareto set of an arbitrary decision space X is usually not practical because of its size, one needs to be less ambitious in general. Therefore, the ε-approximate Pareto set is a practical solution concept as it not only represents all alternatives X but also consists of a smaller number of elements. Of course, an ε-Pareto set is more attractive as it consists of Pareto-optimal decision alternatives only.

Diversity can be defined in various ways. Here, we associate diversity with approximation quality in the sense that we require a good approximation quality in *every* region of the Pareto set. Furthermore, the measurement of proximity is performed in the objective space only. The main reason for excluding the decision space from the considerations is that any measurement regarding diversity or proximity relies on the decision space being a metric space. This is not always the case and therefore hinders the derivation of results which are valid for all multiobjective optimization problems.

According to Definition 15, the ε value stands for a relative "tolerance" allowed for the objective values. In contrast, using equation (3.4) we would allow a constant additive (absolute) tolerance. The choice of the ε value is application specific: a decision maker should choose a type and magnitude that suits the (physical) meaning of the objective values best. The ε value further determines the maximal size of the solution set, and therefore, of the required memory according to equations (3.3) and (3.5). This is especially important in higher dimensional objective spaces, where the concept of ε-dominance can reduce the required number of solutions considerably.

3.2.2 Algorithm to Maintain an ε-approximate Pareto Set

After the definition of the type of solution set we are aiming at, the next step is to implement this solution concept algorithmically. First, a select operator is presented which leads to the maintenance of an ε-approximate Pareto set. The idea is that new solutions are only accepted if they are not ε-dominated by any other element of the current archive. If a solution is accepted, all dominated solutions are removed.

Algorithm 2 select operator for ε-approximate Pareto set

1: **Input:** A, x
2: **if** $\exists x' \in A$ such that $x' \succ_\varepsilon x$ **then**
3: $\quad A' := A$
4: **else**
5: $\quad D := \{x' \in A : x \succ x'\}$
6: $\quad A' := A \cup \{x\} \setminus D$
7: **end if**
8: **Output:** A'

Th. 5: *Let $X^{(t)} = \bigcup_{j=1}^{t} x^{(j)}$, $1 \leq f_i(x^{(j)}) \leq K$, be the set of all decision alternatives created in Algorithm 1*[4] *and given to the* select *operator as defined in Algorithm 2. Then $A^{(t)}$ is an ε-approximate Pareto set of $X^{(t)}$ with bounded size, i.e.,*

1. $A^{(t)} \in P_\varepsilon(X^{(t)})$
2. $|A^{(t)}| \leq \left(\frac{\log K}{\log(1+\varepsilon)}\right)^m$

Proof:

1. Suppose the algorithm is not correct, i.e., $A^{(t)} \notin P_\varepsilon(X^{(t)})$ for some t. According to Def. 16 this occurs only if some $x = f^{(\tau)}$, $\tau \leq t$ is not ε-dominated by any member of $A^{(t)}$ and is not in $A^{(t)}$.

   For $x = x^{(\tau)}$ not being in $A^{(t)}$, it can either have been rejected at $t = \tau$ or accepted at $t = \tau$ and removed later on. Removal, however, only takes place when some new x' enters A which dominates x (line 6). Since the dominance relation is transitive, and since it implies ε-dominance, there will always be an element in A which ε-dominates x, which contradicts the assumption. On the other hand, x will only be rejected if there is another $x' \in A^{(\tau)}$ which ε-dominates x (line 2) and – with the same argument as before – can only be replaced by accepting elements which also ε-dominate x.

[4] The decision alternatives are assumed to be produced by the generate operator in the conceptual evolutionary algorithm introduced in Section 2.2.

2. Every $x \in A^{(t)}$ defines a hyper-rectangle between x and $(1+\varepsilon)\cdot x$ where no other element of $A^{(t)}$ can exist because dominated elements are always deleted from the set. Furthermore, these areas do not overlap since this would mean that the two corresponding points ε-dominate each other, which is precluded by the acceptance criterion. The maximum number of non-overlapping hyper-rectangles in the whole objective space is given by $\left(\frac{\log K}{\log(1+\varepsilon)}\right)^m$. □

Rudolph and Agapie (2000) defined and discussed two conceptual algorithms called VV and PR whose selection strategy can be regarded as special cases of Algorithm 2 for $\varepsilon \to 0$. In the limit, the ε-dominance becomes the normal dominance relation, and the algorithm will always maintain a set of only non-dominated vectors. Of course, according to the previous theorem, the size of this set might grow to infinity during the course of the algorithm.

3.2.3 Algorithm to Maintain an ε-Pareto Set

In the next step we guarantee – in addition to a minimal distance between points – that the points in $A^{(t)}$ are maximal elements of all alternatives generated so far. The following Algorithm 3 has a two level concept. On the coarse level, the objective space is discretized by a division into boxes (see Algorithm 4), where each decision alternative uniquely belongs to one box in objective space. Using a generalized dominance relation on these boxes[5], the algorithm always maintains a set of non-dominated boxes and thereby guarantees the ε-approximation property. On the fine level, at most one element is kept per box. Within a box, each representative vector can only be replaced by a dominating one (similar to Agapie's and Rudolph's algorithm), thus guaranteeing convergence.

Algorithm 3 select operator for ε-Pareto set

Input: A, x
$D := \{x' \in A : \mathsf{box}(x) \succ \mathsf{box}(x')\}$
if $D \neq \emptyset$ **then**
  $A' := A \cup \{x\} \setminus D$
else if $\exists x' : (\mathsf{box}(x') = \mathsf{box}(x) \wedge x \succ x')$ **then**
  $A' := A \cup \{x\} \setminus \{x'\}$
else if $\nexists x' : \mathsf{box}(x') \succeq \mathsf{box}(x)$ **then**
  $A' := A \cup \{x\}$
else
  $A' := A$
end if
Output: A'

[5]The box-dominance relation is not equivalent with the ε-dominance relation, but defined as the normal dominance relation on the box index vector. Likewise, box-dominance is neither a sufficient nor a necessary condition for dominance. Both dominance and box-dominance imply ε-dominance. This property is utilized in the proof of Theorem 6.

Algorithm 4 function box

1: **Input:** x
2: **for all** $i \in \{1, \ldots, m\}$ **do**
3: $\quad b_i := \lfloor \frac{\log f_i(x)}{\log(1+\varepsilon)} \rfloor$
4: **end for**
5: $b := (b_1, \ldots, b_m)$
6: **Output:** b {box index vector}

Now, the invariance property of the above selection strategy can be proved. This guarantees the convergence of the sequence of solution sets to the Pareto set with the required diversity.

Th. 6: *Let $X^{(t)} = \bigcup_{j=1}^{t} x^{(j)}$, $1 \leq f_i(x^{(j)}) \leq K$, be the set of all decision alternatives created in Algorithm 1 and given to the* select *operator as defined in Algorithm 3. Then $A^{(t)}$ is an ε-Pareto set of $X^{(t)}$ with bounded size according to equation (3.3), i.e.,*

1. $A^{(t)} \in P^*_\varepsilon(X^{(t)})$
2. $|A^{(t)}| \leq \left(\frac{\log K}{\log(1+\varepsilon)}\right)^{(m-1)}$

Proof:

1. Suppose the algorithm is not correct, i.e., $A^{(t)} \notin P^*_\varepsilon(X^{(t)})$ for some t. According to Def. 17, this occurs only if some $x = x^{(\tau)}$, $\tau \leq t$ is (1) not ε-dominated by any member of $A^{(t)}$ and not in $A^{(t)}$ or (2) in $A^{(t)}$ but not in the Pareto set of $X^{(t)}$.

   Case (1): For $x = x^{(\tau)}$ not being in $A^{(t)}$, it can either have been rejected at $t = \tau$ or accepted at $t = \tau$ and removed later on. Removal, however, only takes place when some new x' enters A, which dominates x (line 6) or whose box value dominates that of x (line 4). Since both relations are transitive, and since they both imply ε-dominance, there will always be an element in A which ε-dominates x, which contradicts the assumption. On the other hand, x will only be rejected if there is another $x' \in A^{(\tau)}$ with the same box value and which is not dominated by x (line 10). This x', in turn, ε-dominates x and – with the same argument as before – can only be replaced by accepting elements which also ε-dominate x.

   Case (2): Since x is not in the Pareto set of $X^{(t)}$, there exists $x' = x^{(\tau')}$, $\tau' \neq \tau$, $x' \in X^{*(t)}$ with $x' \succ x$. This implies $\mathsf{box}(x') \succ \mathsf{box}(x)$ or $\mathsf{box}(x') = \mathsf{box}(x)$. Hence, if $\tau' < \tau$, x would not have been accepted. If $\tau' > \tau$, x would have been removed from A. Thus, $x \notin A^{(t)}$, which contradicts the assumption.

2. The objective space is divided into $\left(\frac{\log K}{\log(1+\varepsilon)}\right)^m$ boxes, and from each box at most one point can be in $A^{(t)}$ at the same time. Now consider the

$\left(\frac{\log K}{\log(1+\varepsilon)}\right)^{(m-1)}$ equivalence classes of boxes where – without loss of generality – the boxes of each class have the same coordinates in all but one dimension. There are $\frac{\log K}{\log(1+\varepsilon)}$ different boxes in each class constituting a chain of dominating boxes. Hence, only one point from each of these classes can be a member of $A^{(t)}$ at the same time. □

As a result, Algorithm 2 and Algorithm 3 use finite memory and successively update a finite subset of decision alternatives that ε-dominate all alternatives generated so far, thereby solving both the memory bound and the diversity problem. For Algorithm 3, it can be additionally guaranteed that this subset contains only elements which are not dominated by any of the generated vectors. Note that specific bounds on the objective values are *not* used in the algorithms themselves and are not required to prove the convergence. They are only utilized to prove the relation between ε and the size of the archive given in the second claim. The invariance properties of the algorithms assure the monotonicity of the sequence of archived solution sets, which finally solves the third issue, the convergence problem.

3.3 Simulations

This section presents some simulation results to demonstrate the behavior of the proposed algorithms for two example multiobjective optimization problems. We use instances of the conceptual evolutionary algorithm (specified in Algorithm 1) with a common generate operator and examine different select operators. An isolated assessment of the selection strategy requires the generator to act independently from the archive set $A^{(t)}$ to guarantee that exactly the same sequence of points is given to the selection operator for all different strategies. Beyond As the exact implementation of the generator is irrelevant, we use standard multiobjective EAs here and use the decision alternatives in the sequence of their generation as input for the different selection operators.

3.3.1 Convergence Behavior

At first, we are interested in how different selection strategies affect the *convergence* of the sequence $(A^{(t)})$. As a test problem, a two-objective knapsack problem with 100 items is taken from Zitzler and Thiele (1999). The low number of decision variables is sufficient to show the anticipated effects, and it was found advantageous for visualization and comparison to be able to compute the complete Pareto front $f(X^*)$ beforehand via Integer Linear Programming.

The points given to the selection operator are generated by a standard NSGA-II with population size 100, one-point crossover with crossover probability equal to one, and bit-flip mutations (with probability $4/n = 0.04$). Figure 12 shows the output $A^{(t)}$ of sample runs for the different instances after

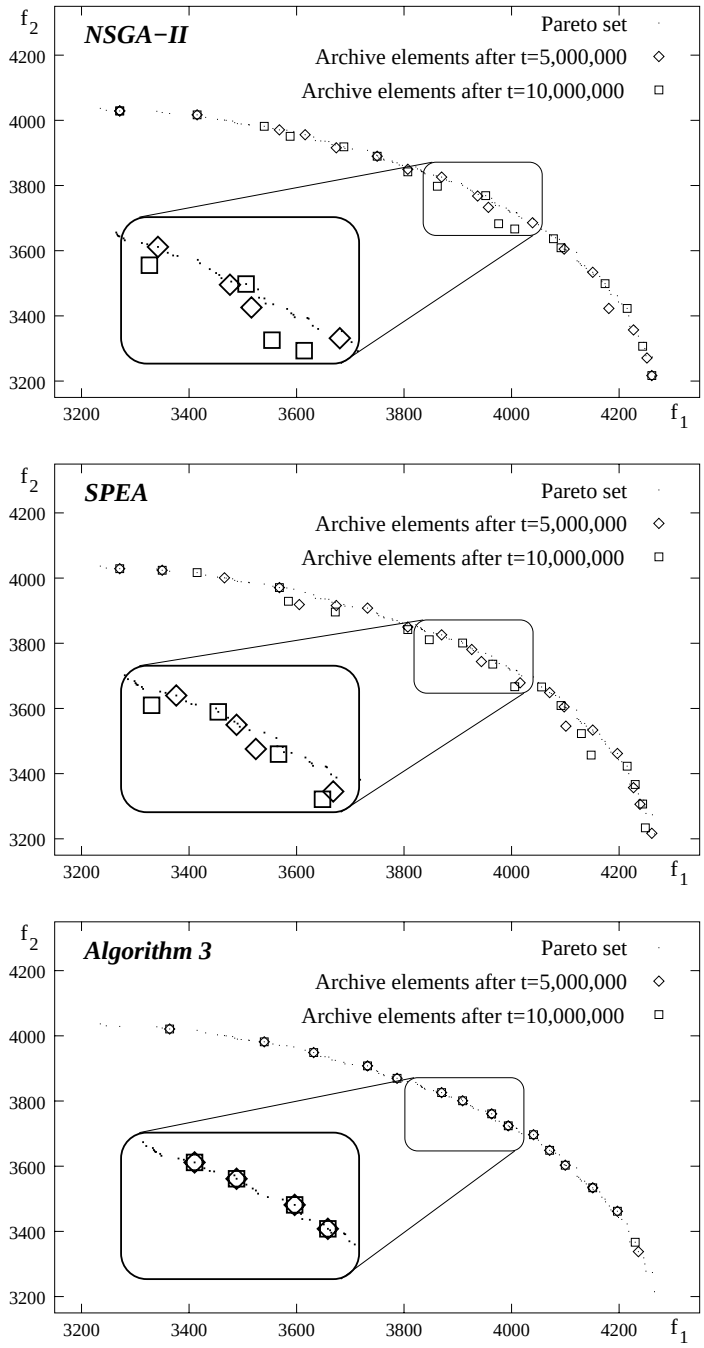


Fig. 12: Objective space of the knapsack problem, the dots show the elements of the Pareto front $f(X^*)$. The different figures correspond to different instances of the select operator: NSGA-II (top), SPEA (middle), and Algorithm 3 (bottom). Each figure shows the objective values for each elements of the solution set $A^{(t)}$ are shown, for $t = 5,000,000$ (with diamonds) and for $t = 10,000,000$ (with boxes). A subset of the samples is highlighted to visualize the negative effect of losing Pareto-optimal solutions in many current archiving/selection schemes.

$t = 5,000,000$ and $t = 10,000,000$ iterations (generated objective vectors), using selection operators from SPEA, NSGA-II (both with maximum archive size of 20) and Algorithm 3 with $\varepsilon = 0.01$.

It is clearly visible that both the archiving (selection) strategies from SPEA and NSGA-II suffer from the problem of partial deterioration: Non-dominated points – even those belonging to the "real" Pareto set – can be lost, and in the long run even be replaced by dominated solutions. This is certainly not desirable, and algorithms relying on these strategies cannot claim to be convergent, even if the generator produces all elements of the Pareto set X^*. In contrast, Algorithm 3 is able to maintain an ε-Pareto set of the generated solutions over time.

The number of function evaluations in this experiment is high, but necessary to produce all Pareto-optimal points in this test case, especially at the extremes of the Pareto front. The non-convergent algorithms can even run infinitely long without converging the Pareto set, although all Pareto-optimal points are generated repeatedly.

3.3.2 Distribution Behavior

In order to test for the *distribution* behavior only those candidates are taken into account which fulfill the requirements for convergence: Rudolph's and Agapie's algorithm AR-I and Algorithm 3. As a test case the following continuous three-dimensional three-objective problem is used:

Maximize $f(x) = (f_1(x),\ f_2(x),\ f_2(x))$,

$$\begin{aligned} f_1(x) &= 3 - (1 + x_3)\cos(x_1\pi/2)\cos(x_2\pi/2) \\ f_2(x) &= 3 - (1 + x_3)\cos(x_1\pi/2)\sin(x_2\pi/2) \\ f_3(x) &= 3 - (1 + x_3)\cos(x_1\pi/2)\sin(x_1\pi/2) \end{aligned} \tag{3.6}$$

subject to $0 \le x_i \le 1, \quad i \in \{1, 2, 3\}$.

The Pareto front of this problem is a surface, a quadrant of the hypersphere of radius 1 around (3, 3, 3). For the results shown in Figure 13 the real-coded NSGA without fitness sharing, crossover using SBX (Simulated Binary Crossover, Deb and Agrawal 1995) with distribution index $\eta = 5$ and population size 100 was used to generate the candidate solutions. The distribution quality is judged in terms of the ε-dominance concept, therefore a discretization of the objective space into boxes (using Algorithm 4 with $\varepsilon = 0.05$) is plotted instead of the actual Pareto set. From all boxes intersecting with the Pareto set the non-dominated ones are highlighted. For an ε-approximate Pareto set it is now sufficient to have exactly one solution in each of those non-dominated boxes. This condition is fulfilled by the algorithm using the selection strategy Algorithm 3, leading to an almost symmetric distribution covering all regions. The strategy of AR-1, which does not discriminate among non-dominated points, is sensitive to the sequence of the generated solution and fails to provide an ε-

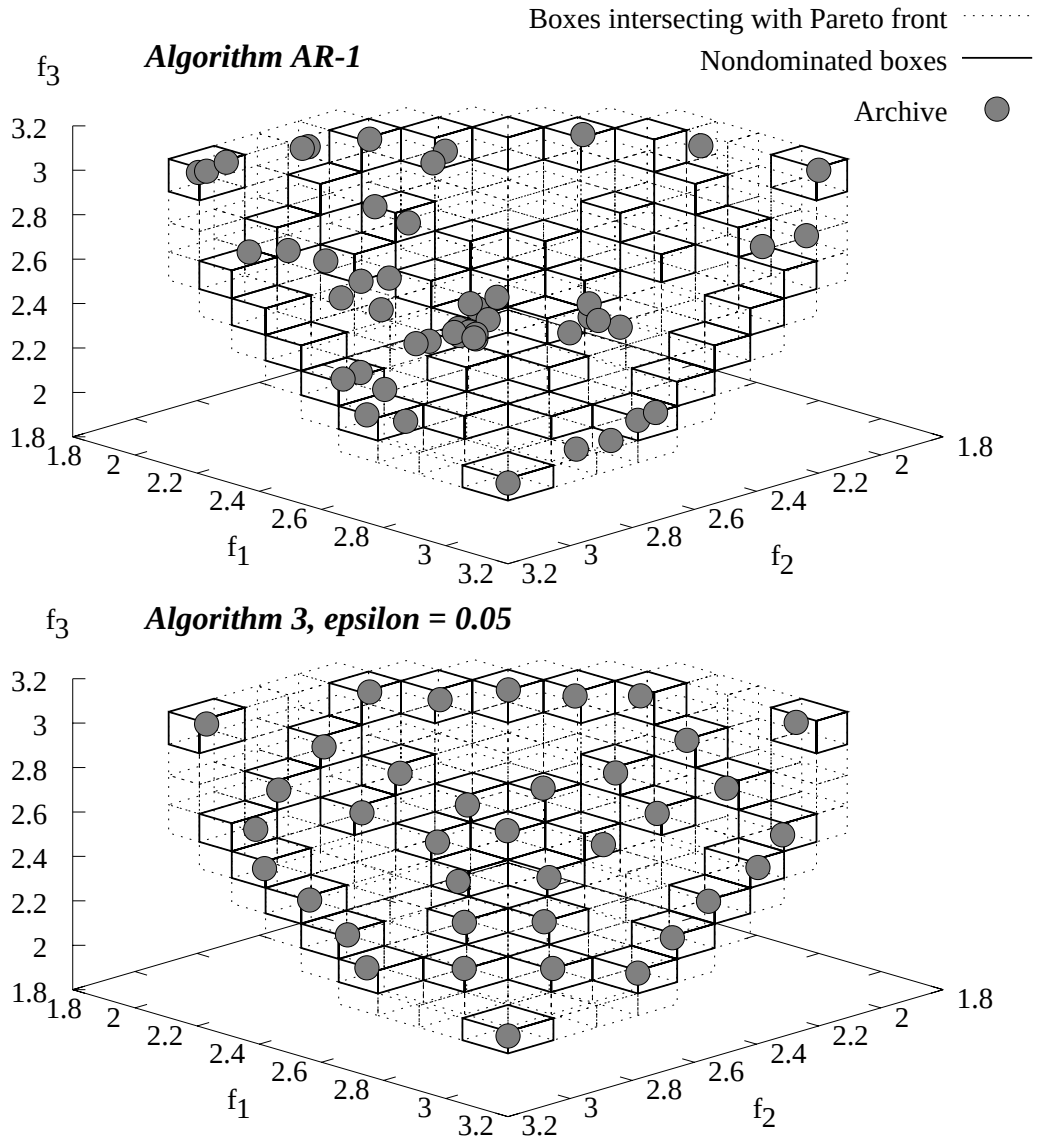


Fig. 13: Objective space of problem (3.6). The discretization into boxes according to Algorithm 4 is indicated by showing, in dashed lines, all boxes that intersect with the Pareto front $f(X^*)$. The non-dominated boxes are drawn in bold lines. The circles correspond to the output A of different instances of the conceptual evolutionary algorithm, Algorithm 1, using the same generate operator. The data set of the upper figure was created using a selection operator according to algorithm AR-1; for data set of the lower figure, a the selection operator according to Algorithm 3 was used.

approximation of the Pareto set of similar quality even with an allowed archive size of 50.

Looking at the graphs of Algorithm 3, one might get the impression that not all regions of the Pareto set are equally represented by archive members. However, these examples represent *optimal* approximations according to the

concepts explained in Section 3.2. They are not intended to give a uniform distribution on a (hypothetical) surface, which might not even exist, as in the discrete case.

3.3.3 Results

The simulation results support the claims of the preceding sections. The archive updating strategy plays a crucial role for the convergence and distribution properties. The key observations are:

- Rudolph's and Agapie's algorithms guarantee convergence, but have no control over the distribution of points.
- The current MOEAs designed for maintaining a good distribution do not fulfill the convergence criterion, as has been demonstrated for SPEA and NSGA-II for a simple test case.
- The algorithms proposed in Section 3.2 fulfill both the convergence criterion and the desired distribution control.

3.4 Possible Extensions

The above baseline algorithms can be extended in several useful ways. In the following, some of these extensions and variations are listed and briefly discussed.

3.4.1 Other Definitions of ε-Dominance

The convergent algorithms can also be implemented with a different definition of ε-dominance. For example, with the dominance definition given in (3.4), grids are uniformly sized in the search space. Although the size of the generated Pareto-optimal set will be different from that presented earlier, the algorithms given so far also maintain the convergence and preserve the diversity.

Although an identical ε value is suggested in the definition of ε-dominance, a different ε_i can be used for each coordinate of the objective space. This way, different precisions among the obtained Pareto-optimal vectors can be achieved in different criteria. The upper bound of the number of Pareto-optimal solution presented above will get modified accordingly.

3.4.2 Guaranteeing Minimum Distances

The ε-dominance definition and the diversity preservation through grids allow a diverse and well convergent set of Pareto-optimal vectors to be obtained by the proposed algorithms. Although diversity among the elements is ensured, the distance between the obtained neighboring Pareto-optimal objective vectors may not be uniform. It is guaranteed by the proposed algorithms that one box

will have only one solution. But in practice, two vectors lying on two neighboring boxes may lie very close to each other. To ensure a good diversity among neighboring vectors, Algorithm 3 may be modified in the following manner. In addition to discouraging two vectors to lie on the same box, the vectors can also be discouraged to lie on the *even* numbered boxes. This way, vectors can only lie on the alternate boxes, thereby ensuring a minimum difference of ε in each objective function value between two neighboring Pareto-optimal vectors.

3.4.3 Steering Search by Defining Ranges of Non-acceptance

In most multiobjective optimization problems, a decision-maker plays an important role. If the complete search space is not of importance to a decision-maker, the above algorithms can be used to search along preferred regions. The concept of ε-dominance will then allow pre-specified precisions to exist among the obtained preferred Pareto-optimal vectors.

3.4.4 Fixed Archive Size by Dynamic Adaptation of ε

Instead of predetermining an approximation accuracy ε, one might ask whether the algorithm would be able to dynamically adjust its accuracy, so as to always maintain a set of vectors of a given magnitude. A concept like this is implemented in PAES (see Section 3.1), where the hyper-grid dividing the objective space is adapted to the current ranges given by the non-dominated vectors. However, PAES does not guarantee convergence.

Here, two modified versions of the proposed converging algorithms are illustrated. The idea is to start with a minimal ε, which is systematically increased every time the number of archived vectors exceeds a predetermined maximum size.

Maintaining an ε-Approximate Pareto Set In order to generate an ε-approximate Pareto set with a given upper bound a on its size, Algorithm 2 can be modified. After the first pair $g^{(1)}, g^{(2)}$ of mutually non-dominating alternatives have been found, an initial ε is calculated according to Theorem 5 as

$$\varepsilon = K'^{\frac{1}{a^{1/m}}} - 1 \tag{3.7}$$

where a is the maximum archive size. K' is set to the current maximum relative range of the m objectives

$$K' := \max_{1 \le i \le m} \left\{ \frac{u_i}{l_i} \right\} \tag{3.8}$$

where u_i and l_i, $u_i \ge l_i > 0$, are the current maximum and minimum values of objective function i.

From this point onwards, new vectors are accepted according to Algorithm 2, where for each element, a time stamp is recorded. If the archive exceeds its maximum size $a_{\max}$, a larger ε must be chosen, using the new

ranges and the above formulas. By this new ε, all archive elements are again compared in the order of their time stamps. Whenever one element is ε-dominated by an older one, the younger one will be removed. This way, the ranges will always be increased in order to cover the whole extent of the current ε-approximate Pareto set. However, if the range of the actual Pareto set decreases in the later part of the run, there will be no possibility to decrease the ε again without in general losing the property given by Theorem 5.

For ε-dominance definition given in equation (3.4), equation (3.7) becomes

$$\varepsilon = \frac{K'}{a^{1/m}}$$

and K' is calculated as

$$K' = \max_{1 \leq i \leq m} \{u_i - l_i\}.$$

Agapie's and Rudolph's algorithms AR-1 and AR-2 also implement a fixed-size archive, but with a constant $\varepsilon = 0$ during the run. This means that the guaranteed minimal distance of vectors is also zero, hence not guaranteeing to maintain an ε-approximate Pareto set.

Maintaining an ε-Pareto Set In Algorithm 3, a simple modification would be to start with a minimal ε using a first pair of mutually non-dominated vectors as in the previous subsection. Afterwards, the increase of ε is taken care of by joining boxes and discarding all but the oldest element from the new, bigger box.

The joining of boxes could be done in several ways, however for ensuring the convergence property it is important not to move or translate any of the box limits, in other words, the assignment of the elements to the boxes must stay the same. A simple implementation could join every second box, while it suffices to join only in the dimensions where the ranges have been exceeded by the new element. This means that in the worst case an area will be ε-dominated which is almost twice the size of the actual Pareto set in each dimension. A more sophisticated approach joins only two boxes at a time, which eliminates the over-covering, but makes a complicated book-keeping of several different ε values in each dimension necessary.

A Bi-start Strategy In cases where the bounds of the Pareto set are much smaller than the bounds on $f(X)$, both algorithms suffer from their inability to increase the precision again after having reached any level of coarseness. In the worst case, they might end up with only one solution ε-dominating the whole Pareto set using a rather large ε.

We illustrate how to use our proposed algorithms to maintain an ε-approximate Pareto set or an ε Pareto set, respectively, with a maximum

predefined cardinality $a_{\max}$. For this a two-step strategy is followed: At first, one of the two dynamic algorithms of the previous section is used to get a rough approximation of the Pareto set. From their results, the ranges of the Pareto set in the objective space can be determined and used to calculate a fixed ε for a second run of the algorithm. Of course, one may use different ε_i for the different objectives. In the second run the only required change to the selection operator is that it never accepts any vectors outside the ranges determined by the first run and hence ensuring that the size of the solution set does not exceed the limit $a_{\max}$.

3.5 Summary

In this chapter, we discussed the ε-(approximate) Pareto set as one possible concept for specifying the desired outcome for evolutionary multiobjective optimization under typical memory constraints. The proposed concept is

- theoretically attractive as it helps to construct algorithms with the desired convergence and distribution properties, and
- practically important as it works with a solution set with bounded size and predefined resolution.

Based on the concept of ε-dominance, the first selection strategy have been constructed that enable and algorithm to maintain an ε-Pareto set of the generated solutions at any time.

As we have exclusively dealt with the selection operator so far, all statements had to be done with respect to the generated solutions only. In order to claim convergence to the Pareto set of the whole search space, one has to include the generator into the analysis. With appropriate assumptions (non-vanishing probability measure for generating of all search points at any time step), this step is trivial: the probability of not creating a specific point goes to zero as t goes to infinity. Analogously to Hanne (1999) or Rudolph and Agapie (2000), results on the limit behavior such as almost sure convergence and stochastic convergence to an ε-Pareto set can be derived in this way.

Though the limit behavior might be of mainly theoretical interest, it is of very high practical relevance that now the problem of partial deterioration, which was imminent even in the elitist MOEAs, can be solved. Using the proposed archiving strategy, the user can be sure to have both, a representative, well distributed approximation, and a true elitist algorithm in the sense that no better solution had been found and subsequently lost during the run. Such an algorithm, called ε-MOEA, has recently been employed by Deb et al. (2003). Besides its favorable bahavior regarding convergence and distribution of solutions, the authors report empirical running time savings of the ε-MOEA of at least one order of magnitude compared to other MOEAs. This is due to its

simple and efficient archiving strategy based on the ε-dominance concept, in comparison to the complex selection strategies of, e.g., SPEA and NSGA-II, which were used in the other algorithms of this study.

Interesting behavior occurs when there are interactions between the archive and the generator. Allowing the archive members to take part in the generating process has empirically been investigated, e.g., by Laumanns et al. (2000, 2001) using a more general model and a parameter called *elitism intensity*. Now, the theoretical foundation is also given so that the archive members are really guaranteed to be the best solutions found.

4

Running Time Analysis

The analysis of the limit behavior in the previous chapter led to very general results, as they were derived independent of the variation operator and independent of the objective function. We proved invariance properties of selection schemes that guarantee, under very mild assumptions, the convergence to the algorithm to a well-defined representative subset of the Pareto set. Results of this generality are necessarily not very strong in the sense that nothing could be said about the convergence time of the algorithms. To judge the efficiency of an algorithm, it is important to make statements about its time complexity for solving a certain problem. This is subject of the running time analysis presented here. Such a running time analysis depends on the specific algorithm and the specific optimization problem at hand. Thus, generality has to be sacrificed in order to arrive at strong and more detailed results.

This chapter contains rigorous running time results of evolutionary algorithms applied to some artificial pseudo-Boolean multiobjective optimization problems. Different population-based algorithms, the simple evolutionary multiobjective optimizer SEMO and two improved versions, FEMO and GEMO, are proposed and analyzed. The analysis is carried out on two bi-objective model problems, LOTZ (Leading Ones Trailing Zeroes) and COCZ (Count Ones Count Zeroes) as well as on the scalable m-objective versions mLOTZ and mCOCZ. For the analysis, we develop and apply two general tools, an upper bound technique based on a decision space partition, and a randomized graph search algorithm. Both techniques facilitate the analysis considerably.

Results on the running time of the different population-based algorithms and for an alternative approach, a multistart (1+1)-EA based on the ε-constraint method, are derived. Main motivation for this analysis is to investigate whether the use of a population is beneficial in solving multiobjective problems, specif-

ically, whether a population-based EA searching concurrently for all optimal solutions in a single run is more efficient than separate runs of a (1+1)-EA searching for different optimal solution separately? The results of this chapter show that for many of the problems considered, the simple algorithm SEMO is as efficient as the (1+1)-EA. For some problems, the improved variants FEMO and GEMO are provably better.

4.1 Methodology and Related Work

A recent overview of the theoretical analysis of evolutionary algorithms is given by Beyer, Schwefel, and Wegener (2002). A major part of this theory is the running time analysis, which addresses the question of how long a certain algorithm takes to find the optimal solution for a specific problem or a class of problems. Such an analysis typically contains the following ingredients:

1. Simple, well-defined algorithms, which are simple instances or stochastic models of EAs,
2. Sample problems (or problem classes), which the algorithms are applied to, and
3. Analytical methods and tools, which are used for the study of the algorithms.

In case of a single objective and discrete search spaces, several results have been achieved regarding the optimization of pseudo-Boolean functions, i.e., real-valued functions with Boolean inputs. Following first results by Mühlenbein (1992) and Rudolph (1997a), a wide range of such problems were treated by Droste et al. (1998, 2002), who successfully applied and considerably extended analytical methods from the field of randomized algorithms. Modeling the EA as a Markov process, Garnier et al. calculated the distribution of the first hitting time of the optimum for the COUNTONES problem (Garnier et al. 1999) and for long-path problems (Garnier and Kallel 2000). He and Yao derived bounds for the expected running time using drift analysis (2001), and exact expressions for the first hitting times of population-based EAs directly from the transition matrix of the associated Markov chains (2002).

In the multiobjective case, only few theoretical results are available. Most studies were concerned with the limit behavior, which was discussed in the previous chapter. Only recently, Scharnow et al. (2002) provided a running time analysis for a (1+1)-EA on the shortest path problem and showed that a multiobjective formulation of the problem can reduce the time to find the single optimum considerably. The first running time analysis of population-based EAs on a multiobjective problem with conflicting objectives was given in Laumanns et al. (2002) for a simple bi-objective model problem.

To achieve running time results for different multiobjective evolutionary algorithms and for different problem scenarios, we proceed as follows:

- Two pseudo-Boolean model problems, which are scalable in the number of decision variables and number of objectives, are introduced.
- Simple individual-based and population-based multiobjective EAs are defined.
- Methods are developed, how population-based EAs can be analyzed in a multiobjective setting.
- Complexity results are derived in terms of bounds of the running time of the different algorithms that hold with high probability, as well as bounds on the expected running time.

After the multiobjective example problems are introduced in Section 4.2, Section 4.3 presents the simple evolutionary multiobjective optimizer, SEMO. A simple and easy to analyze baseline algorithm for multiobjective optimization problems has been missing so far, and SEMO is presented to fill this gap. The analysis of SEMO reveals a similar performance on the bi-objective problems as a (1+1)-EA using multi-starts. In this context, a theorem is developed that can help bounding the running time for a general class of elitist population-based EAs on multiobjective problems. In Section 4.4, we propose two algorithmic improvements, a fair sampling strategy that accelerates the exploration of the optimal set, and a greedy selection mechanism that leads to a faster progress towards the optimal set. With these two improvements, implemented in the algorithms FEMO and GEMO, we are able to prove that population-based algorithms can actually have a lower running time than the (1+1)-EA. The obtained results are generalized to higher dimensional objective spaces in Section 4.5: We derive and apply a theorem on a general graph search procedure that models the behavior of the fair sampling algorithms on the set of optimal solutions. Section 4.6 discusses the changes and difficulties when independent-bit mutations are used instead of one-bit mutations as considered before in the analysis of the different algorithms. Finally, Section 4.7 summarizes the results obtained in this chapter and discusses how these results can lead to a more efficient usage of multiobjective EAs for other problem domains.

4.2 Two Example Problems

This chapter concentrates on optimization problems that are binary decision problems with n variables and $m \geq 2$ objectives. All objective functions are to be maximized. The goal is to find a representation of the Pareto set $X' \subseteq X^*$ such that $f(X') = Y^*$, i.e., every Pareto-optimal objective vector is found together with at least one corresponding decision vector. An alternative and more compact problem formulation, according to the ε-dominance concept of Chapter 3, is to find an ε-Pareto set for $\varepsilon = 0$.

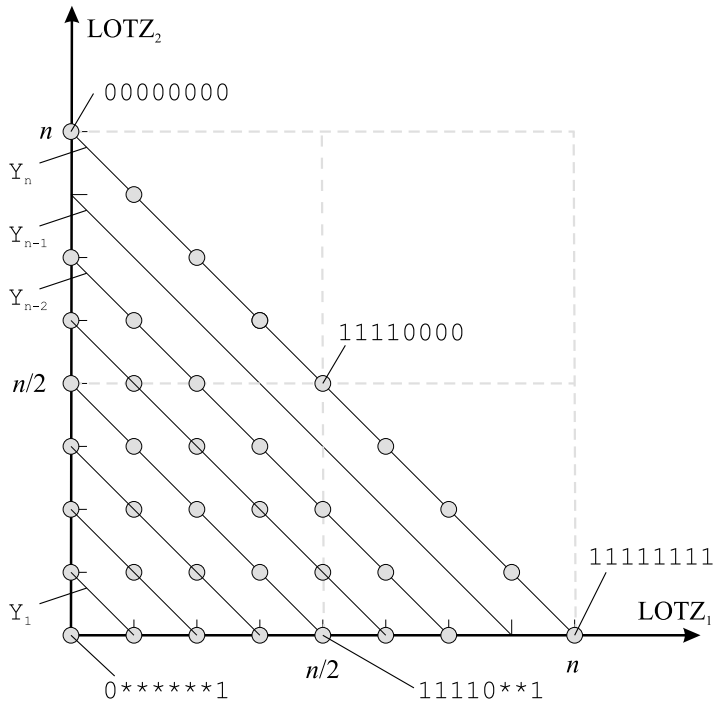


Fig. 14: Objective space of the LOTZ problem with $n = 8$

For the analysis, we assume that the running time of an algorithm equals the number of necessary evaluations of the objective function. As the algorithms defined in the following do not have an explicit stopping rule, we are interested in the running time until all elements of the Pareto front have been identified and are contained in the internal memory of the algorithm, together with one corresponding Pareto-optimal decision vector each.

Unless stated otherwise, the variation operator considered for all algorithms in this chapter is the so-called one-bit mutation. A new decision vector is produced by choosing a random position $j \in \{1, \ldots, n\}$ and replacing the value x_j by $1 - x_j$. All other decision variables remain unchanged.

The first example problem for this analysis is the LOTZ problem. The abbreviation LOTZ stands for "Leading Ones, Trailing Zeroes" and means that we want to simultaneously maximize the number of consecutive ones at the beginning and the number of consecutive zeroes at the end of a bit-string. The first component, the LEADINGONES function, has been analyzed in detail by Rudolph (1997a) and Droste et al. (2002).

Def. 18: *The pseudo-Boolean function* $\mathrm{LOTZ} : \{0, 1\}^n \to \mathbb{N}^2$ *is defined as*

$$\mathrm{LOTZ}(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \right)$$

The objective space of LOTZ can be partitioned into $n + 1$ sets $Y_i, i = 0, \ldots, n$ (see Figure 14). The index i corresponds to the sum of both objective

values, i.e., $(f_1, f_2) \in Y_i$ if $i = f_1 + f_2$. Obviously, Y_n represents the Pareto front Y^*. The sub-domains X_i are defined as the sets containing all decision vectors which are mapped to elements of Y_i. They are of the form $1^a 0 *^{(n-i-2)} 10^b$ with $a + b = i$ for $i < n$, and $1^a 0^b$ with $a + b = n$ for X_n. The asterisk (*) is used as a wildcard symbol and indicates that the corresponding bits can be chosen arbitrarily as zero or one.

The cardinality of the Pareto set $X^* = X_n$ is $|X_n| = n + 1$ and we also have $n + 1$ Pareto optimal objective vectors as $|Y_n| = n + 1$. The next set Y_{n-1} is empty. For the remaining sets with $i = 0, \ldots, n - 2$ we have $|Y_i| = i + 1$ and $|X_i| = |Y_i| \cdot 2^{n-2-i}$. As a consequence, the decision space X contains 2^n different elements, which are mapped to $|Y_n| + \sum_{i=0}^{n-2} |Y_i| = 1/2 \cdot n^2 + 1/2 \cdot n + 1 = O(n^2)$ different objective vectors.

The LOTZ problem has a particular feature: all non-Pareto-optimal decision vectors only have one-bit Hamming neighbors that are either better or worse, but never incomparable to it. This fact facilitates the analysis of the population-based algorithms, which certainly cannot be expected from other multiobjective optimization problems. Therefore, we additionally present another simple multiobjective problem where this condition does not hold. The COCZ problem is a multiobjective extension of the COUNTONES problem and is defined below. The problem consists of two parts: a cooperative part (the first half of the bit-string) and a conflicting part (the second half of the bit-string). In the cooperative part, the objective is to maximize the number of ones in both functions. In the conflicting part, the first objective is to maximize the number of ones and the second objective is to maximize the number of zeroes. The single-objective COUNTONES problem has been extensively studied in the literature (Mühlenbein 1992, Rudolph 1997a, Droste et al. 1998, Garnier et al. 1999).

Def. 19: *The pseudo-Boolean function* COCZ : $\{0, 1\}^n \to \mathbb{N}^2$ *is defined as*

$$\mathrm{COCZ}(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} x_i, \quad \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} (1 - x_i) \right)$$

where $n = 2 \cdot k$ *and* $k \in \mathbb{N}$.

Also for the COCZ, a partition of the objective space (see Figure 15) provides an easy understanding of the problem. We distinguish $n/2 + 1$ sets $Y_i, i = 0, \ldots, n/2$, where the index i corresponds to the number of ones in the first half of the bit-string. All Y_i contain $n/2+1$ elements which distinguish themselves by the number of ones in the second half of the bit-string. $Y_{n/2}$ represents the Pareto front Y^*. The cardinality of the Pareto set $X^* = X_n$ is $|X_n| = 2^{n/2}$. However, more Pareto-optimal decision vectors map to objective vectors in the middle of the Pareto front than to its borders: while $\binom{n/2}{n/4}$ decision vectors map to the objective vector $(3n/4, 3n/4)$, the objective vectors $(n, n/2)$ and $(n/2, n)$ only have one corresponding element in decision space each.

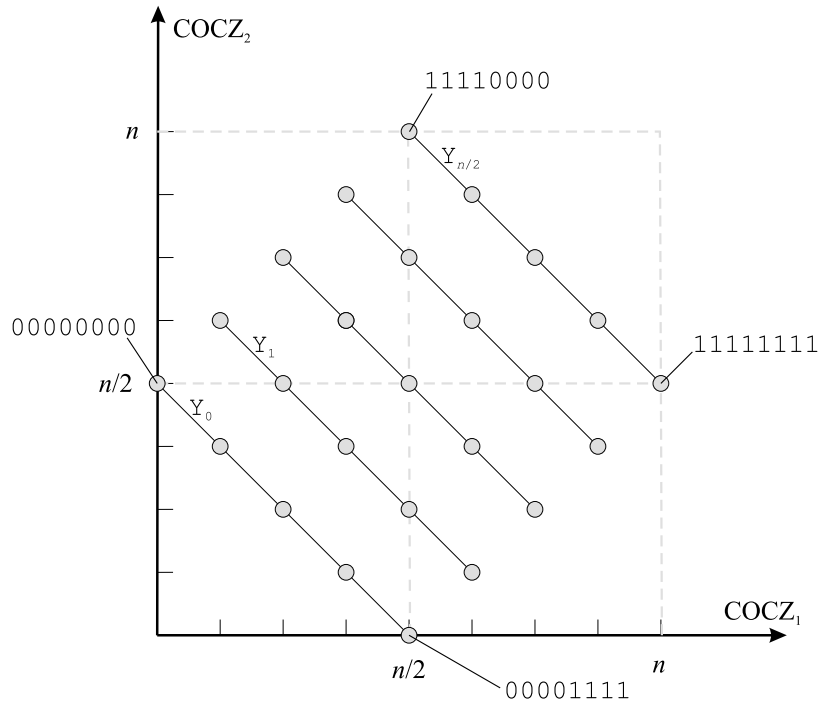


Fig. 15: Objective space of the COCZ problem with $n = 8$

4.3 A Simple Evolutionary Multiobjective Optimizer

The field of evolutionary multiobjective optimization is characterized by a vast variety of EA variants with specialized operators of increasing complexity (Deb 2001; Coello Coello et al. 2002). However, simple algorithms, which can serve as baseline algorithms for comparisons or theoretical analysis, are missing. The necessity for a baseline algorithm was already pointed out by Knowles and Corne (1999) and motivated the invention of the Pareto Archived Evolution Strategy (PAES). The PAES makes use of a complex archiving and selection logic and is therefore difficult to analyze theoretically. We therefore propose and analyze a simple baseline algorithm, which can be seen as a multiobjective generalization of a $(1 + 1)$-EA.

4.3.1 SEMO

The Simple Evolutionary Multiobjective Optimizer (SEMO) represents the simplest instance of a population-based EA for multiobjective optimization. SEMO contains a population of variable size that stores all individuals that are not dominated by any other individuals found so far. At the beginning, the population is initialized with a single element, which is drawn at random from the decision space. In each iteration, one parent individual x is drawn from this population uniformly at random and mutated. The child x' is added to the population, if it is not dominated by any population member and if its objective vector is not

already contained in the population. All individuals that are dominated by the child are in turn deleted from the population.

Algorithm 5 Simple Evolutionary Multiobjective Optimizer (SEMO)

```
Choose an initial individual x uniformly from X
P ← {x}
loop
    Select one element x out of P uniformly.
    Create offspring x' by mutation of x.
    if ∄z ∈ P such that (z ≻ x' ∨ f(z) = f(x')) then
        P ← (P \ {z ∈ P | x' ≻ z}) ∪ {x'}
    end if
end loop
```

4.3.2 Analysis of SEMO on LOTZ

We start our analysis with the SEMO algorithm applied to the LOTZ problem. A run of the SEMO on LOTZ can be divided into two distinct phases: The first phase lasts until the first Pareto-optimal decision alternative has entered the population, and the second phase ends when the whole Pareto set has been found.

Lem. 2: *The expected running time of Algorithm 5 to find the first Pareto-optimal point of LOTZ is* $O(n^2)$.

Proof: During this first phase, the population consists of one individual only, as a mutation changing the objective values yields either a dominating or a dominated individual. Hence, if an offspring is accepted, it will replace the parent from which it was produced. We consider the partition of the search space into distinct subsets X_i as defined above and note that from any subset X_i only points in X_j, $j > i$ are accepted. As there is always a one-bit mutation leading to the next subset, the probability of improvement is at least $1/n$. As there are at most $n-1$ such steps necessary (X_{n-1} is empty) the expected time is at most n^2. □

Lem. 3: *After the first Pareto-optimal point is found, the expected running time of Algorithm 5 until all Pareto-optimal points are found is* $\Theta(n^3)$ *and the probability that the running time is less than* $n^3/c(n)$ *is less than* $(8\mathrm{e}/c(n))^{n/2}$.

Proof: We partition this phase into n different sub-phases. Sub-phase i lasts from the time when i Pareto-optimal solutions have been found to the time when the next solution is found. T_i is a random variable denoting the duration of sub-phase i and the random variable T is the sum of these times. As we always have a contiguous subset of the Pareto set, only the individuals corresponding to the outer points of this subset can create a new Pareto-optimal point. The probability $p_s(i)$ to sample such a candidate in phase i is at least $1/i$ and at

most $2/i$. A subsequent mutation has a success probability of at least $1/n$ and at most $2/n$. Hence, $ni/4 \leq \mathsf{E}[T_i] \leq ni$. As $T = \sum_{i=1}^{n} T_i$, $1/8n^3 + 1/8n^2 \leq \mathsf{E}[T] \leq 1/2n^3 + 1/2n^2$.

To derive a lower bound of the running time which holds with a high probability we consider the run after $n/2$ Pareto-optimal solutions have already been found. In this case the probability to find a new Pareto-optimal solution is at most $4/n^2$. If we allow $n^3/c(n)$ trials, the expected number of successes, S, is at most $4n/c(n)$. With Chernoff's inequality, the probability that we reach the required $n/2 + 1$ successes to find the remaining solutions can be bounded as

$$\mathsf{P}\{S > n/2\} \leq \left(\frac{\mathrm{e}^{\frac{1}{8}c(n)-1}}{(\frac{1}{8}c(n))^{\frac{1}{8}c(n)}} \right)^{4n/c(n)} \leq \left(\frac{8\mathrm{e}}{c(n)} \right)^{\frac{1}{2}n} \tag{4.1}$$

□

From the concatenation of the two phases the following theorem can be derived.

Cor. 2: *The expected running time of SEMO for LOTZ is $\Theta(n^3)$.*

4.3.3 Analysis of SEMO on COCZ and a General Upper Bound Technique

The analysis of the SEMO on the LOTZ problem has been facilitated by the observation that the population does not contain more that one individual until the Pareto set is reached. In this respect, the COCZ is a more realistic problem, because the population will certainly start growing before the Pareto set is reached. Unfortunately, this population growth is hard to analyze in detail and therefore hard to bound. In the worst case, the population might visit the whole objective space and move forward with a broad front of solutions. We can, however, derive the following general upper bound for this worst-case scenario, which holds not only for SEMO, but for a more general class of population-based approaches under the assumption of an elitist selection strategy. The idea behind the following lemma is that it is sufficient for each decision vector to be mutated *once* into a dominating decision vector. The elitist selection strategy then guarantees that this decision vector will be discarded from the population for ever. The result is independent of the sampling (or mating selection) strategy used and depends only on the properties of the variation and replacement selection operator, which can be formalized as:

Lem. 4: **(General upper bound I)** *Let an algorithm be given that iteratively modifies a population P by a sequence of variation and selection operations with the properties*

(1) *For each $y \in Y \setminus Y^*$, the probability that the variation operator applied to any $x \in X$ with $f(x) = y$ produces a dominating decision vector x' with $x' \succ x$ is bounded below by $p(y) > 0$*

(2) *A newly generated decision vector will enter the population P only if it is not dominated by any other element of P.*

(3) *A decision vector is deleted from the population P if and only if a dominating decision vector is included into the population.*

Then the expected number of times the variation operator is applied to non-Pareto optimal decision vectors is bounded above by $\sum_{y \in Y \setminus Y^*} p(y)^{-1}$.

Proof: Consider any objective vector $y' \in Y \setminus Y^*$ and the corresponding set of decision vectors $X' := f^{-1}(y) = \{x \in X | f(x) = y\}$. An element of X' can only undergo variation, if it is present in the population P. Let $T(X')$ denote the total number of times that elements from X' undergo variation until for the first time, an x'' is generated with $f(x'') \succ y'$. Clearly, x'' will enter P and cause all elements of X' being deleted due to property (3). With property (2), this will further mean that no element of X' will ever be accepted again in the population, hence no elements of X' will be subject to variation anymore. As the random variables $T(X')$ are independent it follows with property (1) that $\mathsf{E}[T(X')] \leq p(y)^{-1}$. The summation over all $y' \in Y \setminus Y^*$ leads to the claimed expression. □

In many cases, the summation over all elements in $Y \setminus Y^*$ is impractical because its cardinality is too large. We can, however, also work with larger groups of decision vectors such that a smaller number of groups has to be accounted for. The following lemma is an extension of the previous one for arbitrary partitions of the decision space. It is similar to the fitness level technique (Wegener 2000) of single-objective problems and can be seen as a generalization of this technique for partially ordered objective spaces.

Lem. 5: **(General upper bound II)** *Let the dominated part of the decision space, $X \setminus X^*$ be partitioned into k sets $X_1, \ldots, X_k$ with $\bigcup_{1 \leq i \leq k} X_i = X \setminus X^*$ and $X_i \cap X_j = \emptyset$ for all i, j. Let the dominance relation on sets be defined as*

$$X_i \succ X_j \Leftrightarrow \forall (a, b) \in X_i \times X_j : a \succ b.$$

The sets $d(X_i) := \{X_j : X_j \succ X_i\}$ contain all sets X_j that dominate set X_i. If the algorithm fulfills the same properties as in Lemma 4 and $p(X_i)$ is a lower bound for the probability that a variation applied to an individual $x \in X_i$ produces an individual x' in a dominating decision space subset, i.e., $0 < p(X_i) \leq \min_{x \in X_i} \{\mathsf{P}\left\{x' \in d(X_i) | x \in X_i\right\}$ then the expected number of times the variation operator is applied to non-Pareto optimal decision vectors is bounded above by $\sum_{i=1}^{k} p(X_i)^{-1}$.

Proof: Consider an arbitrary set X_i of the decision space partition. An element of X_i can only undergo variation, if it is present in the population P. Let $T(X_i)$ denote the total number of times that elements from X' undergo variation until for the first time, an $x'' \in d(X_i)$ is generated. Clearly, x'' will enter P and cause all elements of X_i being deleted due to property (3). With property (2), this will further mean that no element of X_i will ever be accepted again in the population, hence no elements of X_i will be subject to variation anymore. As

the random variables $T(X_i)$ are independent it follows with property (1) that $\mathsf{E}[T(X_i)] \leq p(X_i)^{-1}$. The summation over all $i \in \{1, \ldots, k\}$ leads to the claimed expression. □

With the help of this lemma it is now possible to prove upper bounds for different problems, given that an appropriate decision space partition and expressions for the improvement probabilities $p(\cdot)$ can be derived. We demonstrate its use by deriving an upper bound for the expected running time of the SEMO algorithm on the COCZ problem.

Th. 7: *The expected running time of SEMO for COCZ is $O(n^2 \log n)$.*

Proof: We divide the total number of objective function evaluation into those that are required for mutants of non-Pareto-optimal parents and those for mutants of Pareto-optimal parents. Let the parents that are not Pareto-optimal be divided into group $X_{i,j} := \{x \in X \mid f(x) = (n/2 - i + j, n - i - j)\}$, $i, j \in \{0, \ldots, n/2\}$, where the i refers to the Hamming distance from the Pareto set and j to the number of ones in the bit-sting. It is obvious that the $X_{i,j}$ constitute a search space partition according to Lemma 5 and that the SEMO also fulfills the elitist selection conditions (1) and (2). As a result, the total number of mutations of non-Pareto-optimal search points can be bounded by $\sum_{j=0}^{n/2} \sum_{i=1}^{n/2} n/i = O(n^2 \log n)$.

Next, we bound the contribution of mutations of Pareto-optimal points to the total running time. Let k the number of ones of the first Pareto-optimal point found. Now consider a modified SEMO that starts with this point and never accepts any non-Pareto-optimal points (their contribution to the running time has already been bounded) and never accepts any points with more than k ones. The expected time of this algorithm until the remaining Pareto-optimal points have been found is $\sum_{i=1}^{k} n(k+1-i)/i = \Theta(nk \log k)$. For an upper bound, consider the negative assumption that SEMO has to traverse the whole chain starting from $k = n/2$ twice, to move to both ends of the Pareto front. In every case, the claimed bound of $O(n^2 \log n)$ holds. □

4.3.4 Comparing SEMO to a (1+1)-EA using Multistarts

An alternative approach to find a set of Pareto-optimal solutions is the so-called scalarizing approach (Miettinen 1999). The different objective functions are aggregated to form a single-objective surrogate problem, on which a single-objective optimizer can be applied. This scalarization involves parameters to be set that balance the relative importance of the different objectives. The working principle of this approach is to use multiple runs of the single-objective optimizer with different parameter settings such that in each run a different Pareto-optimal solution is found. It is an open problem whether this approach is more efficient compared to a population-based algorithm that is able to search for the whole Pareto front in a single run.

This study addresses this question by a theoretical analysis of the different approaches. As a representative for the single-objective multistart class, we

use the ϵ-constraint method (Haimes et al. 1971). The ϵ-constraint method works by choosing one objective function as the only objective and the remaining objective functions as constraints. By a systematic variation of the constraint bounds, different elements of the Pareto front can be obtained (Chankong and Haimes 1983, p. 285) by solving the constrained single-objective problems

$$\text{maximize} \quad f_1 \tag{4.2}$$

$$\text{subject to} \quad f_i \geq \epsilon_i \quad \forall 2 \leq i \leq m, i \in \mathbb{N}. \tag{4.3}$$

The (1+1)-EA therefore needs a set of different ϵ_i values to be specified. For the following analysis we optimistically assume a best-case scenario, where

- The coordinates of all Pareto-optimal objective vectors $f^{*1}, f^{*2}, \ldots, f^{*|Y^*|}$ are known.
- The single-objective (1+1)-EA, Algorithm 6, is run for each of those f^{*k}, $1 \leq k \leq |Y^*|$, using the constraint bounds $\varepsilon_i = f_i^{*k}$.
- Each of the $|Y^*|$ runs of the (1+1)-EA is independently executed until the optimum is found.

Using information about the optimum might appear unfair, but we are mainly interested in lower bounds for the multistart (1+1)-EA. It is obvious that a realistic variant cannot be better than this one as it has to make at least one run for each Pareto-optimal objective vector to be found, plus a number of run that are redundant because of an non-optimal choice of the ε_i values. In addition, the primary goal of the optimization is to find the Pareto-optimal decision alternatives.

Algorithm 6 Single-objective (1+1)-EA

```
Choose an initial individual x uniformly from X
loop
    Create offspring x' by mutation of x.
    g(x) ← Σ_{f_i ≤ ε_i, 2≤i≤m} (ε_i − f_i(x))          {constraint violation by x}
    g(x') ← Σ_{f_i ≤ ε_i, 2≤i≤m} (ε_i − f_i(x'))         {constraint violation by x'}
    if g(x') < g(x) ∨ (g(x') = g(x) = 0 ∧ f_1(x') > f_1(x)) then
        x ← x'
    end if
end loop
```

The (1+1)-EA proceeds by first minimizing the constraint violation given by the second objective and the different thresholds ε. After the constraint is satisfied, the algorithm turns to optimize the first objective while keeping the second objective value above the constraint boundary. We give an exact expression for the expected running time of the (1+1)-EA on the LOTZ and the COCZ problem.

Th. 8: *The expected running time of the multistart (1+1)-EA is* $\frac{1}{2}(n^3 + n^2)$ *for LOTZ problem and* $\Theta(n^2 \log n)$ *for COCZ.*

Proof: As the Pareto front of the LOTZ problem contains $n + 1$ different elements, the algorithm has to solve $n + 1$ sub-problems with different constraint values. Let T_i, $0 \leq i \leq n$, denote the running time to solve the i-th sub-problem whose optimum is given by the point with $n - i$ leading ones and i trailing zeroes. Each sub-problem can be further divided into n consecutive sub-phases, the first i of which correspond to finding the trailing zeroes starting from the back and the following $n - i$ by finding the leading ones starting from the front. At the beginning of each sub-phase, the bit under concern is with equal probability 0 or 1. If the bit is already set correctly, this sub-phase has length 0, otherwise it follows a geometric distribution with parameter $(n - 1)/n$. Hence, the lengths of all sub-phases are independent and identically distributed with expectation $n/2$. Therefore,

$$\mathsf{E}[T] = \sum_{i=0}^{n} \mathsf{E}[T_i] = (n+1)\frac{n^2}{2} = \frac{1}{2}(n^3 + n^2)$$

holds for the LOTZ problem. For the COCZ problem, the Pareto front contains $n/2 + 1$ different elements. Therefore the algorithm has to solve $n/2 + 1$ sub-problems with different constraint values. In each of the runs, a bitstring has to be produced where the first $n/2$ bits, the cooperative section, are ones. We call this the end of phase 1. Its expected time is $\Theta(n \log n)$ as it is equivalent to solving the COUNTONES problem with $n/2$ bits (Droste, Jansen, and Wegener 1998) and taking into account that on average every second mutation will be realized in the first $n/2$ bits. For the upper bound, consider the worst case, where at the end of the first phase the first $n/2$ bits are set to one and the rest can be any string. Now there are two alternatives. If the constraint is not satisfied, then some of the ones must be mutated to zero. If the constraint is already satisfied, then it might be possible to turn some of the zeroes into ones. In both cases, there are at most $n/2$ such candidate bits, which can flip at a random order, and the expected waiting time for this event is again bounded by $O(n \log n)$. Thus, the expected time for solving one sub-problem is $\Theta(n \log n)$ and with $n/2 + 1$ such problems to solve gives a total expected running time of $\Theta(n^2 \log n)$. □

It follows that running time of the simple population-based optimizer SEMO for both the LOTZ and the COCZ problem is of the same order as the multistart (1+1)-EA in the assumed best-case scenario. In the next section, we investigate whether a population-based approach can be provably better than the (1+1)-EA.

4.4 Two Improved Evolutionary Multiobjective Optimizers

The SEMO was designed as a simple baseline algorithm. The analysis has shown that regarding the asymptotic performance, SEMO is as good as the (1+1)-EA using an ε-constraint method both on the LOTZ and on the COCZ problem. In practise, the simple uniform selection from the population is seldomly used though, and the question arises, whether other selection strategies can improve the running time of the algorithm. In this section, we propose two modifications of the SEMO algorithm, a fair sampling strategy and a greedy selection mechanism. With the fair sampling strategy, the FEMO (Fair Evolutionary Multiobjective Optimizer) can solve the LOTZ problem quicker. Using in addition a greedy selection mechanism, the GEMO (Greedy Evolutionary Multiobjective Optimizer) is also quicker on the COCZ problem.

4.4.1 FEMO and the Fair Sampling Strategy

The main weakness of the SEMO on the LOTZ problem lies in the fact that a large number of mutations are allocated to parents whose neighborhood has already been explored sufficiently. On the other hand, an optimal sampling algorithm would use always the most promising parent at the border of the current population, leading to a running time of $\Theta(n^2)$. Of course, this information is not available in a black box optimization scenario.

The uniform sampling leads to a situation, where the Pareto-optimal individuals have been sampled unevenly depending on when each individual entered the population. The *fair* sampling strategy implemented by FEMO guarantees that at the end all individuals receive about the *same* number of samples.

The FEMO (see Algorithm 7) implements a fair selection strategy by counting the number of times each individual has been mutated (line 6). The sampling procedure deterministically chooses the individual which has produced the least number of offspring so far, ties are broken randomly (line 5).

Algorithm 7 Fair Evolutionary Multiobjective Optimizer (FEMO)

1: Choose an initial individual x uniformly from X
2: $w(x) \leftarrow 0$ {Initialize offspring count}
3: $P \leftarrow \{x\}$
4: **loop**
5: Select one element x out of $\{y \in P | w(y) \leq w(z)\ \forall z \in P\}$ uniformly.
6: $w(x) \leftarrow w(x) + 1$ {Increment offspring count}
7: Create offspring x' by mutation of x.
8: **if** $\nexists z \in P$ such that $(z \succ x' \vee f(z) = f(x'))$ **then**
9: $P \leftarrow (P \setminus \{z \in P | x' \succ z\}) \cup \{x'\}$
10: $w(x') \leftarrow 0$ {Initialize offspring count}
11: **end if**
12: **end loop**

For the analysis of the FEMO on LOTZ we note that the first phase is identical to the SEMO described before. With the following theorem we prove that the total running time of FEMO on LOTZ is bounded by $\Theta(n^2 \log n)$.

The idea is to bound the number times each element that enters P in the second phase will be mutated. Once the first Pareto-optimal point is found, there is exactly one possible parent for each of the remaining n points. We are interested in the number of mutations that must be allocated to *each* of these n parents in order to have at least one successful mutation that leads to the desired child. Lemma 6 and 7 provide upper and lower bounds on the probability that a certain number of mutations per parent are sufficient. In Theorem 9 these probabilities are used to bound the running time of the FEMO algorithm.

Lem. 6: *Let the population P of FEMO applied to LOTZ contain exactly one Pareto-optimal solution and let $c > 0$ be an arbitrary constant. With probability at least $1 - n^{1-c}$, it takes at most $c \cdot n \log n$ mutation trials per solution to generate all remaining n Pareto-optimal solutions.*

Proof: For each individual, the probability that its parent did not generate it within its first $c \cdot n \log n$ mutations is bounded by

$$(1 - 1/n)^{c \cdot n \log n} = (1 - \frac{1}{n})^{cn \log n} = (1 - \frac{1}{n})^{n^{c \log n}} \leq \left(\frac{1}{e}\right)^{c \log n} = \frac{1}{n^c}$$

There are n individuals that must be produced with the given number of trials. These events are independent, so the probability that at least one individual needs more than $c \cdot n \log n$ trials is bounded above by n^{1-c}. □

Lem. 7: *Let $k \in \{1, \ldots, n\}$, $a = k/n$, and $c > 0$ be an arbitrary constant. The probability that $k = a \cdot n$ individuals are produced in $c \cdot n \log n$ mutation steps each is not larger than $\mathrm{e}^{-an^{(1-c-c/n)}}$.*

Proof: The probability that a parent has created a certain offspring within the first $t = c \cdot n \log n$ mutations is $1 - (1 - 1/n)^t$. The probability that this happens independently for a selection of k such parent-offspring combinations can thus be bounded as

$$(1 - (1 - 1/n)^t)^k \leq (1 - \frac{1}{n^{c\frac{n+1}{n}}})^{an} \leq \mathrm{e}^{-\frac{an}{n^{c(n+1)/n}}} = (\mathrm{e}^a)^{-n^{(1-c-c/n)}}$$

□

Th. 9: *With probability at least $1 - O(1/n)$, the running time FEMO needs from the discovery of the first two Pareto-optimal objective vectors of LOTZ until the whole Pareto set has been found lies in the interval $[1/4 \cdot 1/p \cdot n \log n, 2 \cdot 1/p \cdot n \log n]$. Hence, $\mathsf{P}\{T = \Theta(1/p \cdot n \log n)\} = 1 - O(1/n)$. Furthermore, $\mathsf{E}[T] = O(1/p \cdot n \log n)$.*

Proof: Let the Pareto-optimal points be indexed according to the order in which they have entered the set P. Let $k \in \{0, \ldots, n\}$ be the index of the individual that required the largest number of mutations to be produced. We apply Lemma 6 with $c = 2$ and notice that this individual k did not need more than $2/p \cdot \log n$ trials with probability $1 - O(1/n)$.

What remains to be shown for the upper bound is that no node will be *sampled* more than t times during the algorithm. This can be guaranteed since there is always a candidate $x \in P$ with $w(x) \leq t$ (the element that has most recently been added to P). Hence, any element whose weight has reached t will never be sampled again. As there are n such elements, each of which is sampled at most t times, the total number of samples (steps) the algorithm takes does not exceed $T = n \cdot t = 2 \cdot 1/p \cdot n \log n$.

For the lower bound we apply Lemma 7 with $c = 1/2$ and $a = 1/2$ so that $k = n/2$. With a probability of $1 - \mathrm{e}^{-0.5n^{(0.5-0.5/n)}}$ there is an individual in the second half which needs at least $1/2 \cdot 1/p \cdot \log n$ trials. Hence, all individuals in the first half have been sampled at least $1/2 \cdot 1/p \cdot \log n - 1$ times each. Of course, all individuals in the second half must be sampled at least once. The summation over all nodes gives a total number of samples of at least $1/4 \cdot 1/p \cdot n \log n$ with probability $1 - O(1/n)$.

Using the probability bound from Lemma 6, the expected running time can be bounded with $T' = pT/(n \log n)$ and

$$\begin{aligned}
\mathsf{E}[T'] &\leq 1 \cdot \mathsf{P}\left\{0 \leq T' < 1\right\} + 2 \cdot \mathsf{P}\left\{1 \leq T' < 2\right\} + \ldots \\
&\leq 3 + \sum_{c=3}^{\infty} c \cdot \mathsf{P}\left\{T' \geq c - 1\right\} \\
&\leq 3 + \sum_{c=1}^{\infty} (c+2) n^{-c} \\
&\leq 3 + \frac{n}{(n-1)^2} + \frac{2}{n-1}
\end{aligned}$$

as $\mathsf{E}[T] = O(1/p \cdot n \log n)$. □

As discussed earlier, the COCZ problem leads to a possible growth of the population before the Pareto set is found, for the SEMO as well as for the FEMO. Thus it is difficult to derive tight upper and lower bounds for the FEMO on COCZ. Nevertheless, Lemma 5 and the same argumentation as with the SEMO leads to a $O(n^2 \log n)$ upper bound. For the lower bound it is sufficient to note that in order to find the last Pareto-optimal objective vector there is only one bit that must flip, the chance of which is $1/n$. Due to the fair sampling strategy, FEMO will therefore allocate $\Omega(n)$ mutation trials for all $n/2$ other Pareto-optimal members of the population. In summary:

Th. 10: *The expected running time of FEMO for COCZ is bounded above by $O(n^2 \log n)$ and below by $\Omega(n^2)$*

In the next section we propose a further improvement of the algorithm, which avoids the population growth as much as possible.

4.4.2 GEMO and the Greedy Selection Mechanism

The Greedy Evolutionary Multiobjective Optimizer (GEMO) is an extension of the FEMO in order to achieve maximum progress towards the Pareto front. The main idea is to allocate all search effort to offspring of the most recently successful mutant, which is implemented as follows.

As long as only mutually non-dominating individuals are found, the algorithm acts like FEMO, in order to spread out the population, and hence the search effort, fairly and equally. However, when further progress towards the Pareto front is achieved (realized by the fact that a new individual is found that dominates elements of the current population), all other remaining population members are disabled. This means that they cannot produce any offspring for the time being and is implemented by setting their weight to infinity (line 18). When GEMO finally reaches the Pareto front and no further progress is possible, it will again behave like FEMO. Here, it is necessary to re-enable any individual that is re-discovered (line 11). Otherwise, such an individuals would constitute barriers in the objective space that are difficult or, depending on the mutation distribution, impossible to cross.

Due to the special characteristics of the LOTZ problem, the GEMO behaves identically to the FEMO here. On the COCZ problem, the new features of the GEMO algorithm allow us to prove a tight bound of the expected running time. The proof, is again split into the two phases. The idea is that the new greedy selection of the last improved individual reduces the time needed on the way towards the Pareto front considerably.

Th. 11: *The expected running time of GEMO for COCZ is* $\Theta(n^2)$.

Proof: Consider again the two successive phases, where the first phase ends when the first Pareto-optimal point is found. The algorithm starts with a random point. A mutation can cause a step towards the Pareto front (successful step) if in the first half of the bit-sting a zero is flipped, or an incomparable step if any bit in the second half is flipped. If a one in the first half flips, nothing happens, because the mutant is dominated by the parent. If an incomparable mutant is not already in the population, it will be accepted (line 21) while its parent remains, causing the population to grow. A successful mutation leading to a dominating child, however, will delete its parent and all other dominated individuals (line 16). The remaining individuals in the population are temporarily disabled from offspring production (line 18). Now consider the Markov chain with $n/2 + 1$ states, where the state is given by the number of zeroes in the first half of all population members with $w < \infty$. The only possible transitions are from state i to $i - 1$, which happen with probability i/n. The expected time to absorption into state 0, and hence the time to reach the Pareto set, is bounded by $\Theta(n \log n)$.

Algorithm 8 Greedy Evolutionary Multiobjective Optimizer (GEMO)

```
Choose an initial individual x uniformly from X
w(x) ← 0                                          {Initialize offspring count}
P ← {x}
loop
   Select one element x out of {y ∈ P | w(y) ≤ w(z) ∀z ∈ P} uniformly.
   w(x) ← w(x) + 1                                {Increment offspring count}
   Create offspring x′ by mutation of x.
   if ∄z ∈ P such that z ≻ x′ then
      if ∃z ∈ P such that f(z) = f(x′) then
         if w(z) = ∞ then
            w(z) ← 0                              {Reset offspring count}
         end if
      else
         D ← {z ∈ P | x′ ≻ z}                     {Determine individuals dominated by x′}
         if D ≠ ∅ then
            P ← P \ D                             {Delete dominated individuals}
            for all y ∈ P do
               w(y) ← ∞                           {Disable remaining individuals}
            end for
         end if
         P ← P ∪ {x′}                             {Add x′ to population}
         w(x′) ← 0                                {Initialize offspring count}
      end if
   end if
end loop
```

For the second phase, we proceed similarly to the analysis of FEMO on LOTZ and claim the following: if we allocate $c \cdot n$ mutation trials to each element of the Pareto front, each one will have produced its neighbor with a probability depending on c. The failure probability, i.e. the probability that we are not ready after all these cn^2 mutations in total, can thus be bounded above by

$$\sum_{i=1}^{n} \left(1 - \frac{i}{n}\right)^{cn} \leq \sum_{i=1}^{n} \left(\frac{1}{\mathrm{e}^c}\right)^i \leq \frac{1}{\mathrm{e}^c - 1} \leq \left(\frac{1}{2}\right)^{c-1}.$$

To bound the expected total number of mutations T we use the substitution $T' = T/n^2$, and

$$\begin{aligned} \mathsf{E}[T'] &\leq \sum_{c=1}^{\infty} c \cdot \mathsf{P}\left\{c - 1 \leq T' \leq c\right\} \\ &\leq \sum_{c=1}^{\infty} c \cdot \mathsf{P}\left\{T' \geq c - 1\right\} \end{aligned}$$

$$\leq \sum_{c=1}^{\infty} c \cdot \left(\frac{1}{2}\right)^{c-2}$$
$$= 4 \sum_{c=1}^{\infty} c \cdot \left(\frac{1}{2}\right)^{c} = 8$$

leads to an upper bound of $\mathsf{E}[T] = 8n^2$ for the second phase. □

4.5 Higher-dimensional Objective Spaces

In this section, we generalize the two bi-objective problems from the previous sections, LOTZ and COCZ, to arbitrary objective space dimensions. Bounds for the expected running time for the different algorithms are derived for each problem, where the problem size is again determined by the number of decision variables n, while the number of objectives m is considered as a constant. To facilitate the analysis of the fair sampling strategy of the GEMO, we derive a general result on a graph searching process, which serves as a model how GEMO behaves on the Pareto front.

4.5.1 Multiobjective Leading Ones (mLOTZ) Problem

The LOTZ problem can be generalized to an arbitrary even number of objectives m by concatenating $m/2$ bi-objective LOTZ problems of $2n/m$ bits each.

Def. 20: *The pseudo-Boolean function* $m\mathrm{LOTZ} : \{0,1\}^n \to \mathbb{N}^m$ *is defined as*

$$m\mathrm{LOTZ}(x_1, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

with

$$f_k = \begin{cases} \sum_{i=1}^{n'} \prod_{j=1}^{i} x_{j+n'(k-1)/2} & \text{if } k \text{ is odd} \\ \sum_{i=1}^{n'} \prod_{j=i}^{n'} (1 - x_{j+n'(k-2)/2}) & \text{else.} \end{cases}$$

where $m = 2 \cdot m'$, $m' \in \mathbb{N}$, *and* $n = m' \cdot n'$, $n' \in \mathbb{N}$.

The construction principle of the mLOTZ problem is depicted in Figure 16. To solve mLOTZ, we are searching for a representation of the Pareto front with $|Y^*| = (2n/m + 1)^{m/2}$ elements.

Th. 12: *The expected running time of the (1+1)-EA for mLOTZ is* $\Theta(n^{m/2} n^2)$.

Proof: For each of the $|Y^*|$ Pareto-optimal points, we have to find a unique bit-string. For each mutation, there are at least one and at most m bits that can flip for a success. In addition, a bit is set correctly with probability $1/2$ even without mutation, so half of the steps are for free. Hence, the expected

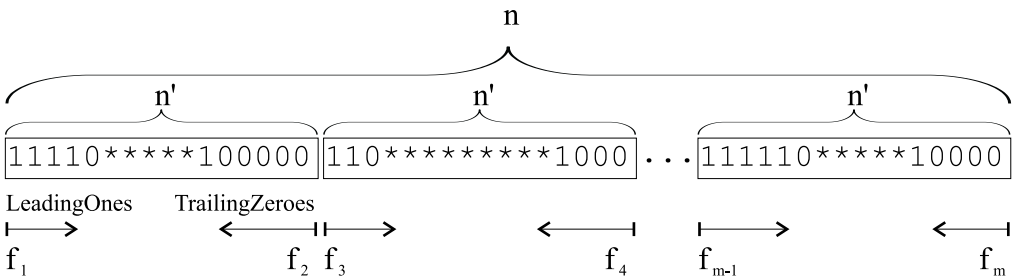


Fig. 16: Schematic view of the mLOTZ problem

running time for each constrained sub-problem is $\Theta(n^2)$, and with the number of $|Y^*| = (2n/m + 1)^{m/2}$ sub-problems to be solved the claim follows. □

To derive an upper bound for the SEMO, we can again make use of the general upper bound of Lemma 5.

Th. 13: *The expected running time of the SEMO and the FEMO applied to mLOTZ is bounded by* $O(n^{m+1})$.

Proof: We have $O((n/m)^m)$ different objective vectors. Each objective vector that is not Pareto-optimal receives on average $O(n)$ mutations until it is improved and deleted forever. With Lemma 5 it follows that the expected total number of mutations of non-Pareto-optimal decision alternatives is bounded by $O(n^{m+1})$. The discovery of the last Pareto-optimal vector takes $O(n/m)^{m/2} \cdot n$ time, which is an upper bound the discovery of all Pareto-optimal vectors. □

For the analysis of the GEMO we again note that the running time is mainly determined by the exploration of the Pareto front. In the general case, the Pareto front can be modeled as a graph, where the nodes correspond to the different Pareto-optimal objective vectors and the directed weighted edges correspond to mutation probabilities. Instead of analyzing GEMO directly, we first define and analyze a more general randomized graph search algorithm, which is similar to a process described and analyzed in Alon (2002). The purpose of this approach is two-fold. First, it gives a more intuitive view of how the different population-based algorithms behave on the Pareto front. Second, it provides a general tool, like the upper bound technique of Lemma 5, which facilitates and shortens the analysis of different real algorithms.

The algorithm assumes that we are given a random starting node v and a random operator $\mathsf{jump} : V \mapsto V$, which returns for each node v a neighbor v' of v with probability $p(v, v')$. The purpose of the following algorithm is to determine V and E using a minimal number of calls to jump.

It will be shown that the above algorithm explores a graph G using $O(\frac{|V|}{p} \log |E|)$ calls to the function jump with high probability, where p is a lower bound on the edge weights. The first lemma bounds the weight of a node during a run of the algorithm.

Algorithm 9 Randomized Graph Search

```
w(v) ← 0
V ← {v_1}; E ← ∅
loop
    Select a node v out of {v' ∈ V | w(v) ≤ w(v') ∀v' ∈ V} uniformly.
    w(v) ← w(v) + 1
    v' ← jump(v)
    if v' ∉ V then
        w(v) ← 0
        V ← V ∪ {v'}; E ← E ∪ {(v, v')}
    end if
end loop
```

Lem. 8: *With probability at most $\Delta \cdot \mathrm{e}^{-\lambda}$, the weight of a node with Δ neighbors exceeds $\frac{\lambda}{p}$ before all neighbors of the node are found.*

Proof: Starting from $w(v) = 0$, the weight increases by 1 after each trial. Let us suppose that the neighbors of v are $\{v_1, ..., v_\Delta\}$. The weight exceeds $\frac{\lambda}{p}$ if it exceeds this value before neighbor v_1 is found *or* before neighbor v_2 is found *or* ... Using the Boole-Bonferroni inequality, we can bound the probability P we are looking for, according to

$$P \leq \sum_{k=1}^{k=\Delta} P_k = \Delta \cdot P_1$$

where P_k denotes the probability that neighbor v_k has not been found yet and $w(v)$ exceeds $\frac{\lambda}{p}$. Now we have

$$P_1 = (1-p)^{\frac{\lambda}{p}} \leq e^{-\lambda} \qquad P \leq \Delta \cdot \mathrm{e}^{-\lambda}$$

□

Lem. 9: *With probability at most $|E| \cdot \mathrm{e}^{-\lambda}$, the weight of some node exceeds $\frac{\lambda}{p}$ before all nodes and edges are found.*

Proof: There are N nodes that must find all their neighbors. The events that the weight of a node exceeds $\frac{\lambda}{p}$ are independent for all nodes. Using the number of neighbors Δ_i of node v_i, we find the probability

$$\sum_{i=1}^{i=|V|} \Delta_i \cdot e^{-\lambda} = |E| \cdot e^{-\lambda}$$

□

Now, we can bound the total running time of the exploration algorithm.

Th. 14: *With probability at least* $1 - |E|^{-c}$*, Algorithm 9 explores all nodes and edges of* G *using not more that* $(c+1)\frac{|V|}{p}\log|E|$ *calls to the function* jump*. The expected number of calls to* jump *is bounded by* $O(\frac{|V|}{p}\log|E|)$.

Proof: With $\lambda = (c+1)\log|E|$ and Lemma 9 we find that with probability $1 - \frac{1}{|E|^c}$, the maximal weight of the nodes in V does not exceed $\frac{(c+1)}{p}\log|E|$ at the time when the whole graph has been explored. Therefore, the function jump has been called at most $|V|\frac{(c+1)}{p}\log|E|$ times. For the expectation, let the random variable T denote the number of calls to jump, divided by $\frac{|V|}{p}\log|E|$. With

$$\begin{aligned}
E(T) &\leq 1 \cdot P\{0 \leq T < 1\} + 2 \cdot P\{1 \leq T < 2\} + \ldots \\
&\leq 3 + \sum_{c=3}^{\infty} c \cdot P\{T \geq c-1\} \\
&\leq 3 + \sum_{c=1}^{\infty} (c+2)|E|^{-c} \\
&\leq 3 + \frac{|E|}{(|E|-1)^2} + \frac{2}{|E|-1} \\
&= O(1)
\end{aligned}$$

the bound on the expected number of calls to jump follows. □

Th. 15: *The running time of the GEMO for mLOTZ is bounded by* $O(n^{m/2} \cdot \frac{m}{2} n \log n) = O(n^{m/2} n \log n)$ *with probability at least* $1 - O(n^{-m})$*. The expected running time is bounded by* $O(n^{m/2} n \log n)$.

Proof: Consider again two phases depending on whether a Pareto-optimal point has been found so far. In the first phase, there are two types of mutations that lead to accepting new points: successful mutations (the new points dominates the old point) and indifferent mutations (the new point and the old point are incomparable). Successful mutations cause the active population to shrink to one element x and increase the objective sum $s = \sum_{i=1}^{m} f_i(x)$ by one. Indifferent mutations can cause the population to grow, but keep s constant. Therefore, at most n successful mutations are needed in the first phase. Since the probability of a successful mutation is always at least $1/n$, the expected running time of the first phase is $O(n^2)$. At the end of the first phase, P contains exactly one element of the Pareto set, and we can describe the behavior in phase 2 by the the model of Algorithm 9. In this case, $|V| = (2n/m+1)^{m/2}$, $|E| \leq m|V|$ and $p = 1/n$. Application of Theorem 14 leads to the claimed bounds. □

4.5.2 Multiobjective Count Ones Problem (mCOCZ) Problem

The idea of the mCOCZ is again a concatenation of multiple bi-objective COCZ problems. However, the bits shall be re-arranged so that the first $n/2$ bits represent the cooperative part, where the number of ones contributes equally to all

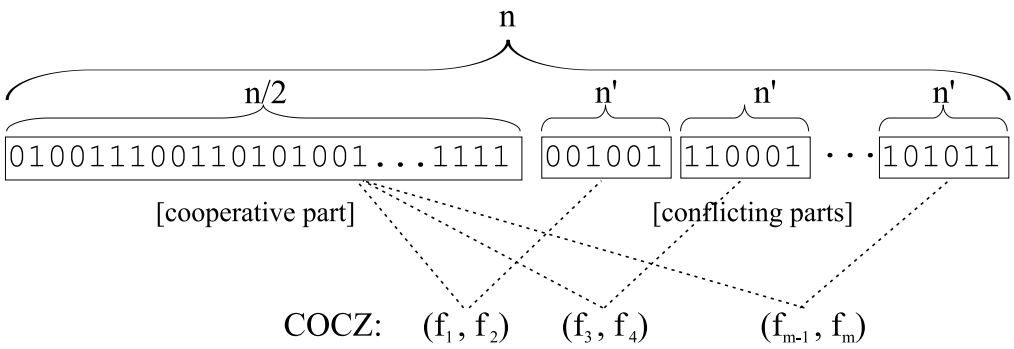


Fig. 17: Schematic view of the mCOCZ problem

objectives. The remaining parts represent the mutual tradeoffs between pairs of objectives.

Def. 21: *The pseudo-Boolean function* $m\text{COCZ} : \{0,1\}^n \to \mathbb{N}^m$ *is defined as*

$$m\text{COCZ}(x_1, \ldots, x_n) = (f_1,\ f_2, \ldots,\ f_m)$$

with

$$f_j = \sum_{i=1}^{n/2} x_i + \begin{cases} \sum_{i=1}^{n'} x_{i+n/2+(j-1)n'/2} & \text{if } j \text{ is odd} \\ \sum_{i=1}^{n'} (1 - x_{i+n/2+(j-2)n'/2}) & \text{else.} \end{cases}$$

where $m = 2 \cdot m'$, $m' \in \mathbb{N}$ *and* $n = m \cdot n'$, $n' \in \mathbb{N}$.

This gives rise to a Pareto front with $|Y^*| = (n/m+1)^{m/2}$ different elements. The construction principle of the mCOCZ problem is depicted in Figure 17.

Th. 16: *For the mCOCZ problem, the following bounds for the expected running times hold:*

- $\Theta(n^{m/2} n \log n)$ *for the (1+1)-EA,*
- $O(n^{m+1})$ *for the SEMO and the FEMO.*

The running time of the GEMO for mCOCZ is bounded by $\Theta(n^{m/2} n \log n)$ *with probability at least* $1 - O(n^{-m})$.

Proof: The (1+1)-EA has to solve $|F^*| = (n/m + 1)^{m/2}$ constrained sub-problems. For each sub-problem we have to at least solve a COUNTONES problem in the first $n/2$ bits, which takes $\Theta(n \log n)$ time. This already proves the lower bound. For the upper bound we pessimistically assume that we then optimize each of the $m/2$ trade-off sections separately. As all bits within the same section are interchangeable, this is equivalent with solving $m/2$ COUNTONES problems of n/m bits, which again takes on average $\Theta(n \log n)$ time and proves the upper bound.

For SEMO and FEMO, we again apply Lemma 5 with a search space partition where each objective vector constitutes its own subset. In each subset, let k denote the Hamming distance from the Pareto set, i.e. the number of zeros among the first $n/2$ bits. For each k there are $|F^*|$ subsets in the partition whose elements have a Hamming distance k to the Pareto set and whose probability of improvement p_k equals k/n (flipping one of the zeroes in the first half of the bit-string). With Lemma 5 we can bound the number of mutations allocated to non-Pareto optimal points by $O(n^{m/2} n \log n)$. On the Pareto front, we want to bound the expected time to find the k-th element under the conditions that $k-1$ elements are already found. Such a bound is given $O(kn)$, and the summation over all k from 1 to $|F^*|$ leads to a total expected running time of $O(n^{m+1})$.

For GEMO, let us again consider two phases depending on whether a Pareto-optimal point has been found so far. In the first phase, there are two types of mutations that lead to accepting new points: successful mutations and indifferent mutations. Successful mutations cause the active population to shrink to one element x and increase the objective sum $s = \sum_{i=1}^{m} f_i(x)$ by one. Indifferent mutations can cause the population to grow, but keep s constant. Therefore, at most n successful mutations are needed in the first phase. Since the probability of a successful mutation is always at least $1/n$, the expected running time of the first phase is $O(n^2)$. At the end of the first phase, P contains exactly one element of the Pareto set, and we can describe the behavior in phase 2 by the the model of Algorithm 9. In this case, $|V| = (n/m+1)^{m/2}$, $|E| \leq m|V|$ and $p \geq 1/n$. Application of Theorem 14 leads to the claimed bound. □

4.6 From One-bit to Independent-bit Mutations

So far, all algorithms in this chapter have been considered with one-bit mutations only as this simplifies the analysis. The disadvantage is that the one-bit mutation operator is less relevant in practise, because it is a local operator and will fail on problems, where jumps of more than one bit are required to proceed the search.

In this section, we discuss the use of the independent-bit mutation operator, i.e. in a mutation step each bit is flipped independently with probability $1/n$. We expect that upper bounds derived by the general technique of Lemma 5 also hold in the case of independent-bit mutations. Our argumentation relies mainly on the success probabilities. We can simply disregard all mutations where not exactly one bit flips. The expected waiting time from one one-bit mutation to the next is $(1-1/n)^{-(n-1)} = O(\mathrm{e})$, which does not change the order of the total expected running time. Lemma 5 then assures us that we can work with the success probabilities alone and completely ignore those mutations, where not exactly one bit flips.

The situation is different when we try to verify more tighter bounds, like

only for the FEMO on the LOTZ problem. Here, the independent-bit mutations can produce incomparable individuals and hence cause the population to grow before it reaches the Pareto set. This problem is somehow similar to the one encountered in the analysis of FEMO on COCZ with the one-bit mutations, and it is a so far unsolved challenge to prove tight bounds for both cases.

The effect of mutations that flip more than one bit has to be taken into account also for the GEMO algorithm. Here, we are faced with the following type of negative event that might enlarge the running time: Offspring can be accepted into the population with a larger Hamming distance from the Pareto set than their parent. If this offspring then happens to be the first in the population to produce a dominating child, then the minimum Hamming distance of the active part of the population can increase. It is therefore necessary to bound the negative effect of this event.

Altogether it can be stated that bounds for the case of independent-bit mutations can be derived for all scenarios considered in this chapter, but that it remains open whether these bounds are tight.

4.7 Summary

In this chapter, running time results were derived for different types of evolutionary algorithms on different pseudo-Boolean multiobjective optimization problems. To achieve this aim, we constructed simple optimization problems as well as simple algorithms, which represent essential ingredients of the analysis.

As model problems for the analysis, we defined mLOTZ and mCOCZ, which are composed of unimodal pseudo-Boolean functions and which are scalable in the number of objectives m. We proposed and analyzed various population-based multiobjective EAs:

- The simple population-based algorithm SEMO, and
- Two improved population-based algorithms, FEMO and GEMO,

and compared their running time against a multi-start (1+1)-EA based on the ε-constraint method. To facilitate the analysis of population-based EAs, we derived two analytical tools,

- A general upper bound technique based on a partition of the decision space and
- A general randomized graph search algorithm.

While the former can be used to bound the search effort in non-Pareto optimal regions, the latter helps to analyze the time spent for the exploration of the Pareto front itself.

The bounds on the expected running times are summarized in Table 4. It was shown that on the bi-objective problems, the SEMO needs a running time of the

	(1+1)-EA	SEMO	FEMO	GEMO
LOTZ	$\Theta(n^3)$	$\Theta(n^3)$	$\Theta(n^2 \log n)$	$\Theta(n^2 \log n)$
COCZ	$\Theta(n^2 \log n)$	$O(n^2 \log n)$	$O(n^2 \log n)$	$\Theta(n^2)$
mLOTZ	$\Theta(n^{m/2} n^2)$	$O(n^{m+1})$	$O(n^{m+1})$	$O(n^{m/2} n \log n)$
mCOCZ	$\Theta(n^{m/2} n \log n)$	$O(n^{m+1})$	$O(n^{m+1})$	$O(n^{m/2} n \log n)$

Tab. 4: Bounds on the expected running time derived in this chapter.

same order as the (1+1)-EA, while it loses its competitiveness with increasing number of objectives. This is mainly due to SEMO's poor ability to expand the population on the Pareto set. The GEMO on the other hand has the lowest running time on all problems, except for the mCOCZ for $m > 2$, where an upper bound of the same order as for the (1+1)-EA could be given. This bound, however, is not necessarily tight bound, which allows to speculate whether GEMO is also quicker on this problem.

To derive tight bounds for the expected running time it is necessary to give lower and upper bounds of the same growth rate. For some of the analyzed cases, the upper bound might be too pessimistic so that it was not possible to prove a matching lower bound. One possibility to approach this problem is to bound the population-size during the run of the algorithm. However, to be able to store the whole Pareto front, a large enough population size must be allowed, implying that in the worst case all possible objective vectors will be visited. If it is not possible to bound the size of the population on its way towards the Pareto set, only the general upper bound of Lemma 5 holds, and the running time advantage of the population-based EAs compared to the (1+1)-EA vanishes in most cases.

Another possibility is to circumvent this problem through the design of the algorithm, which was one motivation for the design of the GEMO. The strategy of the GEMO to focus the search effort on the most recently successful offspring counteracts the tendency of the population to increase, as long as successful mutations are possible. It can therefore be viewed as an algorithm that implicitly notices when the Pareto front is reached and automatically adapts its behavior. The rationale behind the GEMO strategy is that we first want to find the Pareto set on the quickest way possible and only thereafter to spread out the population as quick as possible. This way, the GEMO represents the opposite of the (1+1)-EA, which searches in all different directions from the start, while the behavior of the FEMO and the SEMO is somehow in between these two extremes.

There certainly exist multiobjective problems where it is not easy to traverse the Pareto set, e.g. if its elements are at a large Hamming distance from each other. In such a case it might be necessary to spread out the population early in the search in order to make all Pareto-optimal points reachable. Clearly, the GEMO strategy would be wrong here, because it was designed to prevent exactly this. Nevertheless, it can be argued that such a scenario is anyway the domain of scalarizing approaches like the multi-start (1+1)-EA presented in this

chapter.

It was proven for two types of problems that specific population-based multiobjective optimizers have a provable lower running time than a specific traditional scalarizing method using multistarts of single-objective optimizers. So far, it has often been claimed, but never shown, that the use of a population is beneficial in the context of multiobjective optimization, even without the use of recombination. The results presented in this chapter are the first theoretical evidence for this claim. Instead of the explicit co-operation using traditional recombination operators, implicit cooperation via measuring the successes in the population is used here.

The importance of both, simple baseline algorithms and simple problems, for the development of a theoretical foundation of multiobjective evolutionary algorithms becomes apparent from recent follow-up work in this area. Thierens (2003) presents a heuristic, non-rigorous running time estimation of SEMO on a multiobjective COUNTONES problem, while Giel (2003a, 2003b) provides a rigorous analysis of SEMO with one-bit and independent-bit mutations for the same problem as well as for LOTZ and another test function.

The availability of appropriate methods and tools has shown to be crucial for the analysis. The general upper bound technique presented in this chapter, for instance, allowed us to derive upper bounds for a number of different scenarios and to shorten the proofs considerably. One of the many challenging problems in this area is to give tight bounds for the cases where the population of a multiobjective EA spreads out before the Pareto set is reached.

The theoretical analysis led to the definition of two new selection schemes that so far have not been used in standard MOEAs. The results suggest that their use might be beneficial also in practical applications, which is an interesting area of future research.

5

Applications in Automotive Engineering

So far, this thesis dealt with the analysis of multiobjective evolutionary algorithms. Theoretical results have been obtained concerning their properties and performance. Such results are important to better understand the behavior of the algorithms. This way, their fields of application, but also their limitations can be judged realistically. Nevertheless, the ultimate aim of dealing with optimization algorithms is to apply them to practical real-world optimization problems. Thus, the purpose of this chapter is two-fold. Firstly, we want to address multiobjective optimization problems that are important engineering design problems by itself. Secondly, these case studies shall serve as examples to highlight and demonstrate selected issues of the preceding theoretical investigations on practical applications.

In this chapter, we treat multiobjective decision problems that arise at different stages during the design process of automotive systems. We first propose a generic problem solving procedure, which contains an automated, computational part and an interactive part. The automated part integrates simulation tools with multiobjective optimization techniques based on randomized search algorithms. We then present and describe the simulation environment used to evaluate the different decision alternatives. Simulation tools are frequently used to model various parts of the system that has to be designed, and of its environment, because experiments with the real system are too time consuming or too expensive, and integration into a computer-aided optimization process is difficult. The remaining sections demonstrate the use of the proposed methodology in three case studies of application problems in the automotive industry.

The first application is a design space exploration for road trains. Road trains are a new vehicle concept for the European freight traffic sector, and it is necessary to explore their potential concerning various economic and environmental criteria at a preliminary design stage. For the design space exploration,

a simple approach based on the SEMO algorithm from Chapter 4.3 is used. A parameter optimization of adaptive cruise control systems is the second application area. The task is to devise a filter structure for an optimal controller behavior regarding driving performance, safety and fuel consumption. On the algorithmic side, different instances of the generate operators are applied and compared. The focus is to evaluate which problem representation and variation operators performs best. This performance is assessed based on the results of Chapter 2.4. A system identification problem represents the third case study. The aim is to fit a vehicle dynamics simulation model to data acquired in real driving tests. A simple, but realistic model is needed for a later integration into a vehicle dynamics controller. This case study is an example, where a standard multiobjective EA exhibits the problem of deterioration, which was discussed in Chapter 3.1. Therefore, the new algorithm based on the ε-dominance, Algorithm 3 is applied to guarantee convergence together with the required diversity of design alternatives.

5.1 An Evolutionary Multiobjective Design Framework

Like many areas of engineering design, vehicle development has changed to a more and more complex process in recent years. Engineers have to meet conflicting demands concerning efficiency, performance, costs, etc. The design problems are typically characterized by the presence of multiple decision criteria or objectives. Additional difficulties arise due to increasingly complex system models used in the design process.

Approaches to cope with difficult design problems were typically categorized into experimental and analytical methods. Experimental methods rely on the intuition and experience of the engineer and often follow a trial-and-error principle. Analytical methods represent a more formal and systematic approach, but are normally limited to very simple models. With the advent of sophisticated computational techniques emerged a third group: design methods based on artificial and computational intelligence (Zurada et al. 1994). Techniques like expert systems, fuzzy logic, neural networks, and evolutionary computation allow to incorporate the knowledge of the designer into a computer-aided design process, even with complex system models.

The multiobjective design procedure we adopt here contains the following components:

- A definition of the design space and the objective functions,
- A system model, which provides the mapping of design alternatives to objective values,
- An optimization algorithm, which iteratively samples the design space and tries to improve old or generate new promising solutions automatically, and

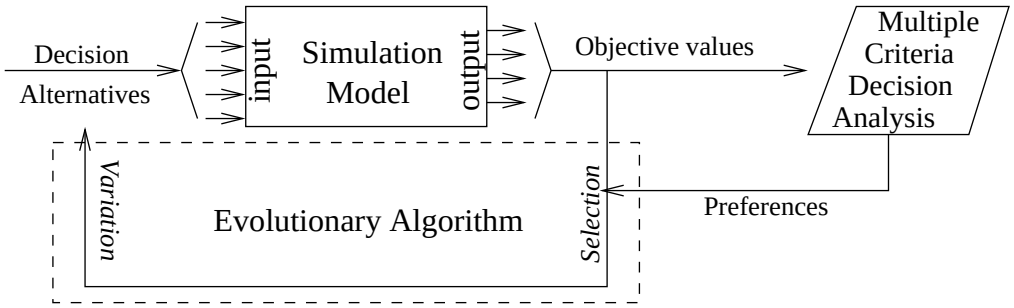


Fig. 18: Integration of system model and multiobjective optimizer in a multiobjective design framework.

- An interface, which allows the decision maker to guide the search process according to his preferences.

Since the design procedure shall be widely applicable, we do not make any assumptions about the system model, such as linearity, differentiability etc. The question, which optimization algorithms are suitable for such a black-box scenario, has been discussed in the introduction of this thesis, motivating the use of evolutionary algorithms.

The design procedure framework is schematically depicted in Figure 18. Each decision alternative is represented by a distinct combination of input parameter values, for which the model can be executed to calculate the corresponding objective values. If the number of different alternatives is small, they can be directly processed further, e.g., by multiple criteria decision analysis (MCDA). If the number of possible combinations is large, an iterative approach is followed using the evolutionary algorithm: From a small set of initial alternatives, the better ones are chosen and used to create new solutions by variation. While the variation step is driven randomly and undirected, the selection step can incorporate preference information obtained from MCDA (second feedback loop).

To simulate and evaluate the behavior of different design alternatives, the traffic simulation tool PELOPS (Ludmann 1998) is used. PELOPS allows to analyze the interactions between vehicle, driver and the environment. It is based on the combination of detailed sub-microscopic vehicle models with microscopic traffic models that enable an investigation of the longitudinal vehicle behavior as well as an analysis of the traffic course. PELOPS contains the most important elements of the traffic system - stretch/environment, driver and vehicle, which are connected by interfaces (see Figure 19).

The stretch module allows a detailed description of the influences of a stationary traffic environment. The course of the road in horizontal and vertical direction is represented by radius and transitions as well as the number and width of the lanes. The actual traffic conditions for a vehicle result from the number of surrounding vehicles and their distances and speeds. The vehicle module uses the cause-and-effect-principle to calculates the driving force starting from the engine operation point over the clutch, transmission and differential

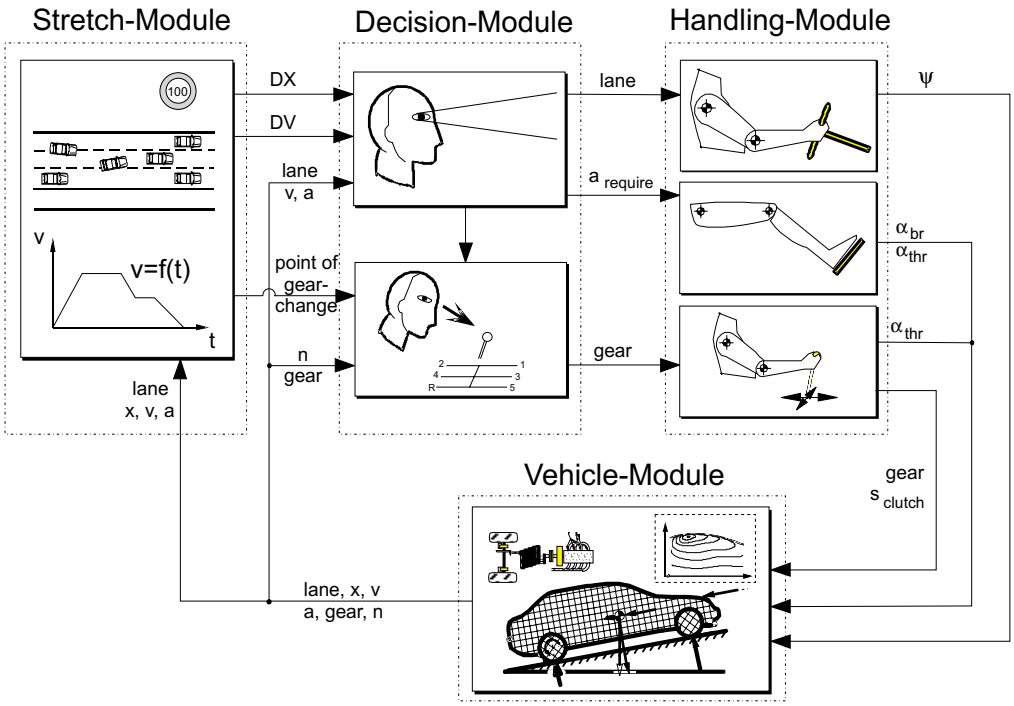


Fig. 19: Elements of the traffic simulation tool PELOPS.

up to the wheels, where the driving force is then balanced with the driving resistances. The operation point is changed by the alteration of the motor torque (cause). From the thereby caused acceleration and speed change results the engine speed (effect) under consideration of the drive-line elements. Only such a detailed description of the vehicle under employment of the cause-and-effect-principle allows the investigation of control engineering equipment, e.g., the adaptive cruise control system investigated in Section 5.3 (Breuer et al. 1999).

The link between vehicle and traffic simulation is represented by the driver module, which is divided into a behavior and a handling model. In the behavior model, the parameters of the local driving strategy is determined by means of the actual driving situation and the vehicle environment. The parameters of the local driving strategy are the desired acceleration, lane and eventually also the shifted gear. In the handling model, these parameters are converted into vehicle-specific quantities such as the angle of the acceleration pedal, brake, etc.

5.2 Design Space Exploration of Road Trains

The first example of a development task which was solved by the described procedure is the layout of the power train for road trains. Increasing the maximum payload is one possible approach to overcome the increasing traffic problems on crowded European highways. We focus on a concept for European freight traffic

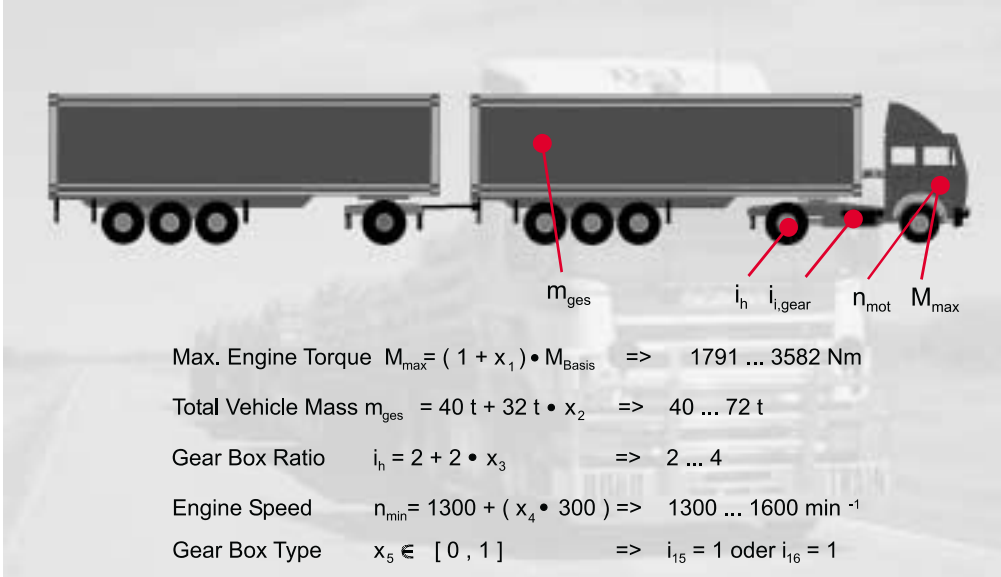


Fig. 20: Schematic view of the road train and its design variables

featuring two semi-trailers, which are connected by a one-axle dolly (Ludmann et al. 1999).

The optimization of a whole new vehicle concept with respect to fuel consumption and driving dynamics is a very complex subject, because the lack of existing data and knowledge leaves a wide open space for experiments concerning the power train and the overall weight of the road train. Furthermore, it is impossible to acquire knowledge in driving tests as prototypes are too expensive to be build. Therefore, the vehicle concept is modeled by a vehicle simulation. The design space exploration of the power-train is achieved by a parameter variation, performed by a multiobjective evolutionary algorithm.

5.2.1 Optimization Problem

A series of aspects have to be taken into consideration when developing a new vehicle concept. On the one hand an optimal combination of vehicle weight and engine power has to be found to ensure efficient driving. On the other hand the correct choice of gear box type and gear ratio influence driving comfort and performance. The design variables and their range are displayed in Figure 20. The first four variables are scale factors of different engine and gear box parameters. The last variable, x_5, represents the choice of the gear box type, specifically, whether the 15th or the 16th gear is chosen to be the direct gear. The direct gear is the most efficient gear, because the power is not transmitted through toothed wheels, but directly through the transmission shaft (gear ratio = 1). If the 15th gear is chosen as the direct gear, the 16th gear has a gear ratio below 1, which is less efficient. However, the low ratio causes higher engine rates and thus lower torque, which increases the gear box durability.

An increase of weight leads to an increase of road and climbing resistance.

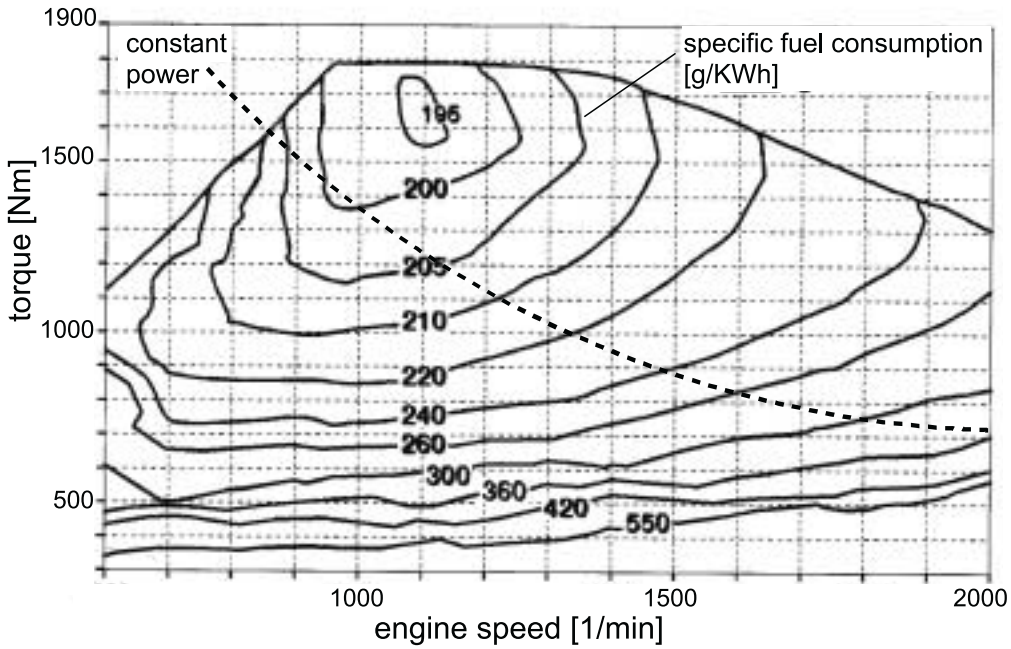


Fig. 21: Engine characteristic graph.

This changes the engine operating point and therefore its efficiency and fuel consumption. Every single driving condition defines a point in the engine characteristic graph (see Figure 21). The number of revolutions is determined by the velocity of the vehicle and the total gear ratio, consisting of rear-axle ratio and transmission ratio. The necessary torque is a result of necessary power output (influenced by velocity, efficiency of the gear box, acceleration, and road gradient) and revolutions. A lower total gear ratio reduces the engine speed. Under the presumption of constant running resistance due to constant velocity and road gradient the required power remains unchanged. The line of constant power indicates this relationship in Figure 21. Long-distance transport vehicles usually drive rather statically, operating at maximum authorized speed. This leads to the assumption that the gear ratio should be low enough to provide an engine operating point in the area of lowest specific fuel consumption. This area, however, is close to the line of maximum torque. Small increases of the running resistance, resulting from headwind or road gradient, cannot be compensated by requesting more torque from the engine, but force the driver to shift gears or to go at a lower speed. Since the drivability of the vehicle requires a large distance between the most frequent engine operating point and the line of maximum torque, resulting in powerful engines and high gear ratios, it opposes the attempt to reduce the fuel consumption. The goal of the optimization is to find a combination of overall weight, gear box, engine and driving strategy that minimizes fuel consumption, optimizes the driving performance and driving convenience.

Ten objective functions are defined to give a complete characterization of

the vehicle performance considering fuel consumption and driveability. The resulting multiobjective optimizaiton problem can be stated as follows:

Maximize $f(x) = (f_1(x), \ldots, f_{10}(x))$,

$$
\begin{aligned}
f_1(x) &= (-1) \cdot t_{a1}(x) && \text{[time for acceleration 0-40 km/h]} \\
f_2(x) &= (-1) \cdot t_{a2}(x) && \text{[time for acceleration 40-90 km/h]} \\
f_3(x) &= v_{max}(x) && \text{[maximal velocity]} \\
f_4(x) &= v_{14}(x) && \text{[maximal velocity, 1.5 gradient, 14th gear]} \\
f_5(x) &= v_{16}(x) && \text{[maximal velocity, 1.0 gradient, 16th gear]} \\
f_6(x) &= (-1) \cdot c_{100}(x) && \text{[average fuel consumption per ton load, 100 km/h]} \\
f_7(x) &= (-1) \cdot c_{80}(x) && \text{[average fuel consumption per ton load, 80 km/h]} \\
f_8(x) &= v_{ave}(x) && \text{[average speed on a highway, including road gradient]} \\
f_9(x) &= (-1) \cdot c_h(x) && \text{[average fuel consumption per ton load on a highway]} \\
f_{10}(x) &= (-1) \cdot g_{tot}(x) && \text{[number of gear shifts on a highway]}
\end{aligned}
$$

subject to $x = (x_1, \ldots, x_5) \in X = [0, 1]^5$.

The characteristic values $t_{a1}, t_{a2}, v_{max}, v_{14}, v_{16}, c_{100}, c_{80}, v_{ave}, c_h, g_{tot}$ are derived by simulation can therefore not be given in closed form. Six simulation scenarios are used, a full-load acceleration, two constant-velocity scenarios (80 km/h and 100 km/h), two scenarios with constant gradient and the engine operating at full load and a highway scenario. The highway scenario consists of an 18 km drive over an empty highway, with road gradient varying from -4.5% to +3.9%.

Another difficulty in the design process of a long-distance freight vehicle is the large application spectrum. Some carriers operate only in a rather even area, like the Netherlands for example. It is obvious that they would prefer a road train version different from one a carrier would choose whose standard route crosses the Alps. The latter puts much more emphasis on the climbing capacity than the other.

5.2.2 Algorithms

This application is an example of a design space exploration at a very early design stage. Here, simplicity of implementation is a main criterion to select a suitable optimization algorithm. Another aspect is that the archive must be very large because the possible size of the trade-off surface increases with the objective space dimension. In this case, its size does not have to be bounded at all because the long duration of the simulation (about 30 seconds) already limits the total number of alternatives to be generated in a reasonable amount of computing time. We therefore use SEMO (see Algorithm 5) and modify it slightly for our needs.

Each individual represents a decision alternative by a vector of design variables $x = (x_1, x_2, \ldots, x_{10}) \in X$. The variation operator for this study only applies mutation. For each component of the decision vector, a random number is drawn from a standard normal distribution and multiplied with a scaling

factor σ. This product is then added to the old component x_i to form the new component x_i':

$$x_i' := x_i + \sigma \cdot r_i, \quad r_i \sim N(0, 1) \tag{5.1}$$

A constant step size $\sigma = 0.02$ was used for simplicity. Recombination turned out not to be of use here since the interdependence of the design variables in every part of the objective space seems to be very high.

In order to avoid genetic drift and an oversampling of easily accessible objective space regions, it is necessary to employ density dependent selection: In each iteration the density is estimated for every point represented by the individuals, and the individuals are selected with a probability reciprocal to this density. This leads to a more uniform distribution of alternatives in the approximated trade-off surface.

To analyze the quality of the vehicles developed by the evolutionary algorithm, a road train version is designed in a traditional way, based on simple rules for optimizing a power train of a truck (Wallentowitz 2000). In addition, two grid searches over the whole design space are performed, each with a total number of 2160 elements. One of them was restricted to a maximum authorized speed of 80 km/h, the other to 100 km/h.

5.2.3 Results

A hierarchical approach was used for the design space exploration with the evolutionary algorithm. The first run of the evolutionary algorithm is performed to narrow down the design variable intervals. An analysis of the trade-offs between the different objectives leads to the conclusion that a focus on reducing the fuel consumption would not necessarily worsen the other objective values to an unacceptable amount. Furthermore, this goal is the main factor for the profitability of a vehicle concept and deserves special attention. Therefore we chose the average fuel consumption on the highway, f_9, as the objective value that defines a ranking of the solutions; f_4 and f_5 can be used to represent the second main part of the driving performance, the required climbing ability. In this case the reduction of the maximum velocity must not exceed 5 km/h. The solutions that did not meet this criterion were removed from the ranking. The remaining individuals were ranked according to the fuel consumption on highways. The top solution was considered as the best version.

With the help of the preliminary solutions shown in Figure 22, the design variable intervals were narrowed down to

$x_1 \in [0.4, 0.6]$	$x_1 \in [0.0, 0, 4]$
$x_2 = 1$	$x_2 = 1$
$x_3 \in [0.3, 0.4]$	$x_3 \in [0.55, 0.85]$
$x_4 \in [0, 0.5]$	$x_4 \in [0, 0.5]$
$x_5 = 0$	$x_5 = 0$
(100 km/h road train)	(80 km/h road train).

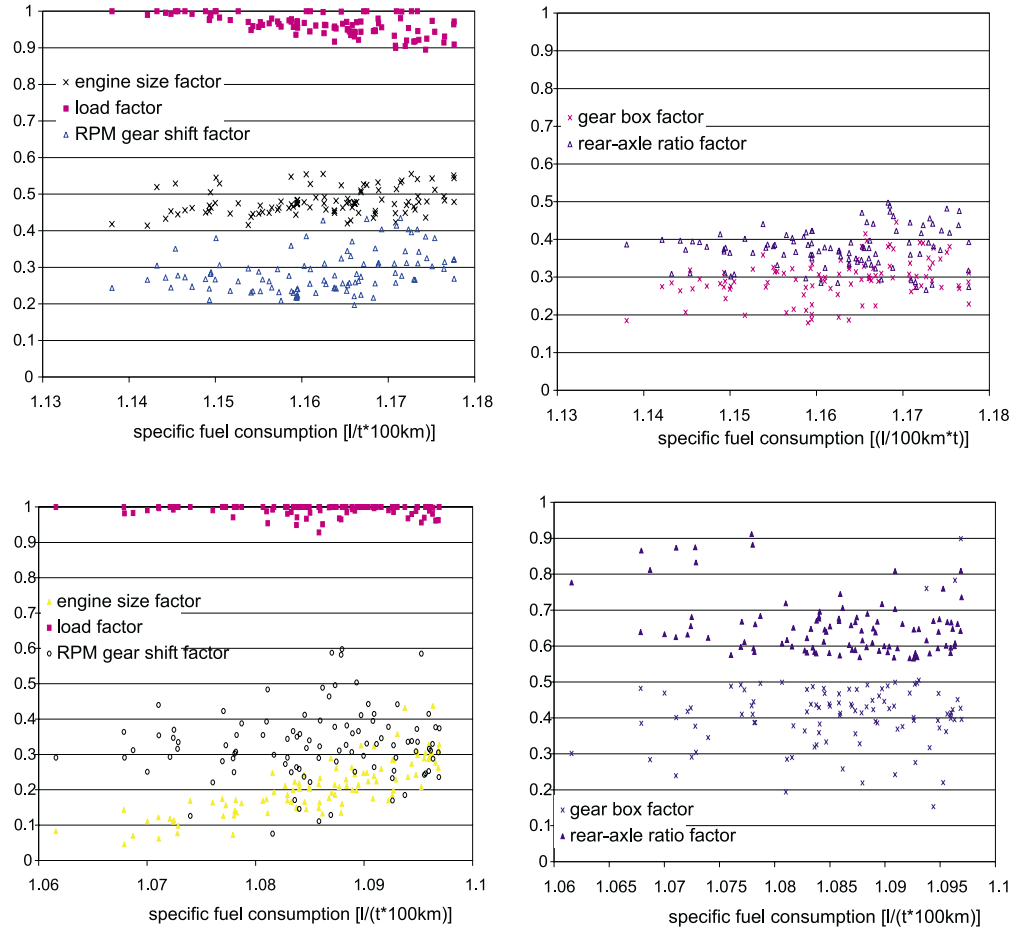
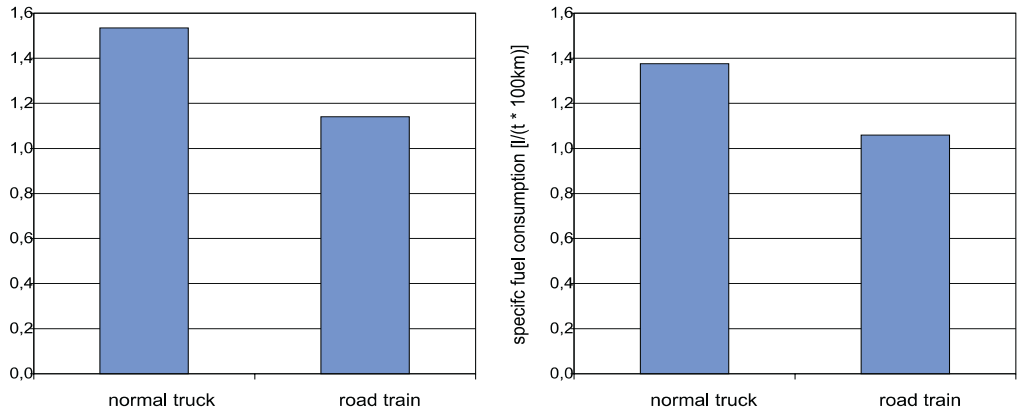


Fig. 22: Design variable and specific fuel consumption for the 100 km/h road train (top) and the 80 km/h road train (bottom)

Limited to those intervals, a second run of the same evolutionary algorithm then fulfilled a more exact approximation of the Pareto set in the region of interest. Of course, there are other ways to cope with the large number of incomparable alternatives in the presence of many objectives. These typically rely on preference information, for instance aggregating (or dropping) objectives, lexicographic ordering or the transformation of objectives into constraints. In many cases, however, it is very difficult to derive an exact numerical representation of the preferences. Moreover, since we had different decision makers with different preferences in mind, the aim is first to explore the Pareto set as broadly as possible with a minimum number of simulations before exploiting interesting regions through restriction of the decision variable space as described above. Finally, it should be mentioned that even dropping highly correlated objectives does not help since these correlations are usually not known in advance, can differ much in different regions of the search space, and they do not contribute to the dimensionality of the Pareto set.



$x_1 = 0.507$
$x_2 = 1$
$x_3 = 0.183$
$x_4 = 0.256$
$x_5 = 0$

$x_1 = 0.157$
$x_2 = 1$
$x_3 = 0.623$
$x_4 = 0.374$
$x_5 = 0$

Fig. 23: Comparison between a normal truck and a road train concerning fuel consumption on a highway, maximum authorized speed of 100 km/h (left) and 80 km/h (right). In the boxes, the decision variables corresponding to the chosen road train are listed.

Final results show a huge advantage of road trains with respect to fuel consumption in comparison to normal truck (see Figure 23). A decrease of 23% (80 km/h-version) respectively 26% (100 km/h-version) is achieved on highways in spite of the rather tough gradients. In steady-state operation fuel consumption advantages of up to 35% are accomplished. With acceleration being at a sensible level the road trains have no disadvantages in climbing ability and required gear shifts.

The comparison of the different road train versions indicates that the evolutionary algorithm is able to generate better solutions than the other approaches. Showing the same climbing ability and acceleration as the traditionally developed versions and the ones obtained by a grid-scan over the whole parameter area, the EA-solution needs about 1% less fuel on the highway. The 100 km/h version is even better than the best version found by a grid-scan of 1000 elements distributed over the narrowed intervals.

Figure 25 shows the relation between the objective function f_4 (maximal velocity in 14th gear with 1.5% road gradient) and f_9 (specific fuel consumption on highway). This relation provides information about the trade-off between drivability and fuel economy. The creation of 1300 individuals already produces a rather large number of solutions, which have to be considered better than any solution that was found without the evolutionary algorithm. This advantage in efficiency becomes even more important when more sophisticated driving scenarios – and thus more time consuming simulations – are used, which is subject to further research.

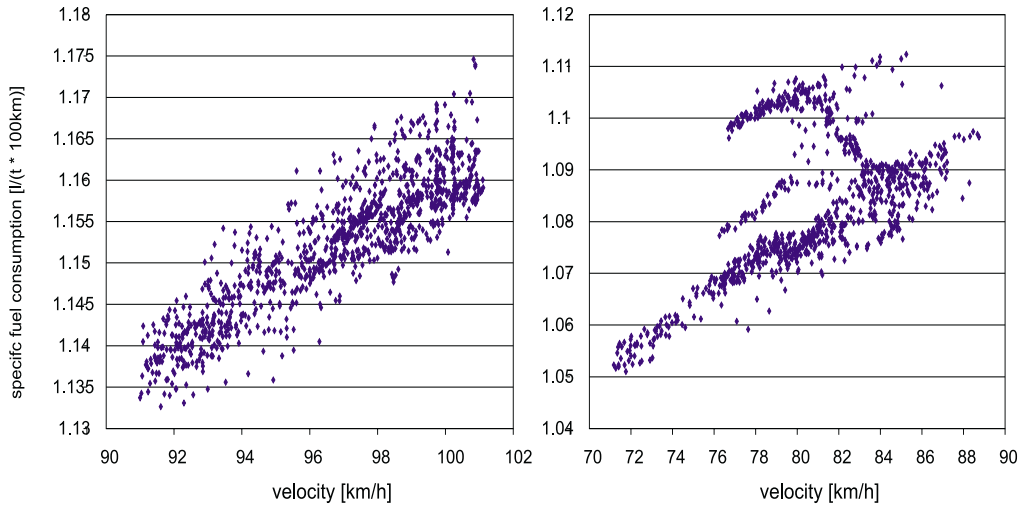


Fig. 24: Trade-off between velocity (1.5% road gradient) and specific fuel consumption on a highway for the 100 km/h road train (left) and the 80 km/h road train (right)

5.3 Parameter Optimization of Adaptive Cruise Control Systems

Crowded motorways and a higher average vehicle speed create increasing difficulties for drivers. The automobile industry tries to compensate these additional demands by inventing driver assistance systems such as antilock braking system (ABS), cruise control (CC) and electronic stability control (ESC). In contrast to the systems mentioned above, adaptive cruise control (ACC) has not been thoroughly established yet.

The ACC-system is an enhanced cruise control, not only designed to keep the vehicle's speed constant, but also to analyze the traffic situation in front of the vehicle and regulate its longitudinal dynamics accordingly. Thus, it especially suits the demands of truck drivers, who frequently have to follow a leading vehicle. Used effectively, ACC-systems can increase driving safety, make driving more comfortable and reduce fuel consumption. However, it is rather difficult to develop a controller that meets the drivers' requirements concerning its speed regulating behavior as well as safety criteria and fuel efficiency.

Since experimental testing of each modified controller variant would enormously raise development costs and time, the ACC-system's behavior is evaluated and analyzed by simulation. This offers the possibility to improve the development process further by applying numerical optimization techniques such as evolutionary algorithms to optimize the ACC-controller.

5.3.1 Optimization Problem

The longitudinal controller is responsible for the translation of the incoming data about the traffic situation in front of the vehicle and its own driving con-

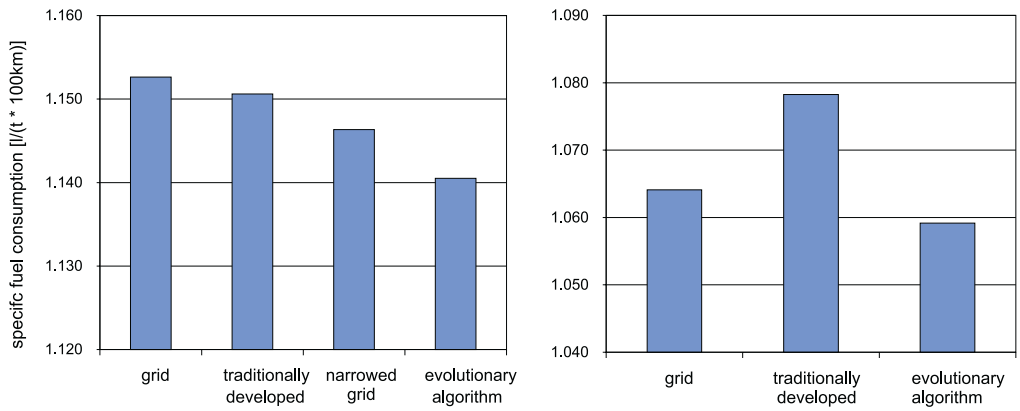


Fig. 25: Specific fuel consumption on a highway for the 100 km/h road train (left) and the 80 km/h road train (right).

dition into a desired acceleration. The data produced by the sensor contains some deviation. This requires four different filters in order to create a smooth acceleration signal. The influence of these filters can be regulated by four integer parameters, here represented by the design variables $x_1, \ldots, x_4$. Strong filters result in very smooth signals. However, they somewhat delay the vehicle's reaction to incoming data and that weakens its driving performance. Two further design variables, x_5, x_6 are used to define the longitudinal controller's reaction to the vehicle's distance from the leading vehicle and their relative velocity. Each setting of the design variables represents a decision alternative, and the resulting controller performance is determined through simulation.

Four objectives are defined to give a sufficient characterization of the ACC-system's longitudinal controlling behavior considering driving comfort, fuel efficiency and safety: All these objective functions are computed within the simulation. The resulting multiobjective optimization problem can be stated as follows (where the function values of f and g are calculated by the simulator):

Maximize $f(x) = (f_1(x), \ldots, f_4(x))$,

$$\begin{aligned} f_1(x) &= (-1) \cdot c_{ave}(x) && \text{[average fuel consumption]} \\ f_2(x) &= (-1) \cdot t_{acc}(x) && \text{[time for acceleration]} \\ f_3(x) &= (-1) \cdot d_{vel}(x) && \text{[velocity deviation]} \\ f_4(x) &= (-1) \cdot d_{acc}(x) && \text{[acceleration deviation]} \end{aligned}$$

subject to $x = (x_1, \ldots, x_6) \in X = \{1, 2, \ldots, 99\}^2 \times \{1, 2, \ldots, 16\} \times \{1, 2, \ldots, 8\}^3, g(x) \geq d_{min}$ [minimum follow-up distance]

5.3.2 Algorithms

In order to approximate the Pareto set for the constrained multiobjective integer programming problem above, three methods are applied and compared: a grid

search and two evolutionary algorithms. The computation time of the simulator makes exhaustive search or complete enumeration of all alternatives impractical. Thus, a grid search with 2^{15} representative solutions regularly distributed in the decision variable space is performed. In comparing all these alternatives to each other, the dominated ones are eliminated and the remaining represent a first approximation to the non-dominated set as a baseline for comparison.

For the evolutionary algorithms, SPEA2 (Zitzler et al. 2002), an improved version of the Strength Pareto Evolutionary Algorithm (Zitzler and Thiele 1999), is applied. The select operator of SPEA2 works with a fixed-size population A; it is basically a $(\mu + \lambda)$-strategy. Its fitness assignment scheme of SPEA2 has been analyzed in Chapter 2.3.2 and was found to possess several desirable properties. Based in this fitness assignment, selection is performed in two steps: environmental selection and mating selection. The best μ individuals out of the old parent population $A^{(t-1)}$ and the λ new offspring $B^{(t-1)}$ survive and form $A^{(t)}$ according to the following scheme. First, all non-dominated individuals, i.e., those with fitness value $\Phi(a) = 0$ are selected. If there are more than μ such solutions, a truncation procedure is invoked which iteratively removes the individual which is closest to the others. If less than μ individuals are non-dominated, the space is filled with the dominated individuals in ascending order of their fitness values. In the mating selection step, the parent population of size μ is created by binary tournament selection (with replacement) based on the fitness values.

The generate operator uses two different variation schemes that give rise to two different instances of the algorithm, r-SPEA2 (using real-valued individuals) and i-SPEA2 (using integer-valued individuals):

Real-valued individuals Many standard search operators are based on a floating-point representation of (real-valued) decision variables. Therefore a continuous relaxation of the search space to $[0, 99]^2 \times [0, 16] \times [0, 8]^3$ is used, and the variables are rounded to their integer part (plus 1) before each run of the simulation tool. For the recombination, we use the SBX-operator (Deb 2001) with distribution index $\eta = 5$. The offspring individuals are then mutated by adding normal distributed random numbers according to Equation 5.1, where the standard deviation σ is set to 5 per cent of the interval length.

Integer-valued individuals As the relaxation produces an artificial blow-up of the search space a direct representation of the decision variables as integer numbers might be more appropriate. It also eliminates the potential problem of mapping several different individuals to the same decision alternative by the rounding procedure. Search operators working directly on integer variables are not so common in evolutionary computation. We adopt the techniques from Rudolph (1994), who developed an EA for integer programming with maximum entropy mutation distributions that enables self-adaptive mutation control similar to real-valued evolution strategies. A successful application to a mixed integer design problem for chemical

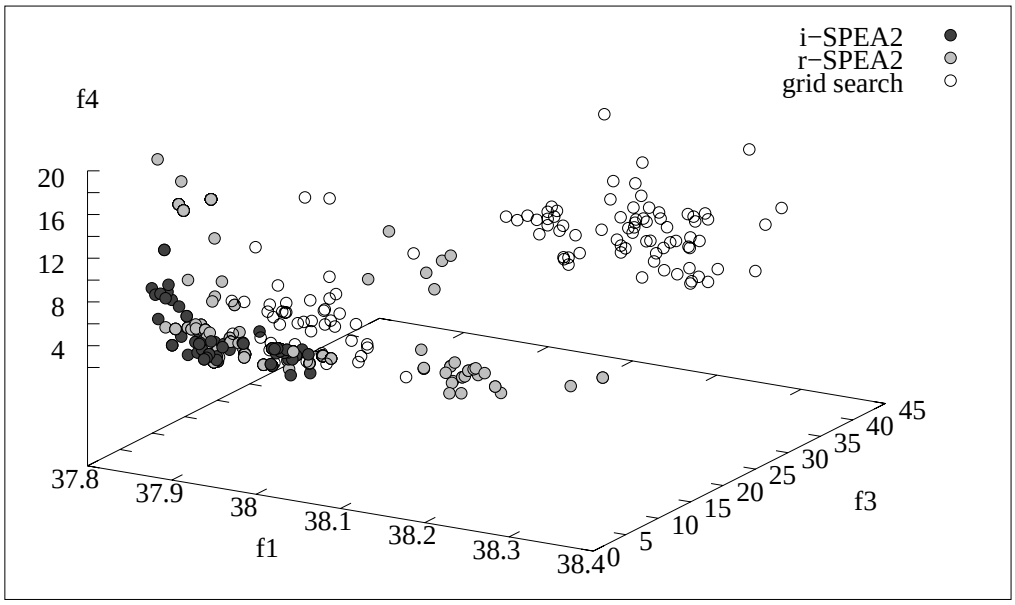


Fig. 26: Scatter plot of the non-dominated solutions produced by the grid search and the evolutionary algorithm with continuous relaxation (r-SPEA2) and direct integer coding (i-SPEA) for the objective function values f_1, f_3, f_4

plants is reported in Emmerich et al. (2000). Here, the initial mutation step size was set to $s = 2$ for all variables.

Both version of SPEA2 were terminated after 3000 objective function evaluations and we chose $\mu = \lambda = 20$. During the run, a separate archive of all non-dominated solutions was maintained and output as the approximation to the non-dominated set at the end of the run.

5.3.3 Results

To evaluate the performance of the evolutionary algorithm, a grid search over the whole parameter area is performed, along with a manual optimization of the ACC-controller. The grid search contains 16384 elements, requiring a computation time of almost 137 hours[1]. As both instances of the evolutionary algorithm only used 3000 function evaluations each, and since their internal operations and data processing can be neglected compared to the simulation, they have a clear advantage in terms of computation time.

As a first interesting observation from the output of the different algorithms, no trade-off is visible for the second objective f_2 (acceleration / deceleration time). All algorithms have found the optimal value of 66.6 for almost all non-dominated alternatives. This is the optimal value attainable by immediate full acceleration, without any delays caused by the ACC system. Hence, it cannot be improved further. The remaining objective values of the different non-

[1]This estimate is based on the average running time of the simulation on a PC with an AMD ATHLON 1800 processor

dominated sets are displayed in Figure 26. The trade-off characteristic is visible from the three-dimensional scatter plot.

One goal of this case study is to conduct a systematic performance assessment and comparison of the different techniques to exemplify the theoretic results obtained in Chapter 2.4. We start with the *hypervolume* indicator (Zitzler and Thiele 1998) as an example of an absolute quality indicator with strong inferential power. The hypervolume indicator calculates the normalized volume of the dominated space to evaluate a single non-dominated set alone. As it requires a bounded objective space, a reference cuboid is defined between the ideal point f^* and the nadir point given by the maximal objective function values of the maximal elements of the output of all three algorithms. $I_{\mathrm{H}}(A)$ gives the fraction of this reference volume that is dominated by A. It is clear that the algorithms should minimize the dominated space. The results given in the last column of Table 5 show

$$I_{\mathrm{H}}(A_{\mathrm{i-SPEA2}}) > I_{\mathrm{H}}(A_{\mathrm{r-SPEA2}}) > I_{\mathrm{H}}(A_{\mathrm{gridsearch}})$$

which allows to conclude

$$A_{\mathrm{i-SPEA2}} \not\preceq A_{\mathrm{r-SPEA2}} \not\preceq A_{\mathrm{gridsearch}}.$$

These statements are quite weak, and we have to apply relative quality indicators to arrive at stronger statements. We consider two relative quality indicators proposed by Zitzler and Thiele (1999), the *coverage* indicator I_{C} and the *binary hypervolume* indicator I_{H2}. Both indicators are among those with strongest inferential power (see Table 3).

$I_{\mathrm{H2}}(A, B)$	i-SPEA2	r-SPEA2	grid search	$I_{\mathrm{H}}(A)$
i-SPEA2		0.0038	0.223	0.949
r-SPEA2	0.002		0.188	0.913
grid search	0.0003	0.0018		0.726

Tab. 5: Results of the binary hypervolume indicator $I_{\mathrm{H2}}(\cdot, \cdot)$ applied to all pairs of algorithms and the absolute hypervolume indicator $I_{\mathrm{H}}(\cdot)$ (last column).

The coverage indicator provides information about how much of one algorithm's output has also been reached by the other algorithm. Specifically, $I_{\mathrm{C}}(A, B)$ calculates the relative number of points of set B that are dominated by at least one point in set A. Table 6 shows the results. It can be seen that none of the points found by the grid search is better than any point in the non-dominated sets of the evolutionary algorithms. Also, the SPEA2 working with the floating point representation does not cover much (less than 10%) of the solutions produced by the integer version, which in turn is able to dominate nearly half of the solutions of its competitor. However, as far as the preference relations on approximation sets are concerned, the relations listed in Table 3 lead to the conclusion that the output of all algorithms is mutually incomparable, because

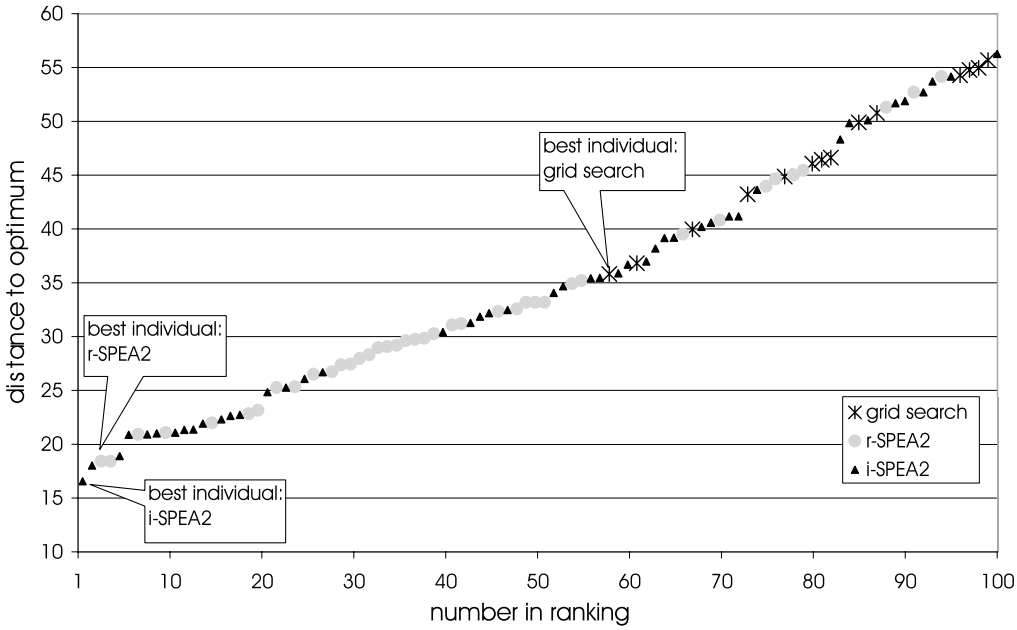


Fig. 27: Ranking of solutions according to the scalar utility function (5.2).

all values of the coverage indicator are strictly smaller than one. The same conclusion can of course be drawn from the binary hypervolume indicators, whose results are also listed in Table 5. The binary hypervolume $I_{\mathrm{H2}}(A, B)$ evaluates to the volume dominated by set A, but not dominated by set B.

$I_{\mathrm{C}}(A, B)$	i-SPEA2	r-SPEA2	grid search
i-SPEA2		0.423567	0.991597
r-SPEA2	0.070588		0.991597
grid search	0	0	

Tab. 6: Results of the coverage quality indicator $I_{\mathrm{C}}(\cdot, \cdot)$ applied to the output of all pairs of algorithms.

This situation of mutually incomparable approximation sets is very typical for a comparative study, because the performance differences are seldom so strong that one set entirely dominates another. Nevertheless, the indicator values provide insight, in which aspects the output of the algorithms differs. In our case, for instance, the order of the indicator values is always the same, showing that both evolutionary algorithms at least largely dominate the output of the grid search, with a slight advantage of the integer-valued version. Such conclusions can of course be drawn, however, one has to be very careful with the interpretation of the results, especially to avoid general statements such as “algorithm A is better than algorithm B” when this is formally incorrect.

We conclude this performance assessment by investigating further performance *aspects* in more detail. These aspects correspond to different preferences of the decision maker, i.e., which regions of the objective space he is mostly in-

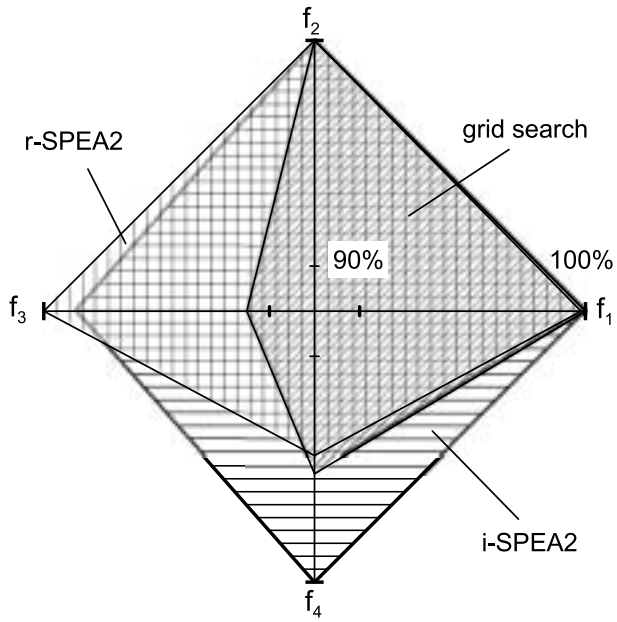


Fig. 28: This graph shows how well each algorithm could approach the minimal value in each objective dimension separately. The values on each axis is calculated by dividing the overall minimal value by the minimal value produced by each algorithm.

terested in. Such an assessment is of course a *subjective* one. All the following indicators can be seen as special cases of the distance-indicator I_{D} (see Table 2).

One possibility is to define a utility function based on a weighted distance to an ideal point f^*, which is given by the minimal objective values in each dimension. The difference between each objective value and the optimum value in the corresponding category is multiplied with a factor which represents the importance of the category. Thus, the interpretation of the results reflects an adaptation to the decision maker's preferences. In this case, the objectives f_1 and f_3 are considered most important, f_2 is least important. Representing the distance to the optimal solution, the sum of those values gives the overall quality of the individual

$$D(x) = 150(f_1(x) - f_1^*) + (f_2(x) - f_2^*) + 6(f_3(x) - f_3^*) + 4(f_4(x) - f_4^*) \quad (5.2)$$

with $f^* = (37.8339, 66.6, 2.06935, 3.03196)$. Accordingly, a ranking of the individuals developed by the different optimization strategies can be produced. The best 100 solutions are displayed in Figure 27. The two evolutionary algorithms create the best solutions, with a slight advantage of the integer-version in terms of density close to the optimum solution. 46 out of the top 100 solutions were created by this integer-version, 40 by the double-version and only 14 by the grid-search.

Another aspect that can be taken into consideration in order to compare the different optimization approaches is the total objective space that their solutions extend to. The minimum overall objective function value, divided by the minimum value of a single approach, determines the quality of the optimization

regarding the corresponding objective. Figure 28 visualizes the performance of the different optimization approaches in terms of objective space exploration. While all three approaches reach the optimum in the objectives f_1 and f_2, the grid search shows a performance almost 10% below the optimum in f_3 and 5% in f_4. The integer-version of the EA proves to be the best optimization method with a good performance in all four objectives and an average value of 99.62%.

5.4 Model Fitting for a Vehicle Dynamics Simulation

A crucial element of the application process of simulation and optimization techniques in vehicle development is model fitting. Here, a MATLAB simulation is to be fitted to data acquired in real driving tests.

The two most important elements of modeling are simplification and exactness. In this case, an extended bicycle model is used for vehicle driving dynamics research. The original bicycle model is a rather simple representation of real cars, because the four tire contact points are centralized in the longitudinal axle of the car. Time-delays for the lateral tire force generation enable a sufficiently exact reproduction of real vehicle measurements. However, it is necessary to fit a number of vehicle parameters in order to achieve a satisfying performance of the vehicle model.

5.4.1 Optimization Problem

The two transfer functions yaw rate and lateral acceleration with the input steering angle are the most important criteria when analyzing the quality of lateral vehicle dynamics simulation. Both phase lag and gain have to represent the original vehicle behavior with maximum precision.

The first main task was to develop a model structure that enables sufficient exactness while remaining as simple as possible. To achieve that, the original bicycle model was extended. Tire phase lags create a delayed reaction to steering angle changes. In addition to that, the vehicle's rolling behavior was modeled in a simple way in order to represent the body movement in relation to the tires. The following 8 real-valued design variables enable a sufficient adjustment to different vehicles:

$$
\begin{aligned}
J_z &: \text{yaw inertia} \\
c_{\alpha,f} &: \text{tire stiffness, front} \\
c_{\alpha,h} &: \text{tire stiffness, rear} \\
t_f &: \text{phase lag front tire} \\
t_r &: \text{phase lag rear tire} \\
c_w &: \text{roll stiffness} \\
d_w &: \text{roll damping}
\end{aligned}
$$

$$J_x \quad : \quad \text{roll inertia}$$

These parameters mainly represent tire characteristics and vehicle mass distribution. The data for other parameters like vehicle mass and length can simply be measured. Therefore it is not necessary to include those values in the design parameter set.

Typical manoeuvres for vehicle parameter identification are steering angle sweeps with a constant lateral acceleration of about $4\text{m}/\text{s}^2$. Both transfer functions mentioned above can be derived from those manoeuvres. The comparison of simulation and driving tests results in the following four-objective optimization problem:

Maximize $f(x) = (f_1(x), \ldots, f_4(x))$,

$$\begin{array}{rcll} f_1(x) & = & (-1) \cdot a_a(x) & \text{[gain deviation, lateral acceleration]} \\ f_2(x) & = & (-1) \cdot t_a(x) & \text{[phase lag, lateral acceleration]} \\ f_3(x) & = & (-1) \cdot a_y(x) & \text{[gain deviation, yaw rate]} \\ f_4(x) & = & (-1) \cdot t_y(x) & \text{[phase lag, yaw rate]} \end{array}$$

subject to $x = (x_1, \ldots, x_8) \in X = [0, 1]^8$.

5.4.2 Algorithms

Given is a multiobjective optimization problem with eight real-valued, normalized decision variables and and objective function with four components. As we are not primarily focused on a comparison of different algorithms, we start directly with the SPEA2 described in the previous section.

The generate operators used for this problem again apply recombination and mutation. A simple discrete recombination was chosen which creates one offspring solution x' from two parents $x^{(a)}$ and $x^{(b)}$. For each decision variable x_i, $i \in \{1, \ldots, 8\}$, one parent was determined randomly and its decision variable copied to the child. The resulting child is then mutated using normal-distributed random variables, again according to Equation 5.1. The mutation step sizes σ were chosen in each iteration adaptively: they are determined by the absolute difference of the parent variables, divided by 2, i.e.,

$$\sigma_i := \frac{1}{2}|x_i^{(a)} - x_i^{(b)}|.$$

The first run of this SPEA2 version soon reached a situation, where the population stagnated and no further progress was visible. Instead, the population was oscillating around a certain area in objective space. The reason for this was the problem of deterioration, which was discussed in Chapter 3.1.

To overcome this convergence problem, the selection operator of SPEA2 had to be replaced with the select operator maintaining an ε-Pareto set (see Algorithm 3). Such convergence problems, which have been verified for many

multiobjective EAs, indicate that the algorithm is operating close to the Pareto set. To achieve further progress, special care has to be taken regarding the selection and deletion of solutions from the archive population. Using Algorithm 3, instead, guarantees that the set of archived solutions never deteriorates and thus monotonously converges to the Pareto set.

5.4.3 Results

The evolutionary algorithm was able to find solutions of a sufficient quality rather quickly. Regarding the final approach to the Pareto set, the selection algorithm maintaining an ε-Pareto set showed a considerably better performance than SPEA2. In the first part of the optimization process, the user can derive interesting information about the model behavior by analyzing the trade-offs between the objective functions, since the archive is still rather widely spread as visualized in Figure 29.

In the course of the optimization, the focus of the decision maker shifts from diversity to examining the area of the best solutions in detail. To achieve this goal, we made use of two extensions of the algorithm described in Section 3.4, ranges of non-acceptance and a dynamic adaptation of the ε values. The adaptation, however, was not automated, but user driven. Both measures represent examples of exploiting the second feedback loop in our design framework, the interaction of the decision-maker with the optimization algorithm. The development of the solutions, especially the focusing on interesting regions, is also visible in Figure 29 for different stages. The upper diagram shows that in the third stage, more emphasis was put on improving the phase lag deviation of the yaw rate, so all solutions above a certain threshold were prohibited. This measure subsequently led to a considerable better approximation of the lower part of the trade-off surface, keeping the gain deviation of the lateral acceleration still in the interval between [0.002, 0.01].

The discovered solutions create a simulation environment, in which lateral vehicle dynamics can be simulated with a rather simple model and a sufficient exactness. The exactness is demonstrated in Figure 30, where the simulated time series are plotted against the data of the real vehicle obtained in driving tests.

The simplicity of the model results in a low computation time, which enables an online use for controlling the vehicle dynamics not only on a general purpose processor, but also in a vehicle-specific architecture, where it is impossible to run a full scale vehicle simulation. This integration is subject of ongoing research.

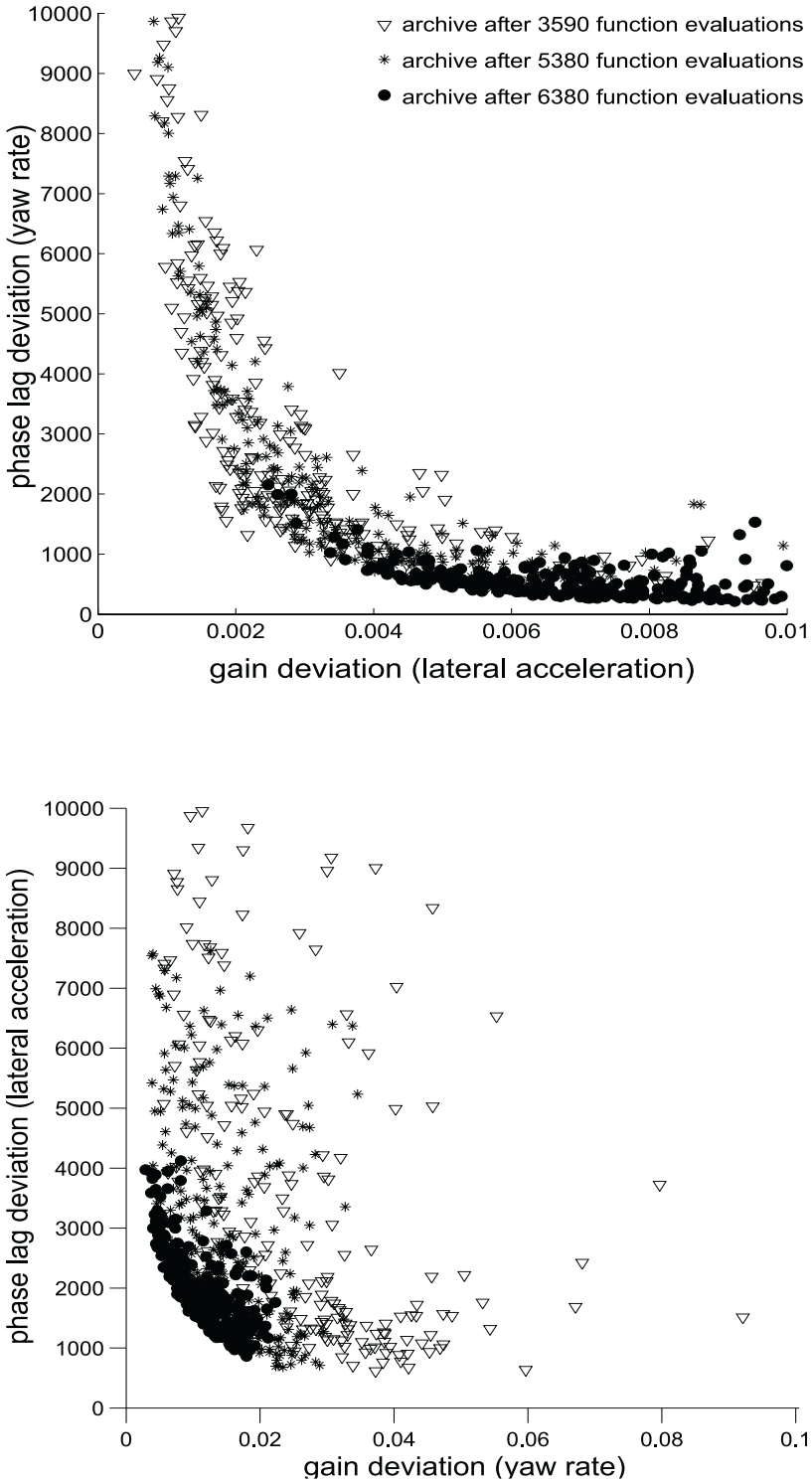


Fig. 29: Selected trade-offs between the four objective functions.

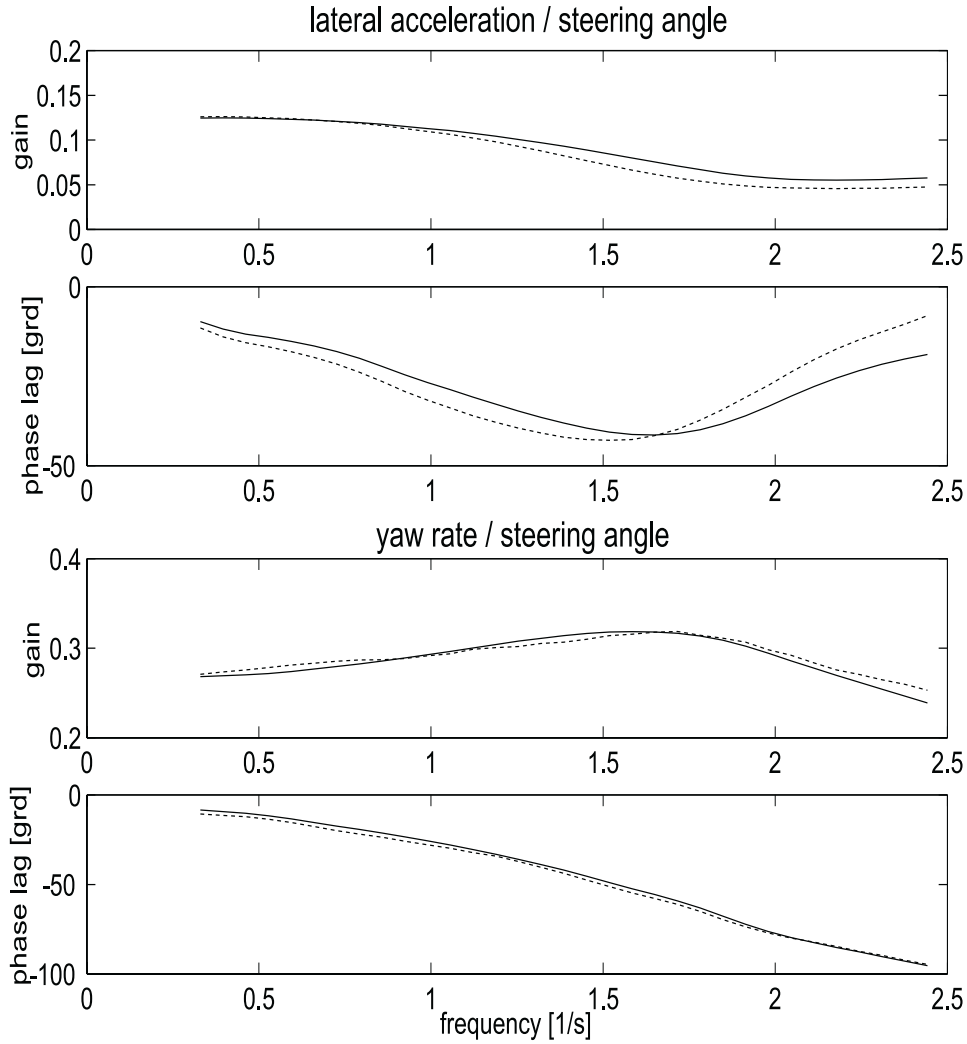


Fig. 30: Comparison of simulated and real vehicle behavior. The solid line is an interpolation of the measured data from the real vehicle, the dashed line represents the simulation data obtained from the vehicle model.

5.5 Summary

Evolutionary algorithms proved to be a powerful optimization tool in three different applications. The advantage of black-box optimization of complex systems, along with the approximation of the Pareto optimal set of solution provides the development engineer with detailed information about trade-offs between the different objective functions and helps to understand the problem at hand.

The road train example showed that even a simple approach is suitable for a design space exploration task at an early design stage. In addition to a detailed overview of the tradeoffs between the ten objectives, the evolutionary algorithm was able to present a solution that dominates the one found by the engineer on a trial-and-error base.

The problem related to the design of an adaptive cruise control system was a parameter optimization of filters used in controller. Here, the effectiveness of different variation operators was to be compared. The comparative study was carried out based on the previous results about quality indicators and performance assessment.

The last case study was a model fitting problem. Preliminary trials with a standard multiobjective EA revealed convergence problems. Therefore, the new selection operator to maintain an ε-Pareto set had to be applied. Through a manual adjustment of the ε values, the approximation quality was steadily increased in the areas of interest of the designer.

Bibliography

Alon, N. (2002). A random process for searching a graph (comment). Personal communication.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press.

Bäck, T., D. B. Fogel, and Z. Michalewicz (Eds.) (1997). *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing and Oxford Univertity Press.

Bell, D. E., R. L. Keeney, and H. Raiffa (1977). *Conflicting objectives in decision. International Series on Applied Systems Analysis 1*. Chichester: Wiley.

Beyer, H.-G., H.-P. Schwefel, and I. Wegener (2002). How to analyse evolutionary algorithms. *Theoretical Computer Science 287*, 101 – 130.

Blickle, T. and L. Thiele (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation 4*(4), 361–394.

Breuer, K., M. Weilkes, and J. Ludmann (1999). Development and In-Vehicle Application of an ACC-controller. In *32nd International Symposium On Automotive Technology And Automation (ISATA'99)*, Vienna, Austria.

Bridges, D. S. and G. B. Mehta (1995). *Representations of Preference Orderings*. Berlin: Springer.

Chankong, V. and Y. Haimes (1983). *Multiobjective Decision Making Theory and Methodology*. Elsevier.

Coello Coello, C. A., D. A. Van Veldhuizen, and G. B. Lamont (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer.

Czyzak, P. and A. Jaskiewicz (1998). Pareto simulated annealing—a metaheuristic for multiobjective combinatorial optimization. *Multi-Criteria Decision Analysis 7*, 34–47.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley.

Deb, K. and R. B. Agrawal (1995). Simulated binary crossover for continuous search space. *Complex Systems 9*, 115–148.

Deb, K., S. Agrawal, A. Pratap, and T. Meyarivan (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer et al. (Eds.), *Parallel Problem Solving from Nature (PPSN VI)*, Lecture Notes in Computer Science Vol. 1917, pp. 849–858. Springer.

Deb, K., M. Mohan, and S. Mishra (2003). Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In C. M. Fonseca et al. (Eds.), *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science Vol. 2632, pp. 222–236. Springer.

Droste, S., T. Jansen, and I. Wegener (1998). A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs. *Evolutionary Computation 6*(2), 185–196.

Droste, S., T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science 276*(1–2), 51–81.

Ehrgott, M. (2000). *Multicriteria optimization*. Berlin: Springer.

Ehrgott, M. and X. Gandibleux (2000). A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. *OR Spektrum 22*, 425–460.

Emmerich, M., M. Grötzner, B. Gross, and M. Schütz (2000). Mixed-integer evolution strategy for chemical plant optimization with simulators. In I. C. Parmee (Ed.), *Evolutionary Design and Manufacutre — Selected papers from ACDM'00*, pp. 55–67. Springer.

Erlebach, T., H. Kellerer, and U. Pferschy (2001). Approximating multiobjective knapsack problems. In *Proceedings of the Seventh International Workshop on Algorithms and Data Structures (WADS 2001)*, Lecture Notes in Computer Science 2125, pp. 210–221. Springer.

Evtushenko, Y. G. and M. A. Potapov (1987). Methods of numerical solution of multicriterion problem. *Soviet mathematics – doklady 34*, 420–423.

Fandel, G. and J. Spronk (1985). *Multiple Criteria Decision Methods and Applications*. Berlin: Springer.

Fleischer, M. (2002). The Measure of Pareto Optima: Applications to Multiobjective Metaheuristics. Technical Report 2002-32, Institute for Systems Research, University of Maryland.

Fleischer, M. (2003). The Measure of Pareto Optima. In C. M. Fonseca et al. (Eds.), *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science Vol. 2632, pp. 519–533. Springer.

Fonseca, C. M. and P. J. Fleming (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation 3*(1), 1–16.

Fonseca, C. M. and P. J. Fleming (1997). Multiobjective optimization. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, Bristol, UK, pp. C4.5:1–9. IOP Publishing and Oxford University Press.

Garnier, J. and L. Kallel (2000). Statistical distribution of the convergence time of evolutionary algorithms for long-path problems. *IEEE Transactions on Evolutionary Computation 4*(1), 16 – 30.

Garnier, J., L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

Giel, O. (2003a). Expected runtimes of a simple multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation (CEC 2003)*, Piscataway, NJ. IEEE Press.

Giel, O. (2003b). Runtime analyses for a simple multi-objective evolutionary algorithm. Technical Report CI-155/03, SFB 531, Universität Dortmund, Germany.

Glover, F. and M. Laguna (1997). *Tabu Search*. Boston: Kluwer Academic Publishers.

Haimes, Y., L. Lasdon, and D. Wismer (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics 1*, 296 – 297.

Hancock, P. (1997). A comparison of selection mechanisms. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, pp. C2.8:1–11. Bristol, UK: IOP Publishing and Oxford University Press.

Hanne, T. (1999). On the convergence of multiobjective evolutionary algorithms. *European Journal Of Operational Research 117*(3), 553–564.

Hanne, T. (2001). Global multiobjective optimization with evolutionary algorithms: Selection mechanisms and mutation control. In E. Zitzler et al. (Eds.), *Evolutionary Multi-Criterion Optimization (EMO 2001)*, Lecture Notes in Computer Science Vol. 1993, Berlin, pp. 197–212. Springer.

Hansen, M. P. and A. Jaszkiewicz (1998). Evaluating the quality of approximations of the non-dominated set. Technical report, Institute of Mathematical Modeling, Technical University of Denmark. IMM Technical Report IMM-REP-1998-7.

He, J. and X. Yao (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence 127*, 57 – 85.

He, J. and X. Yao (2002). From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 6*(5), 495 – 511.

Helbig, S. and D. Pateva (1994). On several concepts for ϵ-efficiency. *OR Spektrum 16*(3), 179–186.

Horn, J. (1997). Multicriterion Decision Making. In T. Bäck, D. Fogel, and Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, pp. F1.9:1–15. Bristol, UK: IOP Publishing and Oxford University Press.

Hrbacek, K. and T. Jech (1999). *Introduction to Set Theory*. New York: Marcel Dekker, Inc.

Hwang, C.-L. and A. S. M. Masud (1979). *Multiple Objectives Decision Making—Methods and Applications*. Berlin: Springer.

Kaliszewski, I. (1994). *Quantitative Pareto analysis by cone separation technique*. Boston: Kluwer.

Keeney, R. L. and H. Raiffa (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley.

Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi (1983). Optimization by simulated annealing. *Science 220*(4598), 671–680.

Knowles, J. and D. Corne (2002). On metrics for comparing non-dominated sets. In *Congress on Evolutionary Computation (CEC 2002)*, Piscataway, NJ, pp. 711–716. IEEE Press.

Knowles, J. D. and D. W. Corne (1999). Approximating the non-dominated front using the Pareto Archived Evolution Strategy. Technical Report RUCS/1999/TR/005/A, Department of Computer Science, University of Reading, UK.

Knowles, J. D. and D. W. Corne (2000). Approximating the non-dominated front using the Pareto Archived Evolution Strategy. *Evolutionary Computation 8*(2), 149–172.

Kursawe, F. (1990). Evolutionsstrategien für die Vektoroptimierung. Diplomarbeit, Universität Dortmund, Germany (in German).

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. In H.-P. Schwefel and R. Männer (Eds.), *Parallel Problem Solving from Nature (PPSN)*, pp. 193–197. Springer.

Laumanns, M., L. Thiele, E. Zitzler, E. Welzl, and K. Deb (2002). Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving From Nature (PPSN VII)*, Lecture Notes in Computer Science Vol. 2439, pp. 44–53. Springer.

Laumanns, M., E. Zitzler, and L. Thiele (2000). A unified model for multi-objective evolutionary algorithms with elitism. In *Congress on Evolutionary Computation (CEC 2000)*, pp. 46–53. IEEE Press.

Laumanns, M., E. Zitzler, and L. Thiele (2001). On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In E. Zitzler et al. (Eds.), *Evolutionary Multi-Criterion Optimization (EMO 2001)*, Lecture Notes in Computer Science Vol. 1993, pp. 181–196. Springer.

Ludmann, J. (1998). *Beeinflussung des Verkehrsablaufs auf Strassen: Analyse mit dem fahrzeugorientierten Verkehrssimulationsprogramm PELOPS*. Schriftenreihe Automobiltechnik. Forschungsgesellschaft Kraftfahrwesen mbH Aachen (in German).

Ludmann, J., D. Neunzig, M. Weilkes, and H. Wallentowitz (1999). The effectivity of new traffic-technologies and transportation-systems in suburban areas and on motorways. *International Transactions in Operational Research 6*(4), 423 – 439.

Martello, S. and P. Toth (1990). *Knapsack Problems: Algorithms and Computer Implementations*. Chichester: Wiley.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Boston: Kluwer.

Mühlenbein, H. (1992). How genetic algorithms really work: I. mutation and hillclimbing. In R. Männer and B. Manderick (Eds.), *Parallel Problem Solving from Nature (PPSN II*, pp. 15–25. Elsevier Science.

Papadimitriou, C. H. and M. Yannakakis (2000). The complexity of tradeoffs, and optimal access of web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science FOCS 2000*, pp. 86–92.

Pareto, V. (1896). *Cours D'Economie Politique*, Volume 1. Lausanne: F. Rouge.

Reuter, H. (1990). An approximation method for the efficiency set of multiobjective programming problems. *Optimization 21*, 905–911.

Rudolph, G. (1994). An evolutionary algorithm for integer programming. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.), *Parallel Problem Solving from Nature (PPSN III)*, pp. 139–148. Springer.

Rudolph, G. (1997a). *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, Hamburg.

Rudolph, G. (1997b). Modes of stochastic convergence. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, pp. B1.3:1–3. Bristol, UK: IOP Publishing and Oxford University Press.

Rudolph, G. (1998a). Evolutionary search for minimal elements in partially ordered finite sets. In V. Porto et al. (Eds.), *Evolutionary Programming VII, Proceedings of the 7th Annual Conference on Evolutionary Programming*, pp. 345–353. Springer.

Rudolph, G. (1998b). Finite markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae 35*, 67–89.

Rudolph, G. (1998c). On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *IEEE Int'l Conf. on Evolutionary Computation (ICEC'98)*, pp. 511–516. IEEE Press.

Rudolph, G. (2001). Evolutionary Search under Partially Ordered Fitness Sets. In *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, pp. 818–822. ICSC Academic Press.

Rudolph, G. and A. Agapie (2000). Convergence properties of some multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2000)*, Volume 2, pp. 1010–1016. IEEE Press.

Ruhe, G. and B. Fruhwirt (1990). ϵ-optimality for bicriteria programs and its application to minimum cost flows. *Computing 44*, 21–34.

Sayin, S. (2000). Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Math. Program., Ser. A 87*, 543–560.

Scharnow, J., K. Tinnefeld, and I. Wegener (2002). Fitness landscapes based on sorting and shortest paths problems. In J. J. M. Guervas et al. (Eds.), *Parallel Problem Solving From Nature (PPSN VII)*, Lecture Notes in Computer Science Vol. 2439, pp. 54–63. Springer.

Schott, J. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

Skiena, S. (1990). *Implementing Discrete Mathematics*. Redwood City, CA: Addison-Wesley.

Srinivas, N. and K. Deb (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation 2*(3), 221–248.

Thierens, D. (2003). Convergence Time Analysis for the Multi-objective Counting Ones Problem. In C. M. Fonseca et al. (Eds.), *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science Vol. 2632, pp. 354–364. Springer.

Ulungu, E., J. Teghem, P. Fortemps, and D. Tuyttens (1999). MOSA Method: A Tool for Solving Multiobjective Compinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis 8*(4), 221–236.

Van Veldhuizen, D. A. (1999, June). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph. D. thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University.

Vose, M. D. (1999). *The simple genetic algorithm: foundations and theory*. Cambridge, MA: MIT Press.

Wallentowitz, H. (2000). *Longitudinal Dynamics of Motor Vehicles*. Aachen: Forschungsgesellschaft Kraftfahrwesen mbH.

Wegener, I. (2000). Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R. Sarker, X. Yao, and M. Mohammadian (Eds.), *Evolutionary Optimization*, pp. 349–369. Kluwer.

Wu, J. and S. Azarm (2001, March). Metrics for quality assessment of a multiobjective design optimization solution set. *Transactions of the ASME, Journal of Mechanical Design 123*, 18–25.

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

Zitzler, E., M. Laumanns, and L. Thiele (2002). SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K. Giannakoglou et al. (Eds.), *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE).

Zitzler, E. and L. Thiele (1998). Multiobjective optimization using evolutionary algorithms — a comparative case study. In A. E. Eiben et al. (Eds.), *Parallel Problem Solving from Nature (PPSN V)*, Lecture Notes in Computer Science Vol. 1498, pp. 292–301. Springer.

Zitzler, E. and L. Thiele (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation 3*(4), 257–271.

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

Zurada, J. M., R. J. Marks II, and C. J. Robinson (Eds.) (1994). *Computational Intelligence: Imitating Life*. Piscataway, NJ: IEEE Press.