Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tea Tušar

Razvoj algoritma za večkriterijsko optimiranje z diferencialno evolucijo

Magistrska naloga

Mentor: akad. prof. dr. Ivan Bratko
Somentor: doc. dr. Bogdan Filipič

Ljubljana, 2007

Razvoj algoritma za večkriterijsko optimiranje z diferencialno evolucijo

POVZETEK

V številnih praktičnih problemih optimiranja je kakovost rešitev opredeljena z več po naravi različnimi kriteriji, kot so na primer cena, učinek in koristnost rešitve. Mnogokrat si ti kriteriji nasprotujejo, iz česar sledi, da namesto ene obstaja več optimalnih rešitev, kjer vsaka predstavlja nek kompromis med kriteriji. Klasične metode večkriterijske optimizacijske probleme rešujejo tako, da kriterije z uporabo neke funkcije (pogosto je to utežena vsota) prevedejo v en sam kriterij, ki ga nato optimirajo. Šele evolucijski algoritmi večkriterijske probleme rešujejo tako, da vse kriterije obravnavajo neodvisno in kot rezultat vrnejo množico kompromisnih rešitev.

Uveljavljeni večkriterijski evolucijski algoritmi, kot so NSGA-II, SPEA2 in IBEA, iščejo rešitve z istim genetskim algoritmom in se razlikujejo le v načinu določanja najboljših rešitev, ki mu pravimo kriterijska selekcija. V magistrski nalogi predstavljamo nov algoritem DEMO (angl. Differential Evolution for Multiobjective Optimization), ki lahko uporablja poljuben pristop za kriterijsko selekcijo in rešitve tvori z diferencialno evolucijo – evolucijskim algoritmom, ki v reševanju enokriterijskih problemov pogosto doseže boljše rezultate kot genetski algoritmi. DEMO smo implementirali v štirih različicah z različnimi pristopi za kriterijsko selekcijo in ga primerjali z ustreznimi algoritmi NSGA-II, SPEA2 in IBEA (v dveh različicah). Rezultati obsežnih poskusov kažejo, da so različice algoritma DEMO na večini testnih problemov signifikantno boljše kot primerjani algoritmi. Iz tega lahko zaključimo, da je diferencialna evolucija učinkovitejša od genetskih algoritmov tudi na večkriterijskih optimizacijskih problemih.

Na koncu različico algoritma DEMO, ki doseže najboljšo razporeditev vektorjev v prostoru kriterijev, uporabimo na praktičnem problemu, kjer je treba nastaviti parametre gradnje odločitvenih dreves tako, da so dobljena drevesa čim bolj točna in čim manjša. DEMO na preizkušenih domenah strojnega učenja najde dobre kompromise med točnimi in majhnimi drevesi in tako uporabnikom olajša izbiro najbolj zaželene rešitve.

Ključne besede: *večkriterijsko optimiranje, evolucijski algoritmi, diferencialna evolucija, Pareto optimalnost, večkriterijski testni problemi, točnost in velikost odločitvenih dreves*

University of Ljubljana
Faculty of Computer and Information Science

Tea Tušar

Design of an Algorithm for Multiobjective Optimization with Differential Evolution

M.Sc. Thesis

Supervisor: Acad. Prof. Dr. Ivan Bratko
Co-Supervisor: Assist. Prof. Dr. Bogdan Filipič

Ljubljana, 2007

Design of an Algorithm for Multiobjective Optimization with Differential Evolution

ABSTRACT

In many real-world optimization problems the quality of solutions is determined with several fundamentally different objectives, such as, for example, cost, performance and profit. These objectives are often mutually conflicting thus yielding several optimal solutions, where each of them represents a different tradeoff between the objectives. Classical methods solve multiobjective optimization problems by first transforming all objectives into a single one (often using the weighted sum approach) and then optimizing the resulting objective. Evolutionary algorithms, on the other hand, treat all objectives independently and provide as a result a set of tradeoff solutions.

Several state-of-the-art multiobjective evolutionary algorithms, such as NSGA-II, SPEA2 and IBEA use the same genetic algorithm to search solutions and differ only in the procedure used for selecting the best solutions, called environmental selection. In this thesis, a novel algorithm DEMO (Differential Evolution for Multiobjective Optimization) is presented, which can incorporate an arbitrary environmental selection procedure and generates the solutions with differential evolution—an evolutionary algorithm that often outperforms genetic algorithms on singleobjective optimization problems. DEMO was implemented in four variants with different environmental selection procedures and was compared to the corresponding algorithms NSGA-II, SPEA2 and IBEA (in two variants). Results of extensive experiments show that DEMO variants are significantly better than the compared algorithms on most test problems. Therefore, we can conclude that differential evolution is more efficient than genetic algorithms also on multiobjective optimization problems.

Finally, the DEMO variant which achieves the best distribution of vectors in the objective space is used on the practical problem of setting the parameters of decision tree building algorithms in such a way that the obtained trees are as accurate and small as possible. On the tested machine learning domains DEMO finds good tradeoffs between accurate and small decision trees thus enabling the users to easily choose the most desired solution.

Keywords: *multiobjective optimization, evolutionary algorithms, differential evolution, Pareto optimality, multiobjective benchmark problems, accuracy and size of decision trees*

Acknowledgments

During my research and while writing this thesis many special people have helped me with their guidance and advice or by simply being there for me. These are my thanks.

First, I wish to acknowledge my supervisors. Bogdan Filipič patiently guided and encouraged me since I first started working at the Jožef Stefan Institute. He taught me a variety of things: from research skills, like tackling scientific problems and presenting them in Slovene and English, to everyday abilities, such as reading street maps and having patience (he is still working on this one). He has been a true mentor to me and I will always be grateful to him. Many thanks to Ivan Bratko for his time, useful comments and ideas, but most of all, for setting with his work an example for all Slovene students and researchers in Computer Science.

Special thanks to Thiemo Krink for introducing me to Differential Evolution, revising the manuscript and consenting to be on my thesis committee. I am also thankful to Janez Demšar and Eckart Zitzler for helping me in my struggle with statistics and Bernard Ženko for reading the manuscript and the numerous LaTeX and gnuplot tips.

I am very grateful to the participants of the AVN workshops for their comments and advice. Our discussions considerably influenced this work. Thanks also to Matjaž Gams, the head of the Department of Intelligent Systems, who gave me the opportunity to attend the EMO 2005 and EMO 2007 conferences. Both events inspired me and had a great impact on my research.

I also wish to thank all members of the Departments of Intelligent Systems, Knowledge Technologies and Computer Systems at the Jožef Stefan Institute, who provided a friendly working environment. Special thanks go to Andraž Bežek, Bernard Ženko and Mitja Luštrek for always being there for me and often acting as my personal chauffeurs.

Finally, my gratitude goes to my friends and family for their eternal support and patience. Special thanks to my mother, who saw me through my studies, and to my husband—none of this would be possible without his love and support. Thank you so much!

To Dejan

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.2 Contributions . . . 3

1.3 Organization of the thesis . . . 3

2 Background **5**

2.1 Properties of multiobjective optimization . . . 5

2.1.1 Pareto dominance and Pareto optimality . . . 5

2.1.2 Preference-based and ideal principle . . . 7

2.1.3 Approximation sets . . . 9

2.2 Assessment of multiobjective optimizers . . . 11

2.2.1 Dominance ranking . . . 11

2.2.2 Quality indicators . . . 12

2.2.3 Empirical attainment function . . . 17

2.2.4 Multiple testing issues . . . 18

2.3 Multiobjective evolutionary algorithms . . . 18

2.3.1 Basic genetic algorithm . . . 19

2.3.2 Nondominated sorting . . . 20

2.3.3 Strength Pareto approach . . . 21

2.3.4 Indicator-based selection . . . 23

3 Differential Evolution for Multiobjective Optimization **27**

3.1 Differential evolution . . . 27

3.1.1 Algorithm outline . . . 28

3.1.2 The DE/rand/1/bin strategy . . . 28

3.1.3 Advantages, limitations and applications . . . 31

3.2 First adaptations to multiobjective optimization 31
3.3 The DEMO algorithm 33
3.4 Recent adaptations to multiobjective optimization 35
3.5 Comparison between DEMO and the basic GA 36
3.5.1 Benchmark problems 36
3.5.2 Parameters of the algorithms 37
3.5.3 Performance assessment 38
3.5.4 Results and discussion 39
3.5.5 Summary 47
3.6 Comparison of DEMO variants 48
3.6.1 Dominance ranking 48
3.6.2 Configuration of approximation sets 49
3.6.3 Summary 52

4 Optimizing Accuracy and Size of Decision Trees 53
4.1 Introduction 53
4.1.1 Decision trees for classification 53
4.1.2 Optimization problem 56
4.1.3 Related work 56
4.2 Optimization with DEMO 58
4.2.1 Experiments on the modified problem 58
4.2.2 Experiments on the original problem 63
4.2.3 Analysis of the decision space 65
4.2.4 A real-world example 66
4.3 Summary 69

5 Conclusion 71

References 73

A Abbreviations 79

B Complete Results 81

C Razširjen povzetek v slovenskem jeziku 91
C.1 Uvod 91

C.2 Posebnosti večkriterijskega optimiranja 92
C.2.1 Osnovne definicije 92
C.2.2 Dva pristopa 93
C.2.3 Aproksimacijske množice 93
C.3 Diferencialna evolucija za večkriterijsko optimiranje 94
C.3.1 Algoritem DEMO 94
C.3.2 Primerjava algoritma DEMO z večkriterijskimi genetskimi algoritmi . . 95
C.3.3 Primerjava različic algoritma DEMO 95
C.4 Optimiranje točnosti in velikosti odločitvenih dreves 96
C.4.1 Opis problema 96
C.4.2 Optimiranje z algoritmom DEMO 96
C.5 Sklep 97

Izjava o avtorstvu **99**

Chapter 1

Introduction

Many real-world optimization problems involve optimization of several, often conflicting objectives. Consequently, instead of a single optimal solution, a set of optimal solutions (called *Pareto optimal set*) exists for such problems. Each of the Pareto optimal solutions represents a different tradeoff between the objectives and in the absence of preference information, none of them can be said to be better than others.

Because of multiple objectives we deal with two spaces: the space of decision variables (or *decision space*), where the search is conducted, and the space of objectives (or *objective space*), where the solutions are evaluated. While classical optimization methods solve multiobjective problems by converting them into singleobjective ones (thus degenerating the multidimensional objective space into a one-dimensional space), evolutionary algorithms can tackle the optimization of all objectives simultaneously. Since two spaces exist, multiobjective evolutionary algorithms tend to work on two levels: they use a *search procedure* to explore the decision space and a so-called *environmental selection procedure* to select the best solutions according to their positioning in the objective space. Although the two are not completely independent (the search in the decision space is usually guided by the objective values of the solutions), they can be seen as two separate tasks.

In the last twenty years, several multiobjective evolutionary algorithms have been proposed. While these algorithms present different approaches to environmental selection, most of them, including the popular NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001) and IBEA (Zitzler and Künzli, 2004), use a nearly identical Genetic Algorithm (GA) for search-

ing the decision space. Only recently, approaches relying on other evolutionary algorithms, such as Differential Evolution (DE) (Storn and Price, 1997), were proposed.

1.1 Motivation

The motivation for this work was three-fold. First, we wanted to design an efficient algorithm for multiobjective optimization, which would use DE for exploration of the decision space in a straightforward way. While DE-based algorithms for multiobjective optimization have already been proposed in the past (see the related work presented in Section 3.2), they either disregarded DE's basic characteristic of comparing every new solution to its parent or applied it too strictly for multiobjective optimization. Moreover, the existent approaches use only the environmental selection method from NSGA-II, while our aim was to allow combinations of DE-based exploration of the decision space and arbitrary approaches to environmental selection.

Second, since DE often outperforms GAs on singleobjective problems, we wanted to check if this holds also for multiobjective problems. Similar comparisons reported so far (see Sections 3.2 and 3.4) lack: (1) a wide choice of difficult test problems with more than two objectives, (2) proper performance assessment, and (3) inferences about algorithm performance based on statistical tests. Furthermore, when in the past a DE-based algorithm was compared to a GA-based one, the two algorithms often differed in many aspects, not only in the approach used for exploring the decision space. Therefore, whatever the results of such a comparison, they cannot be attributed solely to the exploration approach used.

The third goal of this thesis was to apply our DE-based multiobjective optimization algorithm to the real-world problem of setting the parameters of machine learning algorithms so that the resulting theories would be accurate and simple. Many domain experts who use machine learning algorithms for finding theories that would explain their data are not familiar with these algorithms. They usually do not know how to set the parameters of machine learning algorithms in order to produce the desired results. Moreover, they rarely know beforehand exactly what kind of theory they are looking for. Our approach could help them by exploring the parameter space of the learning algorithms while searching for theories with highest prediction accuracy and lowest complexity. The resulting set of the best found theories gives the users the possibility of comparing the theories among themselves and provides an additional insight into the data. All this helps the users to choose the theory that best suits their needs.

In the following, we limit our discussion on machine learning theories to decision trees for classification, where the best trees (or theories) are regarded as those that are accurate and small.

1.2 Contributions

The first contribution of this thesis is the design of the algorithm called Differential Evolution for Multiobjective Optimization (DEMO), which explores the decision space using DE and selects the best solutions for the next population using an arbitrary environmental selection procedure. DEMO is implemented in four variants: $\text{DEMO}^{\text{NS-II}}$, DEMO^{SP2}, $\text{DEMO}^{\text{IB}_{\varepsilon+}}$ and $\text{DEMO}^{\text{IB}_{\text{HD}}}$, which use environmental selection mechanisms from NSGA-II, SPEA2, $\text{IBEA}_{\varepsilon+}$ and IBEA_{HD}, respectively.

Additionally, we define the so-called basic genetic algorithm: a multiobjective evolutionary algorithm, which uses a GA to explore the decision space and an arbitrary environmental selection procedure to select the best solutions. In this way, the NSGA-II, SPEA2, $\text{IBEA}_{\varepsilon+}$ and IBEA_{HD} algorithms can be simply seen as special cases of the basic GA, enabling us to make pairwise comparison between NSGA-II and $\text{DEMO}^{\text{NS-II}}$, SPEA2 and DEMO^{SP2}, $\text{IBEA}_{\varepsilon+}$ and $\text{DEMO}^{\text{IB}_{\varepsilon+}}$, and IBEA_{HD} and $\text{DEMO}^{\text{IB}_{\text{HD}}}$. The algorithms are compared on 16 state-of-the-art benchmark problems (each with 2, 3 and 4 objectives), where the four variants of DEMO significantly outperform their GA-based counterparts.

Finally, we apply DEMO to the problem of finding parameter settings of a decision tree building algorithm so that the resulting decision trees are accurate and small. This practical study and the comparison study on artificial benchmark problems give additional evidence of the usefulness of DE in numeric optimization.

1.3 Organization of the thesis

Chapter 2 describes the background required for proper understanding of the challenges posed by multiple objectives. Beside the formal definitions of the concepts specific to multiobjective optimization, the chapter presents the recommended performance measures to be used for assessing multiobjective optimization algorithms. The chapter ends with the introduction of the basic GA and a detailed description of the different environmental selection approaches used in this thesis.

Chapter 3 is dedicated to DE and the DEMO algorithm. Starting with a detailed description of DE, it reviews the related work—both preceding and following the publication of DEMO. Two comparison studies follow: the first compares four GA-based algorithms to the corresponding DEMO variants, while the second explores the differences among DEMO variants.

Chapter 4 deals with the problem of optimizing accuracy and size of decision trees. After a short introduction to classification problems and the discussion of the related work, the experiments with DEMO are presented.

Chapter 5 ends the thesis with concluding remarks and ideas for future work.

Chapter 2

Background

This chapter reviews the essential background knowledge on multiobjective optimization and multiobjective evolutionary algorithms. Section 2.1 describes the specialities of multiobjective optimization that distinguish multiobjective optimization problems from singleobjective ones. Among them are, for example, the concept of Pareto dominance and the existence of incomparable solutions and incomparable sets of solutions. All this has a great impact also on the performance assessment of multiobjective optimization algorithms. This important topic is covered in Section 2.2. Finally, Section 2.3 presents evolutionary algorithms and their adjustment to multiobjective optimization, showing in more detail the particular mechanisms of three state-of-the-art algorithms: NSGA-II, SPEA2 and IBEA.

2.1 Properties of multiobjective optimization

In multiobjective optimization, we wish to simultaneously optimize several (possibly conflicting) objectives. This single demand yields many principles that make multiobjective optimization very different from (and usually more challenging than) singleobjective optimization. The basic principles are described here—see (Knowles et al., 2006) for more details.

2.1.1 Pareto dominance and Pareto optimality

The multiobjective optimization problem (MOP) consists of finding the optimum of a func-

tion

$$\begin{aligned} \boldsymbol{f} &: X \to Z \\ \boldsymbol{f} &: (x_1, \ldots, x_n) \mapsto (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)), \end{aligned}$$

where X is an n-dimensional *decision space*, and Z is an m-dimensional *objective space* ($m \geq 2$). Each solution $\boldsymbol{x} \in X$ is called a *decision vector*, while the corresponding element $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) \in Z$ is an *objective vector*. From this moment on we assume without loss of generality that $Z \subseteq \mathbb{R}^m$ and the objectives $f_j : X \to \mathbb{R}$ are to be minimized for all $j \in \{1, \ldots, m\}$.

Definition 2.1 (**Pareto dominance of vectors**)**.** The objective vector $\boldsymbol{z}^1$ *dominates* the objective vector $\boldsymbol{z}^2$ ($\boldsymbol{z}^1 \prec \boldsymbol{z}^2$) $\overset{def}{\iff}$ $z_j^1 \leq z_j^2$ for all $j \in \{1, \ldots, m\}$ and $z_k^1 < z_k^2$ for at least one $k \in \{1, \ldots, m\}$.

Definition 2.2 (**Weak Pareto dominance of vectors**)**.** The objective vector $\boldsymbol{z}^1$ *weakly dominates* the objective vector $\boldsymbol{z}^2$ ($\boldsymbol{z}^1 \preceq \boldsymbol{z}^2$) $\overset{def}{\iff}$ $z_j^1 \leq z_j^2$ for all $j \in \{1, \ldots, m\}$.

Definition 2.3 (**Strict Pareto dominance of vectors**)**.** The objective vector $\boldsymbol{z}^1$ *strictly dominates* the objective vector $\boldsymbol{z}^2$ ($\boldsymbol{z}^1 \prec\prec \boldsymbol{z}^2$) $\overset{def}{\iff}$ $z_j^1 < z_j^2$ for all $j \in \{1, \ldots, m\}$.

When $\boldsymbol{z}^1 = \boldsymbol{f}(\boldsymbol{x}^1)$, $\boldsymbol{z}^2 = \boldsymbol{f}(\boldsymbol{x}^2)$ and $\boldsymbol{z}^1$ (weakly or strictly) dominates $\boldsymbol{z}^2$, we say that the solution $\boldsymbol{x}^1$ (weakly or strictly) dominates the solution $\boldsymbol{x}^2$. Note that $\boldsymbol{z}^1 \prec\prec \boldsymbol{z}^2 \Longrightarrow \boldsymbol{z}^1 \prec \boldsymbol{z}^2 \Longrightarrow \boldsymbol{z}^1 \preceq \boldsymbol{z}^2$.

The weak Pareto dominance is a natural generalization of the $\leq$ relation on $\mathbb{R}$. While $\leq$ induces a total order on $\mathbb{R}$, the $\preceq$ relation induces only a partial order on $\mathbb{R}^m$. This means that two objective vectors (and therefore two solutions) can be incomparable.

Definition 2.4 (**Incomparability of vectors**)**.** The objective vectors $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$ are *incomparable* ($\boldsymbol{z}^1 \parallel \boldsymbol{z}^2$) $\overset{def}{\iff}$ $\boldsymbol{z}^1 \not\preceq \boldsymbol{z}^2$ and $\boldsymbol{z}^2 \not\preceq \boldsymbol{z}^1$.

In case of conflicting objectives the multiobjective optimization problem can have multiple optimal solutions.

Definition 2.5 (**Pareto optimality**)**.** The solution $\boldsymbol{x}^*$ and its corresponding objective vector $\boldsymbol{z}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ are *Pareto optimal* $\overset{def}{\iff}$ there exists no $\boldsymbol{z} \in Z$ such that $\boldsymbol{z} \prec \boldsymbol{z}^*$.

All Pareto optimal solutions compose the *Pareto optimal set*, while the corresponding objective vectors constitute the *Pareto optimal front*.

Example 2.1. *For a better understanding of these definitions consider the MOP presented in Figure 2.1. While x_1 and x_2 denote the decision variables, z_1 and z_2 indicate the objectives to be minimized. For solutions $\boldsymbol{x}^i$ and the corresponding objective vectors $\boldsymbol{z}^i = \boldsymbol{f}(\boldsymbol{x}^i)$, where $i \in \{1,2,3,4\}$, the following statements can be made: $\boldsymbol{x}^3$ strictly dominates $\boldsymbol{x}^4$, while all other pairs of solutions are mutually incomparable. In addition, $\boldsymbol{x}^3$ is Pareto optimal as are all the solutions represented by black points.*

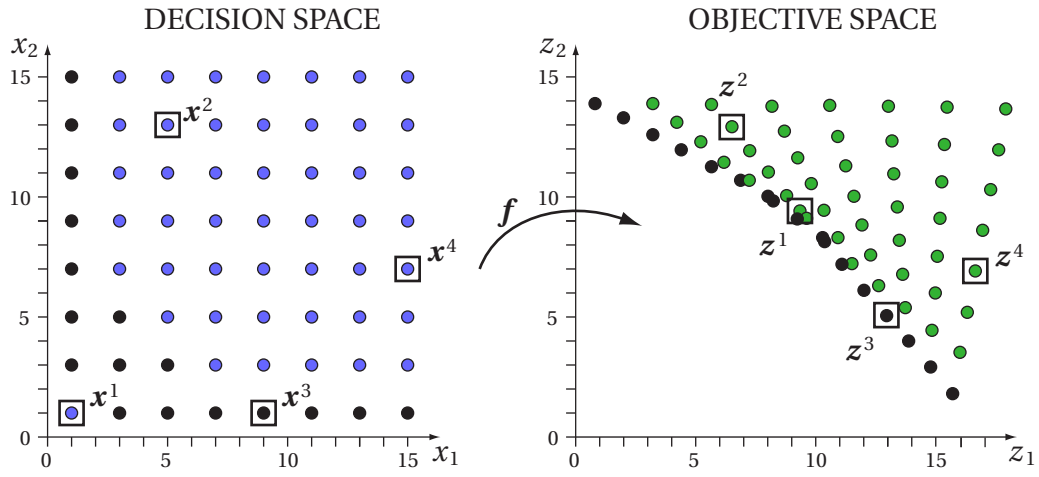


Figure 2.1: The decision and objective space for a MOP. Black points represent Pareto optimal decision and objective vectors.

2.1.2 Preference-based and ideal principle

Each element of the Pareto optimal front represents a tradeoff between the objectives. Without additional preference information we cannot decide among different Pareto optimal solutions. Like in singleobjective optimization, the ultimate goal of multiobjective optimization (at least from the user's perspective) is to obtain a single Pareto optimal solution. This can be done using either the preference-based or the ideal principle as shown in Figure 2.2 (Deb, 2001).

Following the *preference-based principle*, the multiobjective optimization problem is first transformed into a singleobjective one according to some preference for the objectives. This can be achieved, for example, by combining the objectives using a weighted sum. A single solution to the multiobjective problem is then acquired by solving the corresponding singleobjective problem. Using the preference-based principle when the preference for the

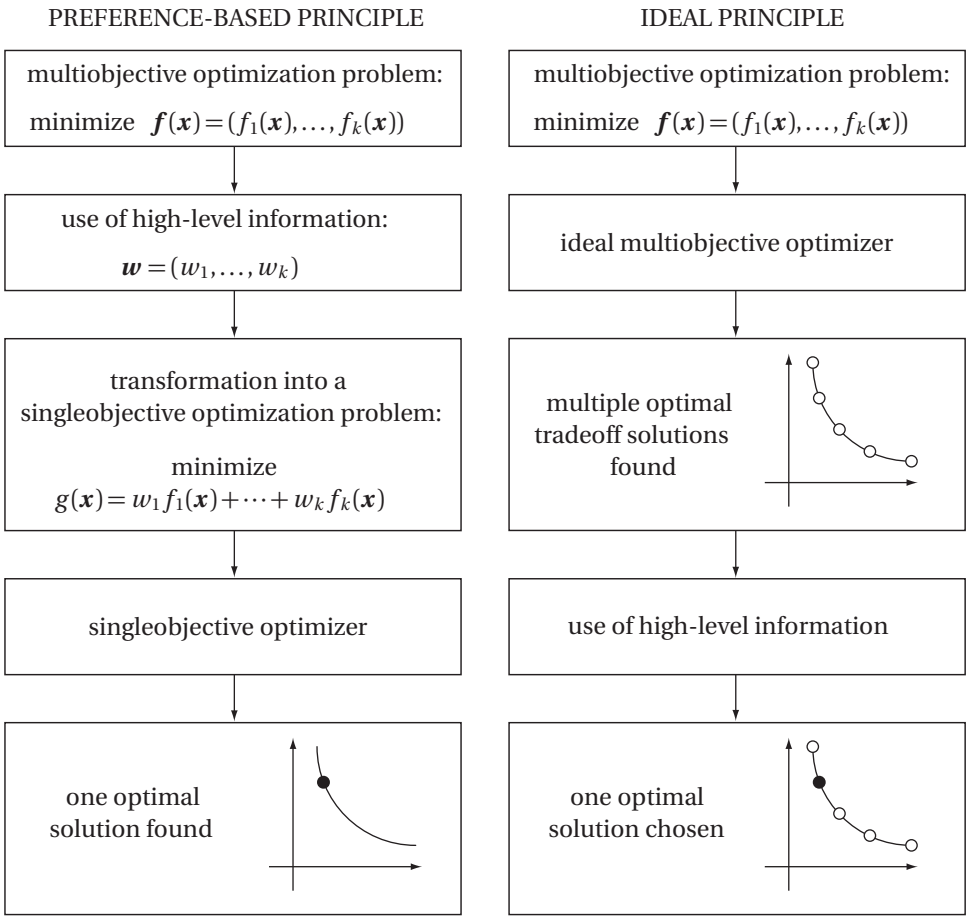


Figure 2.2: The preference-based and ideal principle in multiobjective optimization.

objectives is unknown has some disadvantages. For example, the solutions acquired by this principle depend on the function with which the multiobjective problem is transformed into a singleobjective one. With a different transformation, some other solution could be found. Moreover, some of the functions that are most often used for this transformation (such as the weighted sum) are incapable of reaching the concave portions of the Pareto optimal front.

Using the *ideal principle*, on the other hand, the multiobjective problem is first solved and only then the preference information is used to select a single solution among several alternatives. The ideal principle is ideal in the sense that it does not demand from the user to set a preference for the objectives before optimization. Only when several tradeoff solutions are known, the users chooses the preferred one among them. It is of course reasonable to use this principle only when the preference for the objectives is unknown beforehand.

Otherwise, the preference-based approach should be used.

However, we are interested in providing the user with a set of tradeoff solutions and we only consider methods that follow the ideal principle in this work.

2.1.3 Approximation sets

An algorithm employed for solving MOPs is called a *multiobjective optimizer*. The result of a multiobjective optimizer that follows the ideal principle is usually a set of mutually incomparable solutions, called *Pareto set approximation*, while the corresponding objective vectors form the *Pareto front approximation* or *approximation set* for short.

Definition 2.6 (**Approximation set**)**.** Let $\mathcal{Z} \subseteq Z$ be a set of objective vectors. $\mathcal{Z}$ is called an *approximation set* $\overset{def}{\iff}$ any element of $\mathcal{Z}$ does not weakly dominate any other element in $\mathcal{Z}$. The set of all approximation sets is denoted as Ω.

The objective vectors from a set $\mathcal{Z} \subseteq Z$ which are not dominated by any other vector from $\mathcal{Z}$ are often called *nondominated* vectors.

For approximation sets, the relations of Pareto dominance, weak and strict Pareto dominance and incomparability can be defined similarly as for separate vectors (see the definitions in Subsection 2.1.1).

Definition 2.7 (**Pareto dominance of approximation sets**)**.** The approximation set $\mathcal{Z}_1$ *dominates* the approximation set $\mathcal{Z}_2$ ($\mathcal{Z}_1 \prec \mathcal{Z}_2$) $\overset{def}{\iff}$ every vector from $\mathcal{Z}_2$ is dominated by at least one vector from $\mathcal{Z}_1$.

Definition 2.8 (**Weak Pareto dominance of approximation sets**)**.** The approximation set $\mathcal{Z}_1$ *weakly dominates* the approximation set $\mathcal{Z}_2$ ($\mathcal{Z}_1 \preceq \mathcal{Z}_2$) $\overset{def}{\iff}$ every vector from $\mathcal{Z}_2$ is weakly dominated by at least one vector from $\mathcal{Z}_1$.

Definition 2.9 (**Strict Pareto dominance of approximation sets**)**.** The approximation set $\mathcal{Z}_1$ *strictly dominates* the approximation set $\mathcal{Z}_2$ ($\mathcal{Z}_1 \prec\prec \mathcal{Z}_2$) $\overset{def}{\iff}$ every vector from $\mathcal{Z}_2$ is strictly dominated by at least one vector from $\mathcal{Z}_1$.

Definition 2.10 (**Incomparability of approximation sets**)**.** The approximation sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are *incomparable* ($\mathcal{Z}_1 || \mathcal{Z}_2$) $\overset{def}{\iff}$ $\mathcal{Z}_1 \not\preceq \mathcal{Z}_2$ and $\mathcal{Z}_2 \not\preceq \mathcal{Z}_1$.

Additionally, for approximation sets the better relation is defined in the following way.

Definition 2.11. The approximation set $\mathcal{Z}_1$ is *better* than the approximation set $\mathcal{Z}_2$ ($\mathcal{Z}_1 \triangleleft \mathcal{Z}_2$) $\overset{def}{\Longleftrightarrow} \mathcal{Z}_1 \preceq \mathcal{Z}_2$ and $\mathcal{Z}_1 \neq \mathcal{Z}_2$.

Note that $\mathcal{Z}_1 \prec\prec \mathcal{Z}_2 \Longrightarrow \mathcal{Z}_1 \prec \mathcal{Z}_2 \Longrightarrow \mathcal{Z}_1 \triangleleft \mathcal{Z}_2 \Longrightarrow \mathcal{Z}_1 \preceq \mathcal{Z}_2$. A visual representation of these definitions is provided in the following example.

Example 2.2. *Consider again the MOP from Example 2.1. Figure 2.3 shows three approximation sets $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$. While $\mathcal{A}_1$ and $\mathcal{A}_2$ are incomparable, $\mathcal{A}_1$ is better than $\mathcal{A}_3$ and $\mathcal{A}_2$ strictly dominates $\mathcal{A}_3$.*

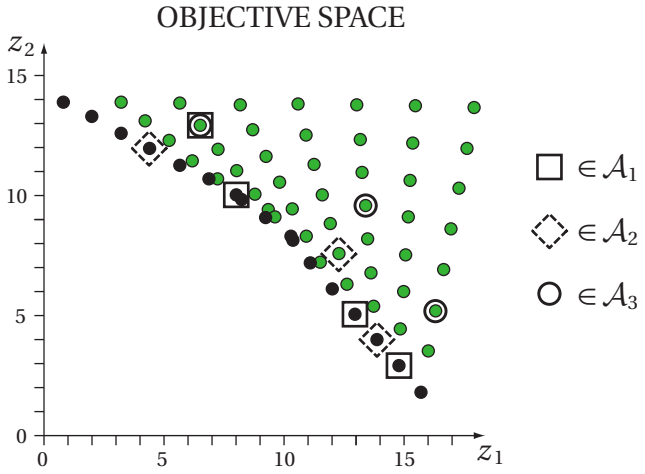


Figure 2.3: Three approximation sets for an example MOP. Black points represent Pareto optimal objective vectors.

Besides searching for approximation sets as close to the Pareto optimal front as possible, multiobjective optimizers usually try to attain and maintain a uniform spread of objective vectors along the Pareto optimal front. In this way, the user can choose among a wide selection of tradeoffs. While being able to achieve an even spread of vectors is often a very desirable property of a multiobjective optimizer, it is not compliant with the Pareto dominance relation defined previously. This means that the convergence to the Pareto optimal front is regarded as the only goal of multiobjective optimization, while the even spread of vectors is merely a preferred property and cannot be formally considered a second goal of multiobjective optimization. This contradicts the notion of two goals in multiobjective optimization found often in the literature (Deb, 2001).

2.2 Assessment of multiobjective optimizers

Performance assessment of optimizers on the given problem should always take into consideration the quality of the obtained solutions as well as the computational cost required to produce them. A frequently used procedure for comparing performance of optimizers on a problem is to first set a maximal number of function evaluations (or CPU time) at disposal and then compare the quality of the solutions produced by the optimizers within this limitation.

In multiobjective optimization, the quality of solutions produced by the optimizers is assessed by comparing their approximation sets. If the optimizers use stochastic methods, they should be run several times on the given MOP and the resulting sets of approximation sets should be compared. Because of the existence of incomparable solutions and incomparable approximation sets, there exist many different approaches to performance assessment of multiobjective optimizers. In this work we follow the guidelines from (Knowles et al., 2006) and use *dominance ranking*, *quality indicators* and the *empirical attainment function* for comparing (sets of) approximation sets.

The choice of the right benchmark MOPs is an additional important aspect to be considered when comparing multiobjective optimizers. This topic will not be covered here—see Subsection 3.5.1 for a description of the benchmark MOPs used in this work.

2.2.1 Dominance ranking

Two multiobjective optimizers A and B can be compared according to the *dominance ranks* of their approximation sets. Suppose that both algorithms were run r times, which yields approximation sets $\mathcal{A}_1,\ldots,\mathcal{A}_r$ for the first optimizer and approximation sets $\mathcal{B}_1,\ldots,\mathcal{B}_r$ for the second one. First, we gather all approximation sets in the collection $\boldsymbol{C} = \{\mathcal{C}_1,\ldots,\mathcal{C}_{2r}\}$. Then, each approximation set is ranked according to the number of approximation sets that are better than the selected approximation set:

$$dom\_rank(\mathcal{C}_i) = 1 + \left|\left\{\mathcal{C}_j \in \boldsymbol{C} \mid \mathcal{C}_j \triangleleft \mathcal{C}_i\right\}\right| \quad \text{for all} \quad i \in 1,\ldots,2r.$$

In this way, we obtain two sets of ranks: $\{dom\_rank(\mathcal{A}_1),\ldots,dom\_rank(\mathcal{A}_r)\}$ for the first and $\{dom\_rank(\mathcal{B}_1),\ldots,dom\_rank(\mathcal{B}_r)\}$ for the second optimizer. At this point, one of the statistical ranking tests can be applied to determine if there exists a significant difference between the values of the two sets. Note that the lower the ranks, the better the corresponding approximation sets.

While dominance ranking possesses the desired property of evaluating approximation sets based solely on Pareto dominance (and not on some preference for the objectives), this limits its expressive power and is therefore at the same time its weakness (see Example 2.3). In addition, as was noted in (Tušar and Filipič, 2007), dominance ranking sometimes performs differently from what would be expected (see also the explanation of the Figure 3.3 in Subsection 3.5.4). Consider the case when the approximation sets $\mathcal{A}_1,\ldots,\mathcal{A}_r$ dominate each other, while the approximation sets $\mathcal{B}_1,\ldots,\mathcal{B}_r$ are incomparable among themselves and incomparable to the sets $\mathcal{A}_1,\ldots,\mathcal{A}_r$. As a consequence, the ranks of the approximation sets $\mathcal{A}_1,\ldots,\mathcal{A}_r$ are mostly greater than 1, while the ranks of the approximation sets $\mathcal{B}_1,\ldots,\mathcal{B}_r$ are equal to 1. This means that according to dominance ranking, the optimizer B is better than the optimizer A, although none of the approximation sets $\mathcal{B}_1,\ldots,\mathcal{B}_r$ actually dominates any of the approximation sets $\mathcal{A}_1,\ldots,\mathcal{A}_r$.

Example 2.3. *Dominance ranks of the three approximation sets from Example 2.2 (see also Figure 2.3) are:* $dom\_rank(\mathcal{A}_1)=1$, $dom\_rank(\mathcal{A}_2)=1$ *and* $dom\_rank(\mathcal{A}_3)=3$. *While dominance ranking correctly establishes that approximation sets* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are better than* $\mathcal{A}_3$, *it does not distinguish between the two incomparable sets* $\mathcal{A}_1$ *and* $\mathcal{A}_2$. *Moreover, dominance ranking provides only ranks of approximation sets and not a measure of difference between approximation sets. Thus, we only know that* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are better than* $\mathcal{A}_3$, *but we do not know how big this difference is, and consequently, we do not know if there is a bigger difference between* $\mathcal{A}_1$ *and* $\mathcal{A}_3$ *than between* $\mathcal{A}_2$ *and* $\mathcal{A}_3$.

2.2.2 Quality indicators

A quality indicator is a function, which assigns a real value to any vector of approximation sets by means of some preference information. In this way (assuming the $\leq$ (or $\geq$) relation on $\mathbb{R}$), a quality indicator induces a total order on the set of approximation sets Ω. This means that for any two (vectors of) approximation sets, we can calculate which one is better according to the chosen quality indicator. While quality indicators can be of any order (see (Zitzler et al., 2003) for a detailed analysis of quality indicators), we deal only with unary and binary indicators here.

Definition 2.12 (Unary quality indicator). The function $I:\Omega\rightarrow\mathbb{R}$, which assigns a real value to any approximation set $\mathcal{Z}\in\Omega$, is called *unary quality indicator*.

Definition 2.13 (**Binary quality indicator**)**.** The function $I : \Omega \times \Omega \to \mathbb{R}$, which assigns a real value to any pair of approximation sets $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Omega \times \Omega$, is called *binary quality indicator*.

With quality indicators we get a finer distinction between approximation sets than with dominance ranking and we can quantify the differences even between incomparable approximation sets. Note, however, that each such inference is preference-based, i.e., with different indicators, approximation sets are evaluated differently. Although by definition, an arbitrary function $\Omega \to \mathbb{R}$ could serve as a unary quality indicator, it is important that an indicator is *Pareto compliant*.

Definition 2.14 (**Pareto compliant indicator**)**.** The unary indicator $I : \Omega \to \mathbb{R}$ is *Pareto compliant* $\overset{def}{\iff}$ for every pair of approximation sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$, for which $\mathcal{Z}_1 \preceq \mathcal{Z}_2$ and $I(\mathcal{Z}_1)$ is not worse than $I(\mathcal{Z}_2)$.

Pareto compliant indicators define refinements of the partial order induced by weak Pareto dominance, while Pareto non-compliant indicators can prefer dominated approximation sets to nondominated ones. In the following, we present three Pareto compliant quality indicators that are frequently used for performance assessment of multiobjective optimizers: the hypervolume indicator I_{H}, the unary additive epsilon indicator $I^1_{\varepsilon+}$ and the unary $R2$ indicator I^1_{R2}.

Hypervolume indicator I_{H}

For an approximation set $\mathcal{Z}$, the indicator value $I_{\mathrm{H}}(\mathcal{Z})$ represents the hypervolume of the subspace $S^{\boldsymbol{r}}_{\mathcal{Z}} \subseteq Z$, bounded by points from $\mathcal{Z}$ on one side and the reference point $\boldsymbol{r}$ on the other side (Zitzler and Thiele, 1999). The reference point must be chosen so that:

$$r_j > z_j \text{ for every } j \in \{1, \ldots, m\} \text{ and every objective vector } \boldsymbol{z} \in \mathcal{Z}.$$

The larger the hypervolume indicator, the better the approximation set. For this indicator, the following holds: if $\mathcal{Z}_1 \triangleleft \mathcal{Z}_2$, then $I_{\mathrm{H}}(\mathcal{Z}_1) > I_{\mathrm{H}}(\mathcal{Z}_2)$. Consequently, if $I_{\mathrm{H}}(\mathcal{Z}_1) < I_{\mathrm{H}}(\mathcal{Z}_2)$, then $\mathcal{Z}_1$ cannot be better than $\mathcal{Z}_2$.

Example 2.4. *Consider again the three approximation sets from Example 2.3 (their corresponding subspaces $S^{\boldsymbol{r}}_{\mathcal{A}_i}$ are shown in Figure 2.4). The hypervolume indicators for these sets are: $I_H(\mathcal{A}_1) = 96.8$, $I_H(\mathcal{A}_2) = 91.6$, and $I_H(\mathcal{A}_3) = 56.6$. This means that the approximation set $\mathcal{A}_1$ is found to be better than the other two sets according to this indicator and the chosen reference point.*

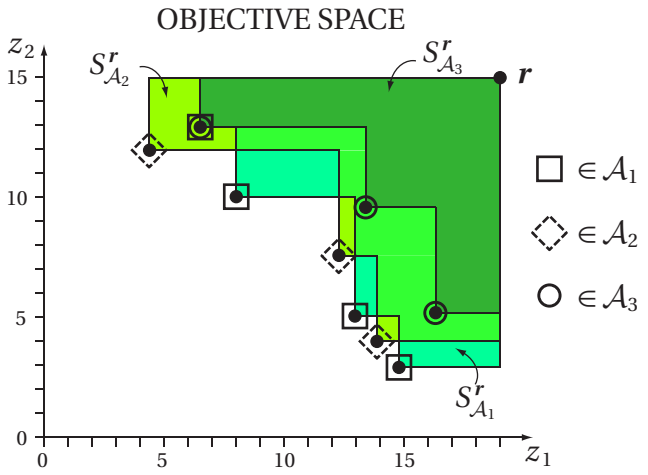


Figure 2.4: The reference point $\boldsymbol{r}$ and the resulting subspaces $S^{\boldsymbol{r}}_{\mathcal{A}_1}$, $S^{\boldsymbol{r}}_{\mathcal{A}_2}$ and $S^{\boldsymbol{r}}_{\mathcal{A}_3}$ for the three approximation sets from Figure 2.3.

Unary additive epsilon indicator $I^1_{\varepsilon+}$

Let us first look at the binary additive epsilon indicator (Zitzler et al., 2003), from which the unary is derived. $I_{\varepsilon+}(\mathcal{Z}_1, \mathcal{Z}_2)$ equals the smallest summand ε to which each vector from $\mathcal{Z}_2$ can be added in every objective such that the resulting approximation set is weakly dominated by $\mathcal{Z}_1$:

$$I_{\varepsilon+}(\mathcal{Z}_1, \mathcal{Z}_2) = \inf_{\varepsilon \in \mathbb{R}} \left\{ \text{for every } \boldsymbol{z}^2 \in \mathcal{Z}_2 \text{ exists a } \boldsymbol{z}^1 \in \mathcal{Z}_1 \text{ so that } \boldsymbol{z}^1 \preceq_{\varepsilon+} \boldsymbol{z}^2 \right\},$$

where $\preceq_{\varepsilon+}$ is the additive *ε-dominance relation:*

$$\boldsymbol{z}^1 \preceq_{\varepsilon+} \boldsymbol{z}^2 \overset{def}{\Longleftrightarrow} z^1_j \leq \varepsilon + z^2_j \text{ for every } j \in \{1, \ldots, m\}.$$

Let us illustrate the meaning of this definition with an example.

Example 2.5. *Examine once more the approximation sets $\mathcal{A}_1$ and $\mathcal{A}_2$ from Figure 2.5. If all vectors from $\mathcal{A}_2$ are increased by $\varepsilon = 2.1$ in each objective, we get the corresponding set $\mathcal{A}'_2$, which is weakly dominated by $\mathcal{A}_1$. Since 2.1 is the smallest possible value of ε so that $\mathcal{A}_1 \preceq \mathcal{A}'_2$, this means that $I_{\varepsilon+}(\mathcal{A}_1, \mathcal{A}_2) = 2.1$.*

Each binary quality indicator can be used as a unary indicator by replacing the second approximation set with a reference set of objective vectors $\mathcal{R}$:

$$I^1(\mathcal{Z}) = I(\mathcal{Z}, \mathcal{R}).$$

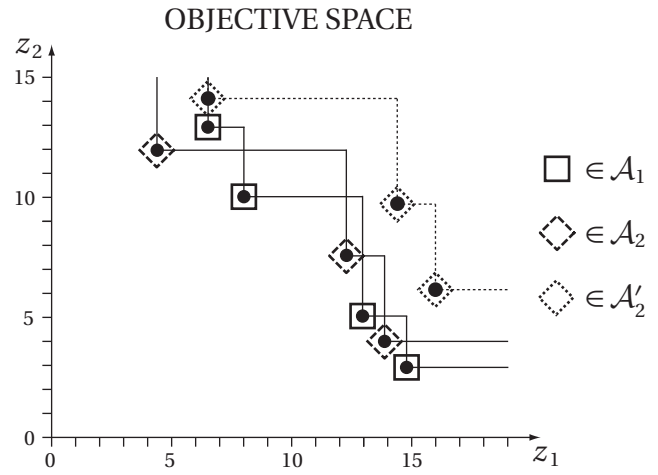


Figure 2.5: The approximation sets $\mathcal{A}_1$ and $\mathcal{A}_2$ and the shifted set $\mathcal{A}'_2$.

The ideal choice for the reference set is the Pareto optimal front. Since it is usually unknown, mutually incomparable objective vectors from the union of all available approximation sets can be used instead.

Smaller values of the unary additive epsilon indicator denote better sets. For this indicator, the following holds: if $\mathcal{Z}_1 \triangleleft \mathcal{Z}_2$, then $I^1_{\varepsilon+}(\mathcal{Z}_1) \leq I^1_{\varepsilon+}(\mathcal{Z}_2)$. Consequently, if $I^1_{\varepsilon+}(\mathcal{Z}_1) > I^1_{\varepsilon+}(\mathcal{Z}_2)$, then $\mathcal{Z}_1$ cannot be better than $\mathcal{Z}_2$.

Example 2.6. *If we choose the Pareto optimal front from Figure 2.3 to be the reference set, the unary additive epsilon indicators for the three approximation sets are:* $I^1_{\varepsilon+}(\mathcal{A}_1) = 5.7$, $I^1_{\varepsilon+}(\mathcal{A}_2) = 3.6$, *and* $I^1_{\varepsilon+}(\mathcal{A}_3) = 5.7$. *The approximation set* $\mathcal{A}_2$ *is the best one according to the unary additive epsilon indicator and the chosen reference set. Note that according to the hypervolume indicator,* $\mathcal{A}_1$ *is better than* $\mathcal{A}_2$. *The disagreement between two unary Pareto compliant indicators means that the approximation sets* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are incomparable.*

Besides the additive epsilon indicators, the multiplicative variants also exist. We will not discuss them here. The interested reader is referred to (Zitzler et al., 2003) for more information.

Unary $R2$ indicator I^1_{R2}

Hansen and Jaszkiewicz (1998) proposed three different binary R indicators (I_{R1}, I_{R2} and I_{R3}). We only use the unary version of I_{R2} in this work, therefore please see (Hansen and Jaszkiewicz, 1998) or (Knowles, 2002) for details on the other indicators.

All R indicators comprise the user's preference in the form of *utility functions:*

$$u : Z \to \mathbb{R}.$$

Suppose that every preference is written as a vector of parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \Lambda$ and that the utility function $u_{\boldsymbol{\lambda}}$ is parameterized with such vectors from Λ. Then, the binary indicator I_{R2} is defined as:

$$I_{R2}(\mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{|\Lambda|} \sum_{\boldsymbol{\lambda} \in \Lambda} \left(\max_{\boldsymbol{z} \in \mathcal{Z}_1} u_{\boldsymbol{\lambda}}(\boldsymbol{z}) - \max_{\boldsymbol{z} \in \mathcal{Z}_2} u_{\boldsymbol{\lambda}}(\boldsymbol{z}) \right).$$

There are several possibilities for the choice of the parameterized utility function $u_{\boldsymbol{\lambda}}$. In this work we use the augmented Chebyshev function (recommended by Knowles et al. (2006)):

$$u_{\boldsymbol{\lambda}}(\boldsymbol{z}) = - \left(\max_{j=1,\ldots,m} \lambda_j |z_j^* - z_j| + \rho \prod_{j=1}^{m} |z_j^* - z_j| \right),$$

where $\boldsymbol{z}^*$ is an ideal reference point (one that weakly dominates all other points) and ρ is a sufficiently small positive real number. The set Λ of vectors of parameters should contain a sufficiently large number of uniformly dispersed vectors, which need to be normalized with respect to the ideal point and the Nadir point (one that is weakly dominated by all other points).

The unary indicator I^1_{R2} is derived from the binary: $I^1_{R2}(\mathcal{Z}) = I_{R2}(\mathcal{Z}, \mathcal{R})$, where $\mathcal{R}$ is the reference set. The smaller the value of the I^1_{R2} indicator, the better the approximation set. If $\mathcal{Z}_1 \triangleleft \mathcal{Z}_2$, then $I^1_{R2}(\mathcal{Z}_1) \leq I^1_{R2}(\mathcal{Z}_2)$. Consequently, if $I^1_{R2}(\mathcal{Z}_1) > I^1_{R2}(\mathcal{Z}_2)$, then $\mathcal{Z}_1$ cannot be better than $\mathcal{Z}_2$.

Example 2.7. *Let us use the I^1_{R2} indicator for evaluating approximation sets $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ from Figure 2.4. If we use the recommended parameters: Pareto front as the reference set, the origin* $(0,0)$ *as the ideal point, the point* $\mathbf{r}$ *as the Nadir point,* 501 *uniformly dispersed parameter vectors and* $\rho = 0.01$*, we get the following indicator values:* $I^1_{R2}(\mathcal{A}_1) = 0.134$*,* $I^1_{R2}(\mathcal{A}_2) = 0.120$*, and* $I^1_{R2}(\mathcal{A}_3) = 0.186$*. This means that according to the I^1_{R2} indicator (and assuming all the presumptions), $\mathcal{A}_2$ is the best approximation set.*

Performance assessment with quality indicators

As already pointed out, in case of comparing the performance of two stochastic multiobjective optimizers on a given MOP, several runs for each optimizer have to be made. After r runs,

we get two sets of approximation sets: $\mathcal{A}_1,\ldots,\mathcal{A}_r$ for the first optimizer and $\mathcal{B}_1,\ldots,\mathcal{B}_r$ for the second one. Afterwards, a Pareto compliant quality indicator is calculated for each approximation set thus yielding values $I^1(\mathcal{A}_1),\ldots,I^1(\mathcal{A}_r)$ and $I^1(\mathcal{B}_1),\ldots,I^1(\mathcal{B}_r)$. It is important to note that all parameters of quality indicators, such as the reference set and the reference point, have to be equal for all evaluated approximation sets. In many cases, it is also reasonable to normalize the approximation sets before the use of indicators. Finally, one of the statistical tests can be used for making inferences about the performance of the two optimizers with regard to the chosen quality indicator.

2.2.3 Empirical attainment function

The third approach to performance assessment is based on the multiobjective concept of *goal-attainment:* an objective vector is attained when it is weakly dominated by the approximation set returned by the optimizer. If the optimizer is run r times, each objective vector can be attained between 0 and r times. The *empirical attainment function* of the objective vector $\boldsymbol{z}$ gives the frequency with which $\boldsymbol{z}$ was attained by the approximation sets $\mathcal{Z}_1,\ldots,\mathcal{Z}_r$ (Grunert da Fonseca et al., 2001):

$$\alpha_r(\boldsymbol{z}) = \frac{1}{r}\sum_{i=1}^{r} \boldsymbol{I}\left(\mathcal{Z}_i \preceq \{\boldsymbol{z}\}\right),$$

where χ is the characteristic function, defined as:

$$\chi(b) = \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Two multiobjective optimizers can be compared on a MOP by performing a statistical test on the results of the corresponding empirical attainment functions. Additionally, this function can be used for visualization of multiple runs of one (or more) optimizers in the following way. Suppose we wish to visualize the objective vectors that have been attained in k% of the runs. The k%-*attainment surface* of the approximation sets $\mathcal{Z}_1,\ldots,\mathcal{Z}_r$ consists of the tightest objective vectors that have been attained in at least k% of the runs:

$$\mathcal{S}_r^{k\%} = \{\boldsymbol{z} \in Z \mid \alpha_r(\boldsymbol{z}) \geq k/100 \wedge \neg(\mathcal{Z}_i \prec\prec \{\boldsymbol{z}\} \text{ for all } i \in 1,\ldots,r)\}.$$

Thus, the k%-attainment surface $\mathcal{S}_r^{k\%}$ divides the objective space into two parts: one, where the objective vectors have been attained in at least k% of the runs, and the other one, where the objective vectors were attained in less than k% of the runs.

Example 2.8. *Suppose that the tree approximation sets $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ from Example 2.3 represent three different runs of a multiobjective optimizer. Figure 2.6 shows all attainment surfaces for these runs: $\mathcal{S}_3^k$ for $k = 1/3, 2/3$ and $3/3$.*

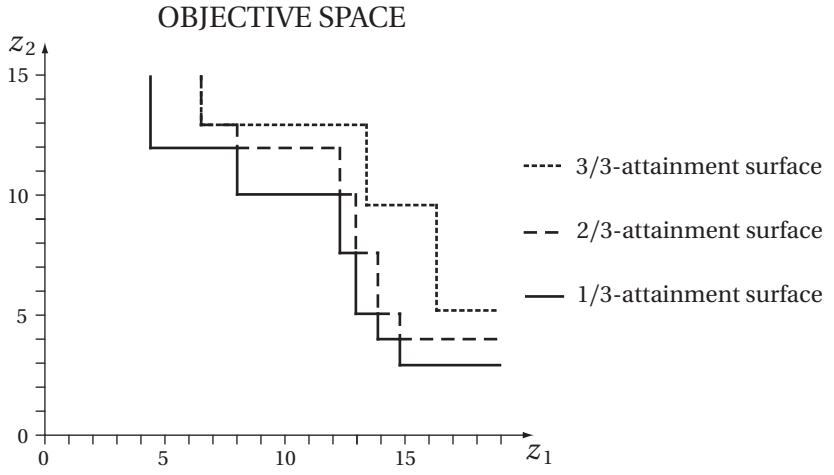


Figure 2.6: Attainment surfaces for the three approximation sets form Figure 2.3.

2.2.4 Multiple testing issues

Several approaches to performance assessment can be simultaneously used for comparing two optimizers on a MOP. In this case, the same data (approximation sets) is used for multiple testing. As this is in contradiction with some assumptions of statistical testing procedures, the significance levels of such tests are not reliable.

One possibility to overcome this problem is to use the *Bonferroni correction* (Bonferroni, 1936) for reducing the significance levels of the statistical tests. For example, if we want to compare two sets of approximation sets with n different quality indicators and we wish to consider an overall significance level α, then we must set the significance level for each comparison to $\alpha_s = \alpha/n$.

2.3 Multiobjective evolutionary algorithms

Evolutionary Algorithms (EAs) are search methods that imitate the principles of the Darwinian theory of evolution. By applying selection, crossover and mutation to a population

of individuals[1], they create better and better offspring individuals. EAs are known to be robust and able to handle all types of functions. Therefore, they have been successfully used in singleobjective optimization.

Since the ideal principle requires elements of the approximation set to be available simultaneously, population-based search methods are the most appropriate for this task. This is why in the last twenty years EAs have been frequently used also as multiobjective optimizers (Coello Coello et al., 2002; Deb, 2001). Some of the most popular Multiobjective Evolutionary Algorithms (MOEAs) are based on Genetic Algorithms (GAs). In the following we review three such algorithms, namely NSGA-II, SPEA2 and IBEA, using a unifying framework, which we call the *basic genetic algorithm*.

2.3.1 Basic genetic algorithm

The outline of the basic GA is presented in Algorithm 2.1. Unlike GAs for singleobjective optimization, the basic GA uses two populations. Population $\mathcal{Q}$ contains the current individuals, while population $\mathcal{P}$ preserves the best individuals found so far.

Basic Genetic Algorithm

Input: Parameters of the algorithm.
1. Initialize populations $\mathcal{P}_0$ and $\mathcal{Q}_0$.
2. Set $t = 0$.
3. Repeat:
   3.1. Set $t = t + 1$.
   3.2. Calculate the objectives for new individuals from $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$.
   3.3. Get $\mathcal{P}_t$ from $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$ with an environmental selection procedure.
   3.4. If stopping criterion met, exit the loop.
   3.5. Fill the mating pool $\mathcal{M}_t$ using tournament selection on $\mathcal{P}_t$.
   3.6. Apply variation to individuals from $\mathcal{M}_t$ to get $\mathcal{Q}_t$ (see Algorithm 2.2).
Output: Nondominated individuals from $\mathcal{P}_t$.

Algorithm 2.1: Outline of the basic GA.

After initialization of the first populations $\mathcal{P}_0$ and $\mathcal{Q}_0$, the main loop is executed for several generations until a stopping criterion is met. At each generation t, we first calculate the objective function for the new individuals from populations $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$. Because all populations have a fixed (and usually equal) size, an environmental selection procedure is ap-

[1]Note that in the field of evolutionary computation, the terms *individual* and *population* are often used instead of solution and set of solutions, respectively.

plied for selecting the best individuals that enter the next population. If only the Pareto dominance concept is used, many individuals are incomparable. Therefore, the environmental selection procedure ranks individuals from both populations according to some preference. Usually this preference comprises information on domination between pairs of individuals and their spread in the objective space. Only the better half of individuals according to this ranking enters the population $\mathcal{P}_t$. Finally, the evolutionary steps of mating selection, crossover and mutation are applied to obtain a new offspring population $\mathcal{Q}_t$ from the parent population $\mathcal{P}_t$.

In the case of numeric MOPs, the individuals are usually encoded as real-valued vectors. In such cases the variation consists of uniform and simulated binary crossover and polynomial mutation (Deb and Agrawal, 1995), as shown in Algorithm 2.2. When tackling combinatorial MOPs, different encoding and variation operators need to be used.

Variation procedure

Input: Mating pool $\mathcal{M}_t$.
1. Create empty population $\mathcal{Q}_t$.
2. For each pair of individuals $\boldsymbol{x}^i$, $\boldsymbol{x}^{i+1}$ ($i = 1,3,\dots$) from $\mathcal{M}_t$ do:
   2.1. Modify the individuals $\boldsymbol{x}^i$, $\boldsymbol{x}^{i+1}$ with uniform crossover.
   2.2. Modify the individuals $\boldsymbol{x}^i$, $\boldsymbol{x}^{i+1}$ with simulated binary crossover.
   2.3. Modify the individual $\boldsymbol{x}^i$ with polynomial mutation.
   2.4. Modify the individual $\boldsymbol{x}^{i+1}$ with polynomial mutation.
   2.5. Add individuals $\boldsymbol{x}^i$ and $\boldsymbol{x}^{i+1}$ to $\mathcal{Q}_t$.

Output: Population $\mathcal{Q}_t$.

Algorithm 2.2: Outline of variation procedure in the basic GA for individuals encoded as real-valued vectors.

The popular algorithms NSGA-II, SPEA2 and IBEA, which will be soon presented in more detail, can be viewed as special cases of the basic GA. The main differences among them lie in the approach used for environmental selection.

2.3.2 Nondominated sorting

The Nondominated Sorting Genetic Algorithm II (NSGA-II) by Deb et al. (2002) is probably the most widely-used MOEA and was applied to many real-world problems. NSGA-II initializes the first parent population $\mathcal{P}_0$ with randomly created individuals and sets the first offspring population $\mathcal{Q}_0$ to be empty. Its environmental selection procedure is based on *nondominated sorting* and the *crowding distance* metric. At generation *t*, the individuals

from populations $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$ are joined and ranked according to the number of individuals that dominate them. All nondominated individuals are allocated into the first front and nondominated sorting is applied again to the remaining individuals. In this way, we get a sequence of fronts, where individuals from precedent fronts are preferred to those from subsequent fronts. The new population $\mathcal{P}_t$ is filled in turn with the individuals from the best fronts. If a front cannot fit into $\mathcal{P}_t$ entirely, the individuals from this front are further ranked according to the crowding distance metric.

Sorting based on crowding distance prefers individuals from less crowded regions of the objective space. For the individual $\boldsymbol{x}^i$, the distance $d_j(\boldsymbol{x}^i)$ between its neighboring individuals $\boldsymbol{x}^{i-}$ and $\boldsymbol{x}^{i+}$ in the objective j is calculated as:

$$d_j(\boldsymbol{x}^i) = \frac{f_j(\boldsymbol{x}^{i+}) - f_j(\boldsymbol{x}^{i-})}{f_j^{\max} - f_j^{\min}},$$

where $f_j^{\max}$ and $f_j^{\min}$ denote the maximum and minimum value of the objective j. The two solutions with extreme values of f_j are assigned the biggest possible d_j. The crowding distance for the individual $\boldsymbol{x}^i$ is then defined as:

$$c(\boldsymbol{x}^i) = \sum_{j=1}^{m} d_j(\boldsymbol{x}^i).$$

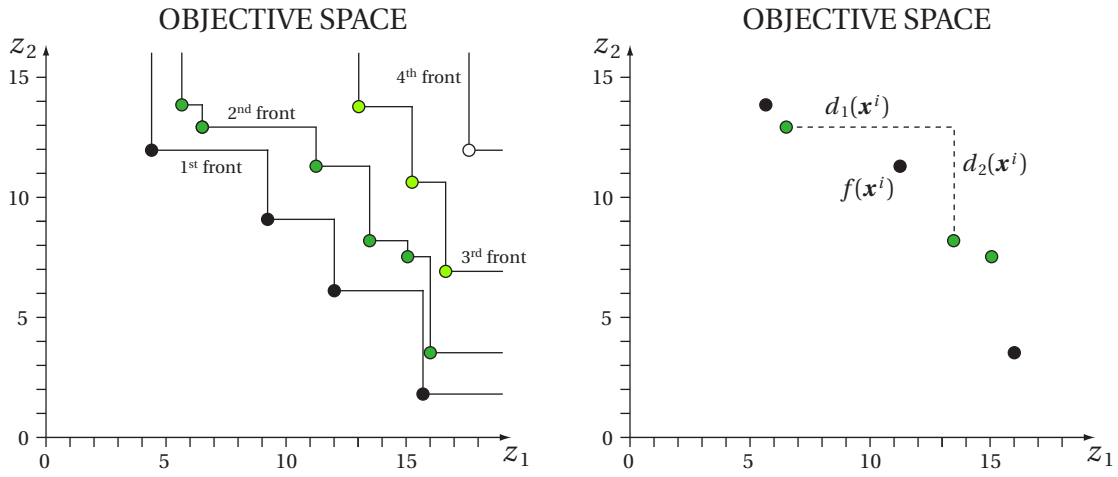
The individuals with the biggest crowding distance are included in the next population. The environmental selection procedure of NSGA-II is illustrated in the next example.

Example 2.9. *Consider the two joined populations $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$ presented in Figure 2.7 (a). After nondominated sorting, only the first front enters the next population entirely. From the second front, three best individuals must be chosen. Figure 2.7 (b) shows the calculation of the crowding distance metric. The individuals represented as black points are the best ones according to this metric and therefore enter the next population.*

The overall worst-case complexity of NSGA-II is $\mathcal{O}(mN^2)$, where m denotes the number of objectives and N is the number of individuals in the joined population $\mathcal{P}_{t-1} \cup \mathcal{Q}_{t-1}$. The complexity is due to nondominated sorting, which requires at most $\mathcal{O}(mN^2)$ operations, while crowding distance calculation and sorting by crowding distance require $\mathcal{O}(mN\log N)$ and $\mathcal{O}(N\log N)$ operations in the worst case, respectively.

2.3.3 Strength Pareto approach

Besides NSGA-II, the Strength Pareto Evolutionary Algorithm 2 (SPEA2) by Zitzler et al. (2001) is a popular genetic algorithm, which performs comparably to NSGA-II regarding conver-



(a) nondominated sorting (b) crowding distance

Figure 2.7: Environmental selection procedure in NSGA-II.

gence to the Pareto optimal front and often achieves a more uniform spread of individuals than NSGA-II.

Instead of a parent and offspring population, SPEA2 uses a basic population $\mathcal{Q}$, which is initialized with random individuals, and an archive of best individuals $\mathcal{P}$, which is initially empty (note that this is just the opposite of the initialization in NSGA-II). Evaluation of individuals is done by measuring their *strength*, *raw fitness* and *density*. The strength of an individual at generation t is equal to the number of individuals from $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$ that are dominated by it (see Figure 2.8 (a)):

$$S(\boldsymbol{x}^i) = \left| \left\{ \boldsymbol{x}^j \in \mathcal{P}_{t-1} \cup \mathcal{Q}_{t-1} \mid \boldsymbol{x}^i \prec \boldsymbol{x}^j \right\} \right|.$$

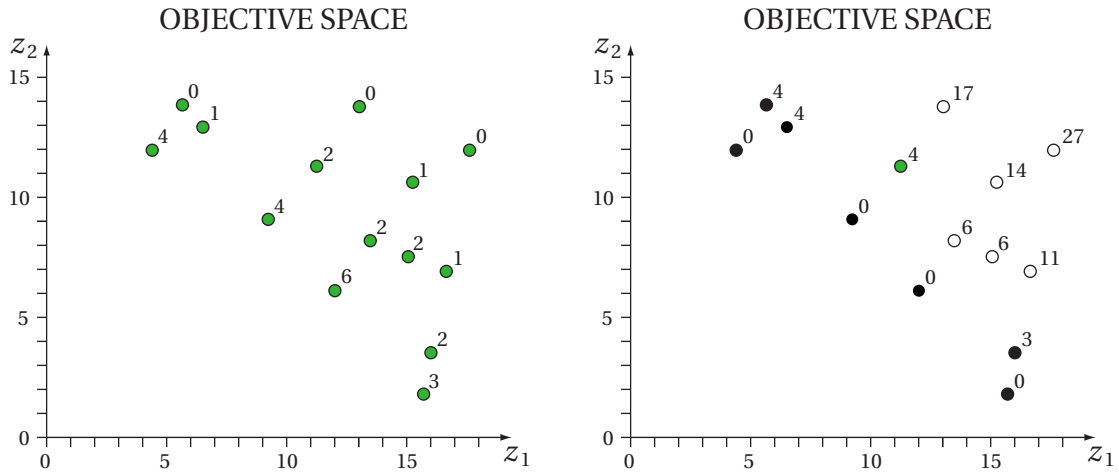
The raw fitness of an individual is computed by summing the strengths of all individuals that dominate it (see Figure 2.8 (b)):

$$R(\boldsymbol{x}^i) = \sum_{\boldsymbol{x}^j \prec \boldsymbol{x}^i} S(\boldsymbol{x}^j).$$

Because with evolution the majority of individuals become nondominated (and have raw fitness equal to 0), the ties are broken using additional information on their spread in the objective space. For all individuals $\boldsymbol{x}^i$ with the same raw fitness $R(\boldsymbol{x}^i)$, the density is calculated as:

$$D(\boldsymbol{x}^i) = \frac{1}{\sigma_i^k + 2},$$

where σ_i^k is the distance to the k-nearest neighbor of $\boldsymbol{x}^i$ (k is a parameter dependent on population and archive size and is usually set to $\sqrt{|\mathcal{P}| + |\mathcal{Q}|}$). The true fitness of the individual is



(a) strength of individuals (b) raw fitness of individuals

Figure 2.8: Environmental selection procedure in SPEA2.

finally calculated by summing the raw fitness and density and should be minimized:

$$F(\boldsymbol{x}^i) = R(\boldsymbol{x}^i) + D(\boldsymbol{x}^i).$$

Example 2.10. *In the example from Figure 2.8, we must choose among the three individuals with raw fitness equal to* 4. *If* $k = 3$, *the individuals represented as black points are chosen for the next population.*

The strength Pareto approach is computationally more expensive than nondominated sorting. Since computing raw fitness of individuals requires at most $\mathcal{O}(mN^2)$ operations and density $\mathcal{O}(mN^2 \log N)$ operations, the overall worst-case complexity of SPEA2 amounts to $\mathcal{O}(mN^2 \log N)$ operations. Again, m denotes the number of objectives and N is the number of individuals in the joined population $\mathcal{P}_{t-1} \cup \mathcal{Q}_{t-1}$.

2.3.4 Indicator-based selection

The Indicator-Based Evolutionary Algorithm (IBEA) by Zitzler and Künzli (2004) is the most recent of the presented algorithms and uses a different approach to environmental selection than NSGA-II or SPEA. Individuals are evaluated according to the preference information of the decision maker, which is given in the form of a Pareto compliant binary quality indicator (see Subsection 2.2.2), and no other explicit diversity preserving mechanism is needed.

IBEA originally uses a single population of variable size instead of two separate populations. Without altering its performance, we can assume that IBEA uses two populations, which are initialized in the same way as in NSGA-II.

Before environmental selection at each generation t, the objective values of all individuals must be normalized to the $[0,1]$ interval. Environmental selection procedure in IBEA ranks the individuals according to their usefulness regarding the chosen quality indicator. For example, an individual $\boldsymbol{x}^i$ from the joined population $\mathcal{R}_{t-1} = \mathcal{P}_{t-1} \cup \mathcal{Q}_{t-1}$ could be evaluated by simply summing up its indicator values with respect to the rest of population:

$$F'(\boldsymbol{x}^i) = \sum_{\boldsymbol{x}^j \in \mathcal{R}_{t-1} \setminus \{\boldsymbol{x}^i\}} I(\{\boldsymbol{x}^j\}, \{\boldsymbol{x}^i\}).$$

The fitness value F', which is to be maximized, is a measure for the loss in quality if $\boldsymbol{x}^i$ is removed from the population. For the $I_{\varepsilon+}$ indicator (see Subsection 2.2.2), the $F'(\boldsymbol{x}^i)$ divided by the population size equals the average ε needed to cover $\boldsymbol{x}^i$ by other population members. However, IBEA uses a slightly different schema, which amplifies the influence of dominating population members over dominated ones:

$$F(\boldsymbol{x}^i) = \sum_{\boldsymbol{x}^j \in \mathcal{R}_{t-1} \setminus \{\boldsymbol{x}^i\}} -e^{-I(\{\boldsymbol{x}^j\}, \{\boldsymbol{x}^i\})/(c\kappa)},$$

where κ is a positive scaling factor and c is the maximum absolute value of I on individuals from $\mathcal{R}_{t-1}$. The fitness defined in this way should be minimized.

The environmental selection procedure consists of repeating the following two steps: (1) the individual $\boldsymbol{x}^*$ with the smallest fitness is removed from $\mathcal{R}_{t-1}$ and added to $\mathcal{P}_t$, (2) the fitness values of the remaining individuals $\boldsymbol{x}$ are updated with:

$$F(\boldsymbol{x}) = F(\boldsymbol{x}) + e^{-I(\{\boldsymbol{x}^*\}, \{\boldsymbol{x}\})/(c\kappa)}.$$

The selection stops, when the population $\mathcal{P}_t$ is full.

While arbitrary Pareto compliant binary indicators can be used with IBEA, the algorithm was presented in combination with the binary additive epsilon indicator $I_{\varepsilon+}$ and the binary HD indicator, defined as:

$$I_{\mathrm{HD}}(\mathcal{Z}_1, \mathcal{Z}_2) = \begin{cases} I_{\mathrm{H}}(\mathcal{Z}_2) - I_{\mathrm{H}}(\mathcal{Z}_1) & \text{if } \mathcal{Z}_1 \prec \mathcal{Z}_2, \\ I_{\mathrm{H}}(\mathcal{Z}_1 \cup \mathcal{Z}_2) - I_{\mathrm{H}}(\mathcal{Z}_1) & \text{otherwise,} \end{cases}$$

where I_{H} is the hypervolume indicator. In the reminder of the thesis we use $\mathrm{IBEA}_{\varepsilon+}$ and $\mathrm{IBEA}_{\mathrm{HD}}$ to denote the two versions of IBEA that use the corresponding quality indicators.

Let m denote the number of objectives and N the number of individuals in the joined population $\mathcal{R}_{t-1}$. If the computation of the indicator of two solutions requires $\mathcal{O}(m)$ operations (as is the case for $I_{\varepsilon+}$ and I_{HD} indicators), the overall worst-case complexity of IBEA is $\mathcal{O}(mN^2)$, which is equal to that of NSGA-II.

Chapter 3

Differential Evolution for Multiobjective Optimization

This chapter presents the main contribution of this thesis, namely the algorithm Differential Evolution for Multiobjective Optimization (DEMO). As its name suggests, DEMO is based on Differential Evolution (DE)—a successful evolutionary algorithm for singleobjective optimization introduced in Section 3.1. The first adaptations of DE for solving multiobjective optimization problems are listed in Section 3.2. After that, the DEMO algorithm is presented in detail in Section 3.3, while Section 3.4 reports on the recent adaptations of DE to multiobjective optimization. The chapter concludes with two empirical studies: Section 3.5 describes the experiments and results of the comparison between DEMO and the basic GA, while Section 3.6 contains the results of the comparison among different DEMO variants.

3.1 Differential evolution

Differential evolution, contrived by Storn and Price (1997), is a simple population-based algorithm that encodes solutions as vectors and uses operations such as vector addition, scalar multiplication and exchange of components (crossover) to construct new solutions from the existing ones.

3.1.1 Algorithm outline

DE, as evolutionary algorithms in general, starts with a population of random solutions, from which better and better solutions are obtained at each generation. For each parent solution, a so-called *candidate* is constructed using one of the many possible strategies. After that, the candidate is evaluated and compared to its parent. If the candidate is better than or equal to its parent, it replaces the parent in the population. Otherwise, the candidate is discarded. This procedure is repeated in turn for each parent solution from population $\mathcal{P}$. After that, all solutions are randomly enumerated so that the order of parents changes. When the stopping criterion is met, the best solution found is returned. The DE algorithm is shown in Algorithm 3.1.

Differential Evolution

Input: Parameters of the algorithm.
1. Evaluate the initial population $\mathcal{P}$ of random solutions.
2. While stopping criterion not met, do:
   2.1. For each solution $\boldsymbol{x}_i$ $(i = 1,\ldots,|\mathcal{P}|)$ from $\mathcal{P}$ repeat:
      (a) Create candidate $\boldsymbol{c}$ from parent $\boldsymbol{x}_i$ (see Algorithm 3.2).
      (b) Evaluate the candidate.
      (c) If the candidate is better than or equal to the parent, the candidate replaces the parent. Otherwise, the candidate is discarded.
   2.2. Randomly enumerate the solutions in $\mathcal{P}$.
Output: The best solution from $\mathcal{P}$.

Algorithm 3.1: Outline of DE.

3.1.2 The DE/rand/1/bin strategy

The candidate is created using one of the so-called DE strategies. In this work, we use the DE/rand/1/bin strategy (see Algorithm 3.2 and Figure 3.1) as this is one of the most frequently used DE strategies, also referred to as *classic DE* in (Price et al., 2005). From the current population and the current parent $\boldsymbol{x}^i$, the candidate $\boldsymbol{c}$ is constructed with the help of three randomly chosen solutions $\boldsymbol{x}^{i_1}, \boldsymbol{x}^{i_2}$ and $\boldsymbol{x}^{i_3}$:

$$\boldsymbol{c} = \boldsymbol{x}^{i_1} + F(\boldsymbol{x}^{i_2} - \boldsymbol{x}^{i_3}),$$

where i, i_1, i_2 and i_3 are pairwise different and F is a scaling factor for the difference vector $\boldsymbol{x}^{i_2} - \boldsymbol{x}^{i_3}$. This step is often referred to as *mutation*. Afterwards, the candidate is subject to

binomial crossover with the parent, where some components of the parent are copied to the corresponding components in the candidate. Let l be a random integer from $\{1,\ldots,n\}$. After binomial crossover, each candidate's component c_k is equal to:

$$c_k = \begin{cases} c_k & \text{if } k = l \text{ or } r_k \leq CR, \\ x_k^i & \text{otherwise,} \end{cases}$$

where each r_k is chosen randomly from the $[0,1]$ interval. Note that the CR probability used here has a different meaning than the crossover probability usually used by GAs. In binomial crossover, the smaller the CR, the greater the influence of the parent. But even if $CR = 0$, the candidate does not inherit all components from its parent, since the component c_l always remains intact for a randomly chosen l.

Candidate creation

Input: Population $\mathcal{P}$ and the chosen parent $\boldsymbol{x}^i$.
1. Randomly select three solutions $\boldsymbol{x}^{i_1}, \boldsymbol{x}^{i_2}$ and $\boldsymbol{x}^{i_3}$ from $\mathcal{P}$, where i, i_1, i_2 and i_3 are pairwise different.
2. Calculate candidate $\boldsymbol{c}$ as $\boldsymbol{c} = \boldsymbol{x}^{i_1} + F(\boldsymbol{x}^{i_2} - \boldsymbol{x}^{i_3})$, where F is a scaling factor.
3. Modify the candidate by binomial crossover with the parent $\boldsymbol{x}^i$ using probability CR.
4. Repair the candidate if necessary.

Output: Candidate $\boldsymbol{c}$.

Algorithm 3.2: Outline of candidate creation with the DE/rand/1/bin strategy.

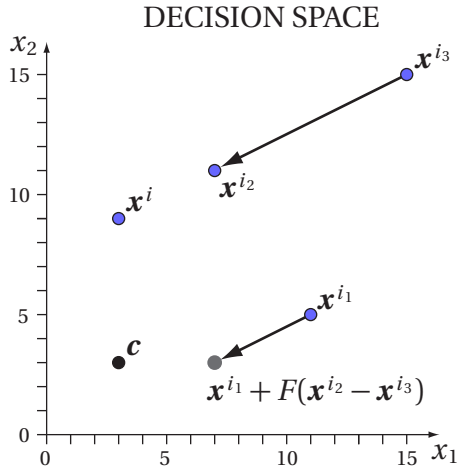


Figure 3.1: Visual representation of candidate creation with DE/rand/1/bin strategy.

Sometimes, the newly created candidate falls out of bounds of the decision space. In such cases, many repair methods can be used. We address this problem by replacing the candidate value violating the boundary constraints with the closest boundary value. In this way, the candidate becomes feasible with as few alterations to it as possible and there is no need for making a new candidate. It is important to note, however, that this repair method may yield more boundary solutions and is biased for problems where the optimal solution lies on one of the bounds of the decision space.

A repair method is needed also in the case when the decision variables are not continuous. For example, if the decision space is discretized, the operations of vector addition and scalar multiplication can result in a point that is not part of the discretized decision space. In such cases, two possible repair schemes can be adopted. According to the *Lamarckian repair*, the candidate's values must be rounded to that of the nearest point in the decision space. The *Baldwinian repair*, on the other hand, rounds the candidate's values only to enable the correct calculation of its objectives, while actually leaving the original values unrepaired. See (Ishibuchi et al., 2005) for a comparison between the two repair schemes on the multiobjective 0/1 knapsack problem.

The basic idea behind DE is that differences between solutions are used as search directions. If the differences lead to new solutions, which are good (better than the existing parent solution used for comparison), then this reinforces the search in this promising direction. Otherwise, the newly obtained (bad) solutions are discarded and the unpromising direction is not fortified. Through evolution in DE, the solutions converge closer together and the differences among solutions are reduced. This serves as an automatic adaptation of the search procedure: at the beginning, the solutions are more diverse and there is more focus on the exploration of the decision space, while in the end, the solutions are closer together and the method concentrates on the exploitation of the decision space.

All DE strategies are written using the DE/x/y/z notation, where x represents the method of selection of the first solution $\boldsymbol{x}^{i_1}$, which can be selected randomly (rand) or as the best vector so far (best); y is the number of difference vectors used; and z defines the type of crossover which can be binomial (bin) or exponential (exp) (Price et al., 2005). Besides the DE/rand/1/bin strategy, the following strategies can also be frequently found in the literature: DE/best/1/bin, DE/best/2/bin, DE/rand/2/exp and similar. For more information on these and other DE strategies, please see (Price et al., 2005) and (Mezura-Montes et al., 2006).

3.1.3 Advantages, limitations and applications

Since no solution can be removed from the population unless a better solution is found, DE implicitly incorporates elitism—a desirable property of evolutionary algorithms that assures the preservation of good solutions. Moreover, the newly obtained good solutions immediately participate in the creation of new candidates, which speeds up the convergence to the optimal solutions. But perhaps the key advantage of DE is its simplicity: it is easy to understand, implement and use.

The biggest disadvantage of DE originates in the limitations of its encoding. Because of operations of vector addition and scalar multiplication, the decision space needs to be a vector space. This does not necessarily mean that the solutions have to be encoded as real vectors, but they need to be encoded in such way, that the addition of two solutions and scalar multiplication have a sensible definition. As no such vector representation of solution exists for combinatorial problems, the DE's principle can only be applied in numerical optimization.

DE has been successfully applied to many real-world optimization problems, such as design of aeronautic shapes (Rogalsky et al., 1999), neural network learning (Ilonen et al., 2003), parameter identification of induction motors (Ursem and Vadstrup, 2003), flexible ligand docking in bioinformatics (Thomsen, 2003) and others. While it is a very efficient and reliable optimizer of non-noisy functions, experiments show that its performance deteriorates on noisy functions, when the fitness of solutions approaches the fitness variance caused by the noise (Krink et al., 2004).

3.2 First adaptations to multiobjective optimization

After the successful application of GAs to MOPs, the first multiobjective optimizers based on DE were proposed. The greatest challenge in adapting DE to multiobjective optimization is the comparison between the parent solution and the newly crated candidate. Researchers have confronted with this challenge in different ways.

Abbass et al. (2001) introduced the Pareto-frontier Differential Evolution (PDE), in which only the nondominated solutions can take part in the reproduction process. The dominance relation is used to determine if the candidate can replace its parent, i.e., only the candidates that dominate their parents are allowed to take their place, while all the other candidates are discarded. PDE was compared to SPEA (Zitzler and Thiele, 1999) on two test problems and

found to outperform it.

In a short technical report, Lampinen (2001) defined a DE's selection rule which was referred to as Generalized Differential Evolution (GDE) in subsequent publications. GDE is designed for multiobjective problems with constraints where some solutions can be infeasible. When the candidate solution and its parent are both feasible, the candidate replaces the parent if it weakly dominates it. This rule is very similar to the selection in PDE. The only difference is that instead of the dominance relation, the weak dominance relation is used.

The following year, Madavan (2002) proposed the Pareto Differential Evolution Approach (PDEA[1]). Like PDE, PDEA uses DE for creating new solutions. It then merges the parent and offspring populations and calculates the nondominated rank (with Pareto-based ranking assignment) and diversity rank (with the crowding distance metric) for each solution. Two variants of PDEA were investigated. The first compares each candidate with its parent. The candidate replaces the parent if it has a higher nondominated rank or, if it has the same nondominated rank and a higher diversity rank. Otherwise the candidate is discarded. This variant was found inefficient—the diversity was good, but the convergence slow. The other variant simply takes the best solutions according to the nondominated rank and diversity rank (like in NSGA-II). This variant has proved to be very efficient and was applied to several MOPs where it produced favorable results.

Xue et al. (2003) presented Multiobjective Differential Evolution (MODE). This algorithm also uses the Pareto-based ranking assignment and the crowding distance metric, but in a different manner than PDEA. In MODE, the fitness of a solution is first calculated using Pareto-based ranking and then reduced with respect to the solution's crowding distance value. This single fitness value is then used to select the best solutions for the new population. MODE was tested on five benchmark problems where it produced better results than SPEA.

Parsopoulos et al. (2004) took a different approach with the Vector Evaluated Differential Evolution (VEDE) inspired by the Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1984). VEDE is a multi-population algorithm, where M populations are evolved simultaneously (while M can be greater than the number of objectives, we assume that M equals the number of objectives for the purpose of this presentation). DE is used for constructing new candidates in each of the M populations and the Pareto dominance criterion (like in PDE) is used to determine if the candidates are to replace their parents. In the end of each gener-

[1]This acronym was not used in (Madavan, 2002). We introduce it to make clear distinction between his approach and other implementations of DE for multiobjective optimization.

ation, the 'best' solution from population i is sent into population $i+1$ (the 'best' solution from the last population is sent to the first population), where it is later used for creating new solutions in the next generation. The 'best' solution from population i is defined as the solution with the lowest value of the objective f_i. In the mentioned study, VEDE outperformed VEGA on all four tested multiobjective problems.

Iorio and Li (2004) used Non-dominated Sorting Differential Evolution (NSDE), an approach identical to PDEA, to solve rotated multiobjective optimization problems. For a presented rotated problem, NSDE achieved significantly better results than NSGA-II.

Three years after the first introduction of GDE, Kukkonen and Lampinen (2004) presented the Extension of Generalized Differential Evolution, also called GDE2. Let us again restrict to the case when the candidate and its parent are both feasible. GDE2 chooses the candidate over its parent when the candidate weakly dominates the parent (as in GDE) or when they do not dominate each other and the parent resides in a more crowded region than the candidate. GDE2 was compared to SPEA, NSGA-II and GDE on five test problems, where it achieved better results than SPEA and comparable performance to NSGA-II and GDE.

3.3 The DEMO algorithm

The idea presented in this work is to use differential evolution for exploring the decision space and environmental selection mechanisms from NSGA-II, SPEA2, IBEA (see their description in Section 2.3) or some other approach to select the best individuals for the next population. This idea is implemented in the DEMO algorithm[2], presented in this section.

The outline of DEMO is shown in Algorithm 3.3. Like DE, the algorithm starts with a population $\mathcal{P}$ of p randomly created solutions. At each generation, the following steps are repeated. A candidate is constructed from its parent (and other solutions from $\mathcal{P}$) using the DE/rand/1/bin strategy described earlier (see Subsection 3.1.2). After that, the candidate is evaluated and compared to its parent. At this point, DEMO differs from DE (see step 2.1 (c) in Algorithms 3.1 and 3.3). In DEMO, the candidate replaces the parent only if it dominates it. If the parent dominates the candidate, the candidate is discarded. Otherwise (when the candidate and parent are incomparable), the candidate is added to the population. After repeating this step p times, we get a population of size between p and $2p$. If the population

[2]DEMO is a generalization of the DEMO/parent variant presented in (Robič and Filipič, 2005), which used the DE/rand/1/bin strategy for candidate creation and environmental selection procedure as in NSGA-II.

has enlarged, it must be truncated to size p using one of the approaches to environmental selection. In the end of generation, the solutions from $\mathcal{P}$ are randomly enumerated (as in DE). The final output of DEMO consists of nondominated solutions from $\mathcal{P}$.

Differential Evolution for Multiobjective Optimization

Input: Parameters of the algorithm.
1. Evaluate the initial population $\mathcal{P}$ of p random solutions.
2. While stopping criterion not met, do:
   2.1. For each solution $\boldsymbol{x}^i$ $(i = 1, \ldots, p)$ from $\mathcal{P}$ repeat:
      (a) Create candidate $\boldsymbol{c}$ from parent $\boldsymbol{x}^i$ (see Algorithm 3.2).
      (b) Calculate the objectives of the candidate.
      (c) If the candidate dominates the parent, the candidate replaces the parent.
          If the parent dominates the candidate, the candidate is discarded.
          Otherwise, the candidate is added to the population.
   2.2. If the population has more than p solutions, apply an environmental selection procedure to get the best p solutions.
   2.3. Randomly enumerate the solutions in $\mathcal{P}$.
Output: Nondominated solutions from $\mathcal{P}$.

Algorithm 3.3: Outline of DEMO.

Note that the newly created candidates that enter the population (either by replacement or by addition) instantly take part in the creation of subsequent candidates. This helps achieving fast convergence to the Pareto optimal front. Moreover, it resembles very closely the steady-state mechanism of DE—on the contrary to the related PDEA algorithm described earlier.

Besides the environmental selection mechanism from NSGA-II, which has been often used in previous DE-based algorithms, DEMO can incorporate an arbitrary environmental selection procedure. In the remainder of this thesis we will use the notations DEMO$^{\text{NS-II}}$, DEMO$^{\text{SP2}}$, DEMO$^{\text{IB}_{\varepsilon+}}$ and DEMO$^{\text{IB}_{\text{HD}}}$ for DEMO variants that use environmental selection procedures from algorithms NSGA-II, SPEA2, IBEA$_{\varepsilon+}$ and IBEA$_{\text{HD}}$, respectively.

Because DEMO immediately discards the dominated solution in the comparison between the candidate and its parent, its population size rarely reaches $2p$ before environmental selection. Therefore, the computational complexity of the employed environmental selection procedure is often smaller than the same environmental selection procedure applied on the simple GA.

3.4 Recent adaptations to multiobjective optimization

Soon after the first proposal of DEMO (Robič and Filipič, 2005), Kukkonen and Lampinen (2005) presented the third evolution step of GDE, called GDE3. In the case of unconstrained problems, GDE3 is almost equal to DEMO—the only difference is that DEMO uses Pareto dominance when comparing two solutions, while GDE3 uses weak Pareto dominance. The performance of GDE3 was compared to that of NSGA-II on unconstrained problems with one, two and three objectives as well as on some constrained two-objective problems. GDE achieved better diversity and sometimes better convergence than NSGA-II.

At the same time, Santana-Quintero and Coello Coello (2005) introduced the ε-MyDE algorithm, which deals with constrained and unconstrained optimization problems. In the case when no constraints are violated, the selection procedure is the following. If the candidate dominates its parent, the candidate is chosen and if the parent dominates the candidate, the parent is chosen. Otherwise, the winner is determined randomly (with 50% chance of each outcome). Additionaly, ε-MyDE uses an external archive of elite solutions. When selecting the solutions for the archive, the ε-dominance relation (see its definition on page 14) is used instead of the Pareto dominance relation. In comparison to PDE, NSGA-II and ε-MOEA (Deb et al., 2003), ε-MyDE achieved good results on two test problems.

Two other approaches that use DE for multiobjective optimization were proposed very recently. Iorio and Li (2006) extended their NSDE algorithm by taking into consideration the directional information of solutions in the decision space. Three different algorithms were presented—one that tries to optimize the convergence to the Pareto optimal set (NSDE with Directional Convergence, or NSDE-DC), one that aims at obtaining well-spread solutions (NSDE with Directional Spread, or NSDE-DS) and one that combines both goals (NSDE with Directional Convergence and Spread, or NSDE-DCS). The basic difference between the three algorithms and NSDE (or PDEA) lies in the way they select the solutions used for the calculation of the new candidate. While NSDE selects these solutions randomly, the new variants select the solutions in such a way that the difference vector points in the general direction of the Pareto optimal set (the NSDE-DC variant), along the Pareto optimal set (the NSDE-DS variant) or either (the NSDE-DCS variant). Although such a limitation on the direction of the search could cause the algorithm to get stuck in a local optimum, tests show that the three presented variants (and especially NSDE-DCS) outperform NSGA-II and its predecessor NSDE on all four presented problems.

The second recent approach by Hernández-Díaz et al. (2006) uses rough sets theory to

help DE to converge closer to the Pareto optimal front. The resulting algorithm, called Differential Evolution for Multiobjective Optimization with Rough Sets (DEMORS), consists of two phases. In the first phase, a DE-based approach is used for finding new solutions. Besides the commonly-used population of solutions, two archives of already evaluated solutions are kept. The solutions are arranged in the archives according to the so-called Pareto adaptive ε-dominance relation between them. In the second phase, a local search procedure based on rough sets theory is applied in order to improve the solutions obtained in the first phase. DEMORS is a rather complicated algorithm and introduces some new parameters in addition to the ones used by DE. Nevertheless, it is able to obtain very good solutions using only a small number of evaluations.

3.5 Comparison between DEMO and the basic GA

We wish to compare the performance of DEMO to that of the basic GA when the environmental selection procedure of both algorithms is the same. For this purpose, we compare DEMO$^{\text{NS-II}}$ to NSGA-II, DEMO$^{\text{SP2}}$ to SPEA2, DEMO$^{\text{IB}_{\varepsilon+}}$ to IBEA$_{\varepsilon+}$, and DEMO$^{\text{IB}_{\text{HD}}}$ to IBEA$_{\text{HD}}$ on 16 benchmark problems and assess the results using appropriate performance indicators. A part of this comparison has been covered in (Tušar and Filipič, 2007).

3.5.1 Benchmark problems

Two test problem suites were used in the experiments. The first consists of the first seven DTLZ test problems from (Deb et al., 2005), while the second comprises the nine WFG test problems presented in (Huband et al., 2005). Both suites comprise difficult problems that present many challenges to multiobjective optimizers, such as the existence of many local Pareto optimal solutions, uneven distribution of points in the objective space and different geometries of the Pareto optimal front. Additionally, the WFG test suite incorporates non-separable problems, deceptive problems and problems where the fitness landscape has flat regions (see (Huband et al., 2006) for a detailed study on benchmark problems for multiobjective optimization).

Let n and m denote the dimensionality of the decision and variable space, respectively. Each of the 16 problems was used three times—each time with a different number of objectives ($m = 2$, 3 and 4). The other parameters were set as follows:

- The parameters of DTLZ problems were set as recommended by Deb et al. (2005), i.e.,

$n = m + k - 1$, where $k = 5$ for DTLZ1, $k = 10$ for DTLZ2 to DTLZ6 problems and $k = 20$ for DTLZ7.

- Parameters of the WFG test suite are: the number of position related parameters k, number of distance related parameters l and number of objectives m. The number of decision variables is calculated as $n = k + l$. Because of some additional requirements (l must be an even number for WFG2 and WFG3, and k must be divisible by $m - 1$), we used the following setting: $k = 6$ and $l = 4$ (consequently $n = 10$), which satisfies all the requirements for $m = 2$, 3 and 4.

All test problems require minimization of all objectives.

3.5.2 Parameters of the algorithms

The parameter settings for the basic GA are the same as the ones used in the comparison between NSGA-II and SPEA2 on the DTLZ1 problem in (Deb et al., 2005):

- population size = 100,
- number of generations = 300,
- tournament size = 2,
- size of the mating pool = 100,
- individual crossover probability = 1,
- variable probability of simulated binary crossover = 1,
- distribution index for crossover $\eta_c = 15$,
- variable probability of uniform crossover = 0.5,
- individual mutation probability = 1,
- variable probability of polynomial mutation = $1/n$,
- distribution index for mutation $\eta_m = 20$.

The parameters of all four DEMO variants were set as in (Robič and Filipič, 2005) (except for the number of generations, which equals the number of generations used by the basic GA):

- population size = 100,
- number of generations = 300,
- DE selection scheme = DE/rand/1/bin,

– scaling factor $F = 0.5$,

– probability in binomial crossover $CR = 0.3$.

$\text{DEMO}^{\text{IB}_{\text{HD}}}$ and IBEA_{HD} used additional parameters: scaling factor $\kappa = 0.05$ and reference point for the hypervolume calculation $\rho = (2,\ldots,2) \in \mathbb{R}^m$.

Each algorithm was run on each problem 30 times. The experiments with NSGA-II, SPEA2 and IBEA were performed using the PISA multiobjective optimization environment (Bleuler et al., 2003).

3.5.3 Performance assessment

The performance assessment (see Section 2.2) was carried out separately for each pair of algorithms. Consider for example the comparison between $\text{DEMO}^{\text{NS-II}}$ and NSGA-II on one problem. First, the bounds of approximation sets of both algorithms were calculated and the approximation sets were normalized to the $[1,2]$ interval. After that, the dominance rank was calculated for each of the 60 approximation sets. The Mann-Whitney rank sum test (Conover, 1999) was used to discover if there are significant differences between the dominance ranks of the two algorithms.

Additional assessment was carried out using unary quality indicators. From the approximation sets of both algorithms, the set containing only nondominated solutions was computed and used as the reference set for the unary indicators $I^1_{\varepsilon+}$ and I^1_{R2}. Other parameters of the I^1_{R2} indicator were as follows: $(0.9,\ldots,0.9),(2.1,\ldots,2.1) \in \mathbb{R}^m$ served as the ideal and Nadir points, $\rho = 0.01$, while 501, 496 and 455 uniformly spread parameter vectors were used for the problems with two, three and four objectives, respectively. The hypervolume indicator I_{H} used the point $(2.1,\ldots,2.1) \in \mathbb{R}^m$ as the reference point. All three indicators were calculated for each approximation set of both algorithms. The significance of these outcomes was tested independently with the Fisher's independent permutation test (Conover, 1999). Because we used four performance metrics (dominance ranking and three indicators) on the same data, the significance level α_s for each significance test was set to $0.05/4 = 0.0125$ using the Bonferroni correction (Bonferroni, 1936).

Finally, the results on two objective problems were compared also by plotting the best, worst and 50%-attainment surfaces. The same procedure was repeated also for comparing DEMO^{SP2} to SPEA2, $\text{DEMO}^{\text{IB}_{\varepsilon+}}$ to $\text{IBEA}_{\varepsilon+}$, and $\text{DEMO}^{\text{IB}_{\text{HD}}}$ to IBEA_{HD}. Performance assessments were done using the PISA environment.

3.5.4 Results and discussion

Tables 3.1, 3.3, 3.5 and 3.7 present the outcomes of dominance ranking, while Tables 3.2, 3.4, 3.6 and 3.8 show the results of statistical tests on the $I^1_{\varepsilon+}$, I_{H} and I^1_{R2} indicators.

Looking at the outcomes of dominance ranking, we can observe that on some problems, approximation sets of DEMO achieve significantly better domination ranks than the approximation sets of the basic GA. Only on two problems (when I_{HD} indicator-based selection was used) the basic GA outperforms DEMO. These cases will be commented in more detail later. On the majority of problems, however, there are no significant differences between the two algorithms with regard to dominance ranking.

As expected, when dominance ranking shows a significant difference between two algorithms, so do the three applied indicators (an exception will be explained later). On the majority of problems, DEMO achieves significantly better results with regard to the chosen indicator. Note that on a few problems (see for example DTLZ5 for $m = 4$ in Table 3.2), DEMO is significantly better than the basic GA with regard to one indicator (I^1_{R2}) and significantly worse with regard to another indicator (I_{H}). This suggests that the outcomes of DEMO and the basic GA are incomparable on such problems.

Besides these results, we also investigated the plots of approximation sets (for $m = 2$ and 3) and plots of attainment surfaces (for $m = 2$) to gain further insight into the comparison between DEMO and the basic GA. Despite statistical tests show that there is almost always a significant difference in indicator values of the two algorithms, in general no noticeable distinction was *visible* between the approximation sets (and attainment surfaces) of DEMO and the basic GA on the DTLZ2, DTLZ4, DTLZ5, DTLZ7, WFG3, WFG4, WFG5, WFG8 and WFG9 problems. On the DTLZ1, DTLZ3 and DTLZ6 problems, where it is very difficult to converge to the Pareto optimal front, and on the non-separable WFG6 problem, DEMO generally attained the Pareto optimal front more efficiently than the basic GA. On the DTLZ3, WFG1, WFG2 and WFG7 problems DEMO achieved better spread of solutions along the Pareto optimal front than the basic GA.

In the following, we review the performance of DEMO and basic GA on selected problems in more detail.

DEMO$^{\text{NS-II}}$ vs. NSGA-II. The comparison between DEMO and the basic GA is very favorable to DEMO when nondominated sorting is used for environmental selection. Let us explore in more detail the outcomes of both algorithms on the DTLZ6 problem. The diffi-

Table 3.1: Outcomes of the Mann-Whitney rank sum test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on dominance ranking for DEMO$^{\text{NS-II}}$ and NSGA-II. The '▲ p-value' ('▽ p-value') denotes the problems, on which DEMO$^{\text{NS-II}}$ is significantly better (worse) than NSGA-II, while '-' indicates there are no significant differences between the two algorithms.

	$m=2$		$m=3$		$m=4$	
DTLZ1	-		▲	3.9×10^{-13}	▲	7.9×10^{-15}
DTLZ2	-		-		-	
DTLZ3	▲	2.0×10^{-11}	▲	3.9×10^{-13}	▲	3.5×10^{-12}
DTLZ4	▲	0.0052	-		-	
DTLZ5	-		-		-	
DTLZ6	▲	4.1×10^{-14}	▲	7.9×10^{-15}	▲	3.9×10^{-13}
DTLZ7	-		-		▲	1.5×10^{-4}
WFG1	-		▲	1.6×10^{-7}	-	
WFG2	-		-		-	
WFG3	-		-		-	
WFG4	-		-		-	
WFG5	-		-		-	
WFG6	▲	1.5×10^{-4}	-		-	
WFG7	-		-		-	
WFG8	-		-		-	
WFG9	-		-		-	

Table 3.2: Outcomes of the Fisher-independent test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on indicator values for DEMO$^{\text{NS-II}}$ and NSGA-II. A '▲' ('▽') under the indicator I means that DEMO$^{\text{NS-II}}$ is significantly better (worse) than NSGA-II regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I. The p-values are omitted for the sake of brevity (see Tables B.1, B.2 and B.3 in Appendix B for complete results).

	$m=2$			$m=3$			$m=4$		
	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}
DTLZ1	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ2	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ3	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ4	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ5	▲	▲	▲	▲	▲	▲	-	▽	▲
DTLZ6	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ7	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG1	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG2	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG3	-	▲	▲	▲	▲	▲	-	▽	▽
WFG4	-	▲	-	-	▲	▲	▲	▲	▲
WFG5	▽	-	▽	-	-	-	-	-	▲
WFG6	▲	▲	▲	▲	▲	▲	-	▲	▲
WFG7	▲	▲	▲	▲	▲	▲	-	▲	-
WFG8	▲	-	▲	-	-	-	-	-	▽
WFG9	-	▲	-	-	▲	▲	-	-	-

Table 3.3: Outcomes of the Mann-Whitney rank sum test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on dominance ranking for DEMO^{SP2} and SPEA2. The '▲ p-value' ('▽ p-value') denotes the problems, on which DEMO^{SP2} is significantly better (worse) than SPEA2, while '-' indicates there are no significant differences between the two algorithms.

	$m=2$		$m=3$		$m=4$	
DTLZ1	-		▲	9.7×10^{-13}	▲	3.2×10^{-13}
DTLZ2	-		-		-	
DTLZ3	▲	2.2×10^{-11}	▲	2.0×10^{-14}	▲	3.2×10^{-13}
DTLZ4	-		-		-	
DTLZ5	-		-		-	
DTLZ6	▲	2.0×10^{-14}	▲	7.9×10^{-15}	▲	1.7×10^{-12}
DTLZ7	-		-		-	
WFG1	-		▲	2.7×10^{-9}	-	
WFG2	-		-		-	
WFG3	-		-		-	
WFG4	-		-		-	
WFG5	-		-		-	
WFG6	▲	2.6×10^{-6}	-		-	
WFG7	-		-		-	
WFG8	-		-		-	
WFG9	-		-		-	

Table 3.4: Outcomes of the Fisher-independent test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on indicator values for DEMO^{SP2} and SPEA2. A '▲' ('▽') under the indicator I means that DEMO^{SP2} is significantly better (worse) than SPEA2 regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I. The p-values are omitted for the sake of brevity (see Tables B.4, B.5 and B.6 in Appendix B for complete results).

	$m=2$			$m=3$			$m=4$		
	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}
DTLZ1	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ2	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ3	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ4	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ5	▲	▲	▲	▲	▲	▲	▽	-	▲
DTLZ6	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ7	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG1	▲	▲	▲	▽	-	-	▲	▲	▲
WFG2	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG3	▲	▲	▲	-	-	-	▽	-	-
WFG4	-	▲	-	-	▲	▲	-	▲	▲
WFG5	▽	▽	▽	▲	▲	▲	▲	▲	▲
WFG6	▲	▲	▲	▲	▲	▲	-	▲	▲
WFG7	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG8	▲	-	▲	▲	▲	▲	-	▲	▲
WFG9	-	-	-	▲	▲	▲	-	▲	▲

Table 3.5: Outcomes of the Mann-Whitney rank sum test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on dominance ranking for DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$. The '▲ p-value' ('▽ p-value') denotes the problems, on which DEMO$^{\text{IB}_{\varepsilon+}}$ is significantly better (worse) than IBEA$_{\varepsilon+}$, while '-' indicates there are no significant differences between the two algorithms.

	$m=2$		$m=3$		$m=4$	
DTLZ1	-		-		-	
DTLZ2	-		-		-	
DTLZ3	-		▲	1.9×10^{-7}	▲	8.3×10^{-9}
DTLZ4	-		-		-	
DTLZ5	-		-		-	
DTLZ6	▲	2.1×10^{-13}	▲	1.9×10^{-12}	-	
DTLZ7	-		-		-	
WFG1	-		-		-	
WFG2	-		-		-	
WFG3	-		-		-	
WFG4	-		-		-	
WFG5	-		-		-	
WFG6	▲	4.2×10^{-7}	-		-	
WFG7	-		-		-	
WFG8	-		-		-	
WFG9	-		-		-	

Table 3.6: Outcomes of the Fisher-independent test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on indicator values for DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$. A '▲' ('▽') under the indicator I means that DEMO$^{\text{IB}_{\varepsilon+}}$ is significantly better (worse) than IBEA$_{\varepsilon+}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I. The p-values are omitted for the sake of brevity (see Tables B.7, B.8 and B.9 in Appendix B for complete results).

	$m=2$			$m=3$			$m=4$		
	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}
DTLZ1	▲	▲	▲	▲	▲	▲	▲	▲	-
DTLZ2	-	-	▲	-	▲	▲	-	▲	▲
DTLZ3	▽	▽	▽	▲	▲	▲	▲	▲	▲
DTLZ4	-	-	▲	-	▲	▲	▲	▲	▲
DTLZ5	-	-	▲	-	-	▲	-	▽	▲
DTLZ6	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ7	-	-	▲	▽	-	▲	-	-	-
WFG1	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG2	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG3	▲	▲	▲	-	▲	▲	▽	▲	▽
WFG4	▽	▽	▽	-	▲	▲	-	▲	▲
WFG5	▽	▽	▽	-	-	-	-	▲	▲
WFG6	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG7	▲	▲	▲	-	▲	▲	-	▲	▲
WFG8	-	-	-	-	-	▲	-	-	▲
WFG9	-	▲	-	▲	-	▲	-	▲	▲

Table 3.7: Outcomes of the Mann-Whitney rank sum test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on dominance ranking for DEMO$^{\text{IB}_{\text{HD}}}$ and IBEA$_{\text{HD}}$. The '▲ p-value' ('▽ p-value') denotes the problems, on which DEMO$^{\text{IB}_{\text{HD}}}$ is significantly better (worse) than IBEA$_{\text{HD}}$, while '-' indicates there are no significant differences between the two algorithms.

	$m=2$		$m=3$		$m=4$	
DTLZ1	-		-		-	
DTLZ2	-		-		-	
DTLZ3	-		▲	3.0×10^{-12}	▲	9.7×10^{-12}
DTLZ4	▲	0.0013	▲	0.0104	-	
DTLZ5	-		-		-	
DTLZ6	▲	1.2×10^{-13}	▲	2.4×10^{-11}	-	
DTLZ7	▽	0.0023	▽	1.3×10^{-7}	▽	1.8×10^{-7}
WFG1	-		▽	0.0058	-	
WFG2	-		-		-	
WFG3	-		-		-	
WFG4	-		-		-	
WFG5	-		-		-	
WFG6	▲	6.1×10^{-6}	-		-	
WFG7	-		-		-	
WFG8	-		-		-	
WFG9	-		-		-	

Table 3.8: Outcomes of the Fisher-independent test ($\alpha = 0.05$, $\alpha_s = 0.0125$) on indicator values for DEMO$^{\text{IB}_{\text{HD}}}$ and IBEA$_{\text{HD}}$. A '▲' ('▽') under the indicator I means that DEMO$^{\text{IB}_{\text{HD}}}$ is significantly better (worse) than IBEA$_{\text{HD}}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I. The p-values are omitted for the sake of brevity (see Tables B.10, B.11 and B.12 in Appendix B for complete results).

	$m=2$			$m=3$			$m=4$		
	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}	$I^1_{\varepsilon+}$	I_{H}	I^1_{R2}
DTLZ1	▲	▲	▲	▲	▲	▲	▲	▲	-
DTLZ2	-	-	▲	-	▲	▲	-	-	▲
DTLZ3	-	▽	▽	▲	▲	▲	▲	▲	▲
DTLZ4	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ5	-	-	▲	▽	▽	▲	▲	▲	▲
DTLZ6	▲	▲	▲	▲	▲	▲	▲	▲	▲
DTLZ7	▽	▽	▽	▽	▽	▽	▽	▽	-
WFG1	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG2	▲	▲	▲	▲	▲	▲	-	▲	▲
WFG3	-	▲	▲	-	▲	▲	▽	▽	▽
WFG4	-	▽	▽	-	▲	▲	-	▲	▲
WFG5	▽	▽	▽	-	-	▲	-	▽	▲
WFG6	▲	▲	▲	▲	▲	▲	▲	▲	▲
WFG7	▲	▲	▲	-	▲	▲	-	▲	▲
WFG8	-	-	-	-	-	▽	-	-	▽
WFG9	▲	▲	-	-	▲	▲	-	-	▲

culty of this problem reflects in poor convergence of certain algorithms to the Pareto optimal front. Figure 3.2 shows that DEMO$^{NS\text{-}II}$ reaches the Pareto optimal front for $m = 2$ and $m = 3$, while NSGA-II does not. The most probable cause for such behavior is the repair method used by DEMO, since in this problem, the Pareto optimal set lies on the bounds of the decision space and boundary points are likely to be found after applying DEMO's repair method.

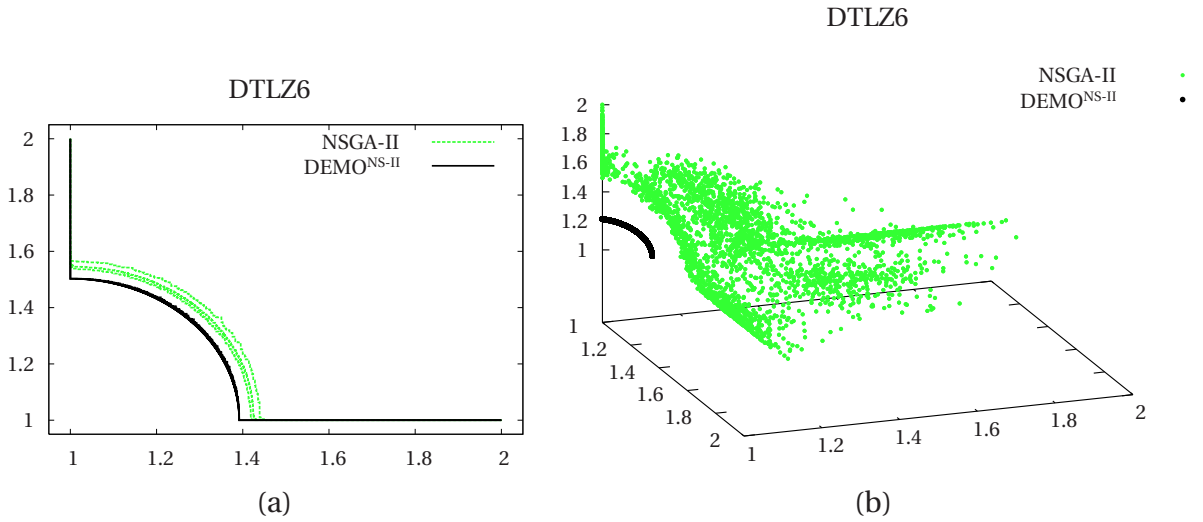


Figure 3.2: Plots of normalized attainment surfaces and approximation sets of algorithms DEMO$^{NS\text{-}II}$ and NSGA-II on the DTLZ6 problem: (a) the best, worst and 50%-attainment surfaces for each algorithm on the problem with two objectives; (b) 30 approximation sets for each algorithm on the problem with three objectives.

It is interesting to note, however, that on the only other problem (DTLZ7) where the Pareto optimal set lies on the bounds of the decision space no big differences between approximation sets could be noticed. This is probably because on this problem, none of the algorithms has difficulties in reaching the Pareto optimal front.

DEMO^{SP2} vs. SPEA2. Using the strength Pareto approach to environmental selection yields very similar results in the comparison between DEMO and the basic GA as the use of non-dominated sorting. The findings from the previous paragraph (on the DTLZ6 and DTLZ7 problems) also hold for DEMO^{SP2} and SPEA2. Similarly, some of the characteristics of the comparison between DEMO^{SP2} and SPEA2 on the WFG1 problem, which will be discussed in more detail shortly, are true also when comparing DEMO$^{NS\text{-}II}$ and NSGA-II.

Consider now the WFG1 problem for $m = 2$. From the plot of attainment surfaces (Figure 3.3) we can see that DEMO^{SP2} covers a wider portion of the Pareto optimal front than

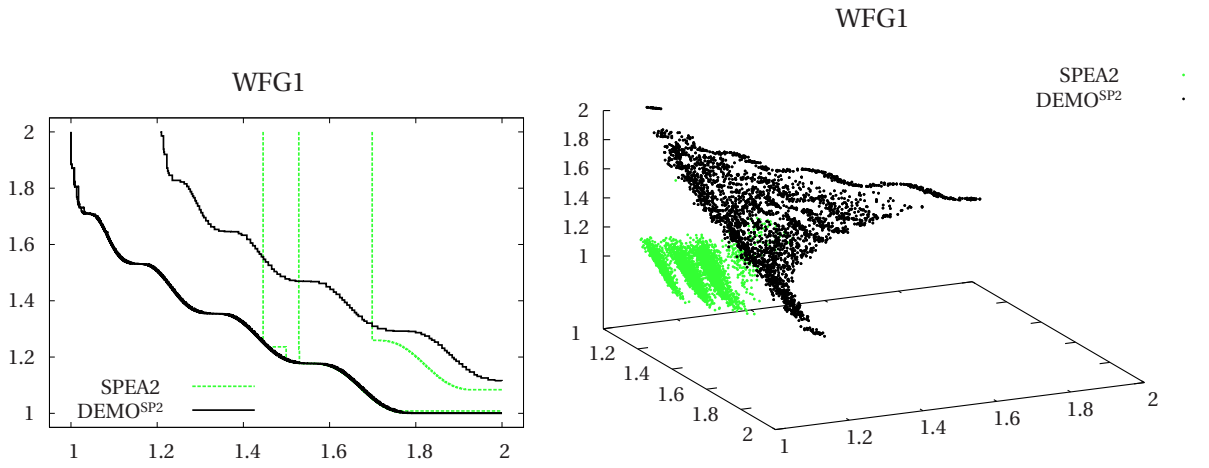


Figure 3.3: Plots of normalized attainment surfaces and approximation sets of algorithms DEMO^{SP2} and SPEA2 on the WFG1 problem: (a) the best, worst and 50%-attainment surfaces for each algorithm on the problem with two objectives; (b) 30 approximation sets for each algorithm on the problem with three objectives.

SPEA2, while having comparable convergence properties in the best and average case (50%-attainment surface) and a little worse in the worst case. When this problem is tackled in three objectives, DEMO^{SP2} loses some of its convergence power while keeping the good coverage. SPEA2, on the other hand, still covers only a small part of the whole front, while achieving much better convergence than DEMO^{SP2}. Although this is not visible from the plots, we wish to point out that neither of the algorithms reached the Pareto optimal front for this problem.

There is an additional interesting aspect of the results on this problem, which is related to the performance assessment using dominance ranking and quality indicators. Note that Tables 3.3 and 3.4 show that DEMO^{SP2} is significantly better than SPEA2 on WFG1 for $m = 3$ with regard to the dominance ranking, and significantly worse than SPEA2 with regard to the $I^1_{\varepsilon+}$ indicator. This happens because the approximation sets of DEMO^{SP2} are never completely dominated by other approximation sets, while the approximation sets of SPEA2 sometimes dominate each other. As a result, the dominance ranking prefers DEMO^{SP2} to SPEA2 although approximation sets of SPEA2 are closer to the Pareto optimal front than approximation sets of DEMO^{SP2}.

DEMO$^{IB_{\varepsilon+}}$ vs. IBEA$_{\varepsilon+}$. From Tables 3.5 and 3.6 (and also Tables 3.7 and 3.8) it is obvious that using indicator-based environmental selection brought DEMO less improvement over the basic GA than using the first two approaches. Most of the findings from the comparison

between DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$ hold also for the comparison between DEMO$^{\text{IB}_{\text{HD}}}$ and IBEA$_{\text{HD}}$.

Consider the DTLZ1 problem for $m = 2$ (Figure 3.4). The vectors from the approximation set found by IBEA$_{\varepsilon+}$ are unevenly spread along the Pareto optimal front. Most of them are gathered around the extremities of the Pareto optimal front and only a few of them are located in the central part of the front. This yields interestingly shaped attainment surfaces by IBEA$_{\varepsilon+}$, which show that DEMO$^{\text{IB}_{\varepsilon+}}$ outperformed IBEA$_{\varepsilon+}$ on this problem. Similar results can be seen also for $m = 3$, where DEMO$^{\text{IB}_{\varepsilon+}}$ again achieves good convergence and spread, while IBEA$_{\varepsilon+}$ has good convergence but poor spread of vectors in the objective space.

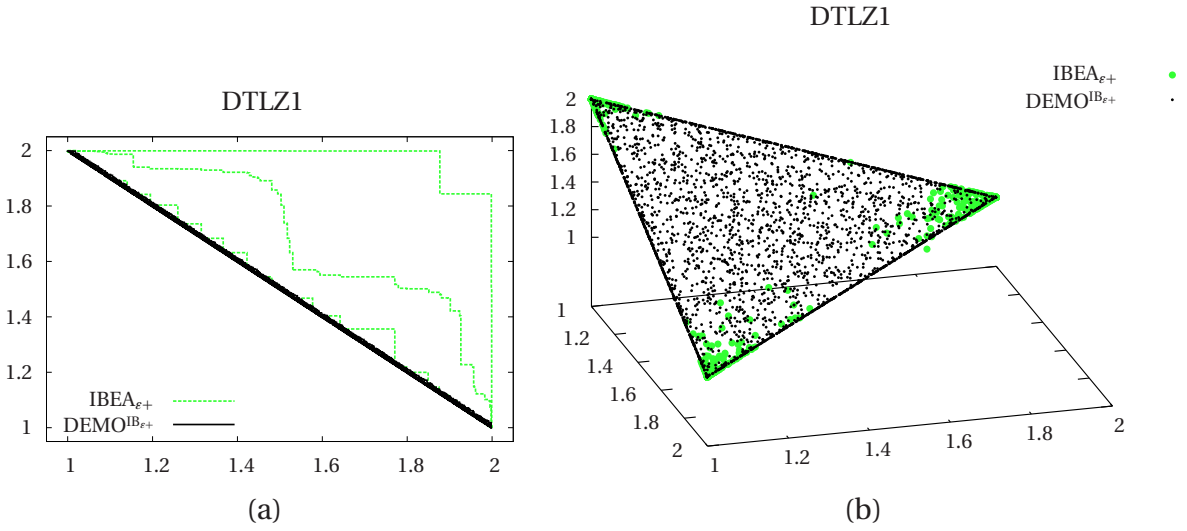


Figure 3.4: Plots of normalized attainment surfaces and approximation sets of algorithms DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$ on the DTLZ1 problem: (a) the best, worst and 50%-attainment surfaces for each algorithm on the problem with two objectives; (b) 30 approximation sets for each algorithm on the problem with three objectives.

DEMO$^{\text{IB}_{\text{HD}}}$ vs. IBEA$_{\text{HD}}$. Using I_{HD} indicator-based selection, DEMO was for the first time outperformed by the basic GA with regard to dominance ranking. The DTLZ7 problem with 2^{m-1} disconnected Pareto optimal regions proved to be very hard for DEMO$^{\text{IB}_{\text{HD}}}$. While the convergence to the Pareto optimal front was not difficult, maintaining diverse solutions was hard for DEMO$^{\text{IB}_{\text{HD}}}$. Out of 30 runs for each objective space dimensionality, DEMO converged to a single point 29 times for $m = 2$, 26 times for $m = 3$ and 25 times for $m = 4$. Note, however, that in combination with all other approaches to environmental selection (including using $I^1_{\varepsilon+}$ instead of I_{HD} in indicator-based selection), DEMO could always maintain diverse solutions.

Let us analyze in more detail also the DTLZ3 problem whose main difficulty rises from its $3^{10}-1$ local Pareto optimal fronts. As shown in the plots in Figure 3.5, IBEA_{HD} has more difficulties in reaching the Pareto optimal front than $\text{DEMO}^{\text{IB}_{\text{HD}}}$. In the case of two objectives, $\text{DEMO}^{\text{IB}_{\text{HD}}}$ performs worse than IBEA_{HD} in the worst case while achieving a much better spread in the best case. On the three-objective problem, $\text{DEMO}^{\text{IB}_{\text{HD}}}$ achieves good results in all 30 runs, while IBEA_{HD} gets stuck in local optima and has a poor spread of solutions.

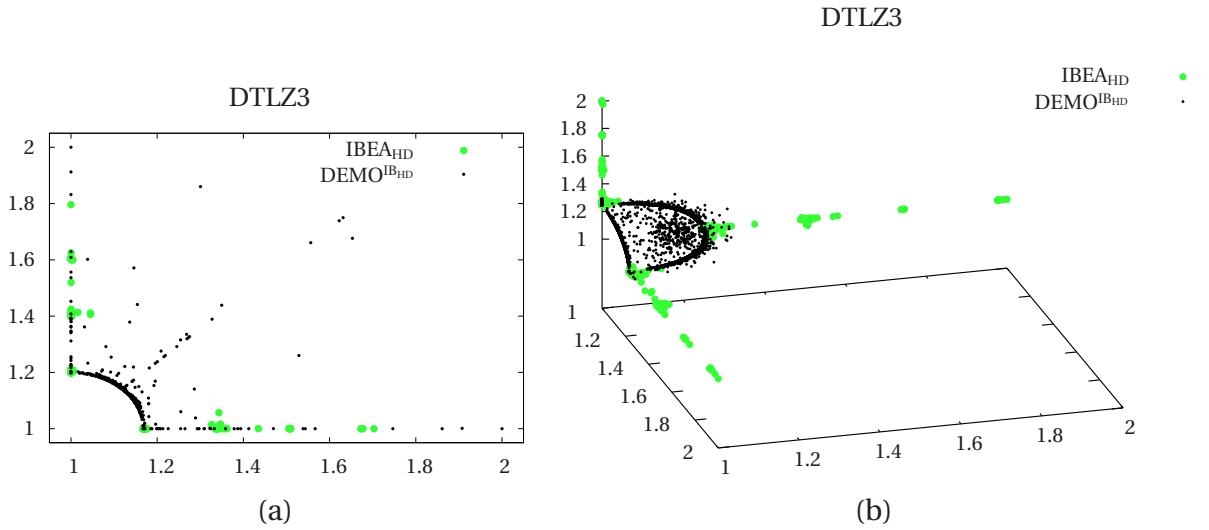


Figure 3.5: Plots of normalized approximation sets of $\text{DEMO}^{\text{IB}_{\text{HD}}}$ and IBEA_{HD} on the DTLZ3 problem: (a) 30 approximation sets for each algorithm on the problem with two objectives; (b) 30 approximation sets for each algorithm on the problem with three objectives.

3.5.5 Summary

In this study, we compared the performance of the well-known multiobjective evolutionary algorithms NSGA-II, SPEA2, $\text{IBEA}_{\varepsilon+}$ and IBEA_{HD} to their DE-based counterparts $\text{DEMO}^{\text{NS-II}}$, DEMO^{SP2}, $\text{DEMO}^{\text{IB}_{\varepsilon+}}$ and $\text{DEMO}^{\text{IB}_{\text{HD}}}$ on 16 state-of-the-art benchmark problems (each with 2, 3 and 4 objectives). The results show that on 17% of the problems, DEMO achieved significantly better dominance ranks than the basic GA, while significantly worse dominance ranks were obtained on only 2% of problems. Furthermore, DEMO outperformed the basic GA with regard to the quality indicators on the majority (81%) of problems and was outperformed by the basic GA on only 9% of the problems.

On the basis of this results we can conclude that, in general, DE explores the decision space more efficiently than a GA (regardless of the chosen environmental selection proce-

dure). A further comparison, showing the differences among DEMO variants, is presented in the next section.

3.6 Comparison of DEMO variants

While the previous section showed that DEMO often achieves better results than the basic GA, this section focuses on the specific properties of DEMO variants to help choose among them. To this end, we use the results obtained in the experiments presented before. First, the variants are compared according to dominance ranking of their approximation sets and then, the configuration of these sets is explored in more detail.

3.6.1 Dominance ranking

This time, the bounds of approximation sets of all four DEMO variants were calculated and the approximation sets were accordingly normalized to the $[1,2]$ interval. The dominance ranks were calculated for each pair of variants separately (four variants yield six pairwise comparisons) and the Mann-Whitney rank sum test was used to check if there are significant differences between these dominance ranks. Because we made six comparisons among the variants, the significance level α_s for each significance test was set to $0.05/6 \doteq 0.008$ using the Bonferroni correction.

Significantly different dominance ranks were achieved only on three problems—see Table 3.9 for a list of all significant differences. On the DTLZ3 problem with two objectives, DEMO$^{\text{NS-II}}$ and DEMO$^{\text{SP2}}$ significantly outperform the two DEMO$^{\text{IB}}$ variants. The reason for this is that DEMO$^{\text{IB}_{\text{HD}}}$ and especially DEMO$^{\text{IB}_{\varepsilon+}}$ achieve some bad results, while the good results of DEMO$^{\text{NS-II}}$ and DEMO$^{\text{SP2}}$ are consistent in all 30 runs. Moreover, DEMO$^{\text{NS-II}}$ and DEMO$^{\text{SP2}}$ achieve better dominance ranks than DEMO$^{\text{IB}_{\varepsilon+}}$ also in the three-objective case.

As explained in the previous section, DEMO$^{\text{IB}_{\text{HD}}}$ converges to a single point in the majority of the runs on DTLZ7 for $m = 2, 3$ and 4. This is the reason why the other three variants significantly outperform DEMO$^{\text{IB}_{\text{HD}}}$ on this problem.

WFG1 skews the relative significance of different parameters thus representing a difficult problem to solve. Huband et al. (2006) have shown that NSGA-II fails to reach the Pareto optimal front on this problem even after more than 25 000 generations. Although the parameter settings of both NSGA-II and WFG1 were different from the ones used in this work, our results lead to similar conclusions: nondominated sorting and the strength Pareto approach

Table 3.9: Problems on which DEMO variants achieved significantly different dominance ranks according to the Mann-Whitney rank sum test ($\alpha = 0.05$, $\alpha_s \doteq 0.008$). With '$\text{DEMO}^{\text{X}} \blacktriangleleft \text{DEMO}^{\text{Y}}$' we denote that DEMO^{X} is significantly better than DEMO^{Y}.

DTLZ3 ($m = 2$):	$\text{DEMO}^{\text{NS-II}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\varepsilon+}}$
	$\text{DEMO}^{\text{NS-II}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\text{HD}}}$
	$\text{DEMO}^{\text{SP2}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\varepsilon+}}$
	$\text{DEMO}^{\text{SP2}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\text{HD}}}$
	$\text{DEMO}^{\text{IB}_{\text{HD}}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\varepsilon+}}$
DTLZ3 ($m = 3$):	$\text{DEMO}^{\text{NS-II}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\varepsilon+}}$
	$\text{DEMO}^{\text{SP2}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\varepsilon+}}$
DTLZ7 ($m = 2,3,4$):	$\text{DEMO}^{\text{NS-II}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\text{HD}}}$
	$\text{DEMO}^{\text{SP2}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\text{HD}}}$
	$\text{DEMO}^{\text{IB}_{\varepsilon+}} \blacktriangleleft \text{DEMO}^{\text{IB}_{\text{HD}}}$
WFG1 ($m = 3,4$):	$\text{DEMO}^{\text{IB}_{\varepsilon+}} \blacktriangleleft \text{DEMO}^{\text{NS-II}}$
	$\text{DEMO}^{\text{IB}_{\varepsilon+}} \blacktriangleleft \text{DEMO}^{\text{SP2}}$
	$\text{DEMO}^{\text{IB}_{\text{HD}}} \blacktriangleleft \text{DEMO}^{\text{NS-II}}$
	$\text{DEMO}^{\text{IB}_{\text{HD}}} \blacktriangleleft \text{DEMO}^{\text{SP2}}$

are limited on this problem for $m = 3$ and 4, while both indicator-based DEMO variants were able to achieve significantly better results.

3.6.2 Configuration of approximation sets

Deb et al. (2005) have shown that on the DTLZ problems with 3 objectives, the algorithms NSGA-II and SPEA2 achieve similar convergence, but a different distribution of objective vectors. On most problems, the objective vectors found by SPEA2 were distributed more uniformly than the objective vectors found by NSGA-II. Here, we wish to verify if the same holds for $\text{DEMO}^{\text{NS-II}}$ and DEMO^{SP2} , and compare the configurations of approximation sets of all DEMO variants.

First, we calculated the I_{H} indicator for each normalized approximation set (all parameters of I_{H} were set the same as in the previous section). For each variant and each problem for $m = 2$ and 3, we plotted the approximation set with the best I_{H}. Here, we present only the plots for the DTLZ5 ($m = 2$) and DTLZ2 ($m = 3$) problems, as they are the most representative (see Figures 3.6 and 3.7).

Like Deb et al. (2005), we found that DEMO^{SP2} in general distributes the objective vectors more evenly than $\text{DEMO}^{\text{NS-II}}$. This is better visible on the three objective problems and

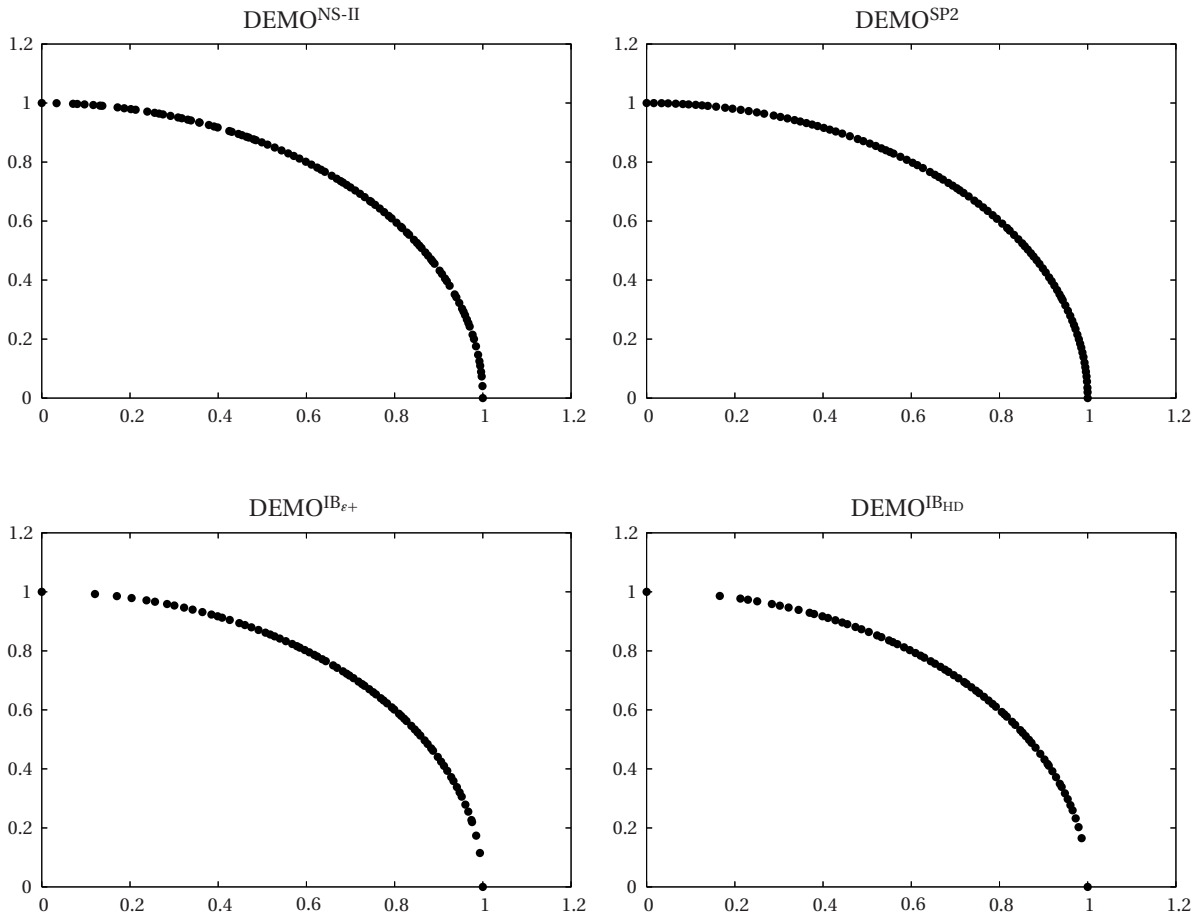


Figure 3.6: Plots of the best approximation sets (according to I_H) found by DEMO$^{\text{NS-II}}$, DEMO$^{\text{SP2}}$, DEMO$^{\text{IB}_{\varepsilon+}}$ and DEMO$^{\text{IB}_{\text{HD}}}$ on the DTLZ5 problem with two objectives.

is due to the strength Pareto approach, which works better than nondominated sorting, although at a higher computational cost. The approximation sets found by DEMO$^{\text{IB}_{\varepsilon+}}$ and DEMO$^{\text{IB}_{\text{HD}}}$ are similar to each other and very different from the ones found by DEMO$^{\text{NS-II}}$ and DEMO$^{\text{SP2}}$. It seems that the indicator-based selection prefers objective vectors that lie on the extremities of the Pareto-optimal front (see Figure 3.7). In addition, on three objective problems DEMO$^{\text{IB}_{\text{HD}}}$ keeps only few vectors near the center of the Pareto optimal front. Note also that while the extremities of the Pareto optimal front are well-represented in both DEMO$^{\text{IB}}$ variants, there is a gap between the most extreme objective vectors and the objective vectors that are closest to them. This gap appears only in problems with concave Pareto optimal fronts.

The properties of approximation sets explained on the example of DTLZ5 and DTLZ2 are in general true also for other problems. In particular, very similar approximation sets (also

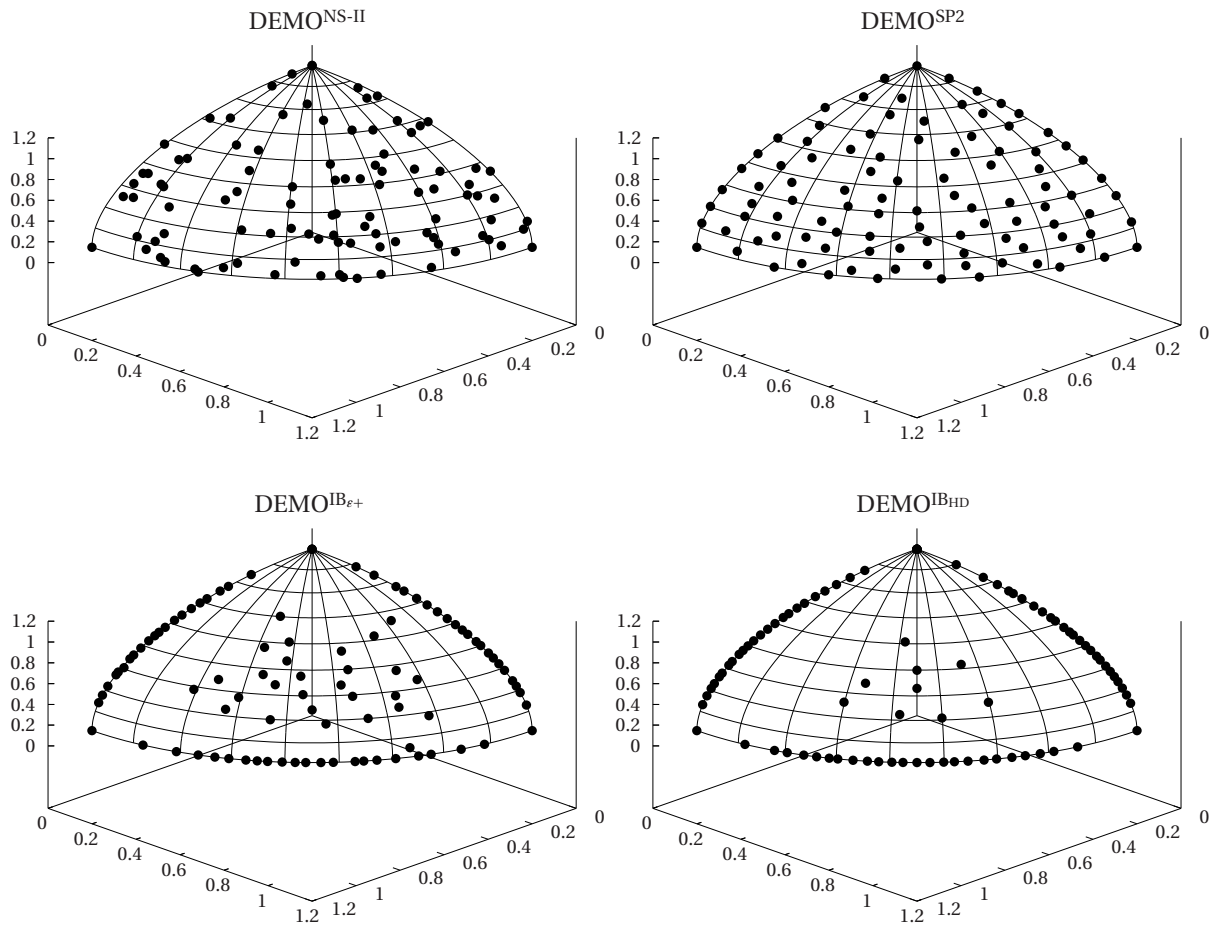


Figure 3.7: Plots of the best approximation sets (according to I_H) found by DEMO$^{NS\text{-}II}$, DEMO^{SP2}, DEMO$^{IB_{\varepsilon+}}$ and DEMO$^{IB_{HD}}$ on the DTLZ2 problem with three objectives.

because of the similar geometry of the Pareto optimal front) are achieved on the DTLZ3, DTLZ4, DTLZ6 (for $m = 2$) and WFG4 to WFG9 problems. On DTLZ1 and WFG3, there are only minor differences among the variants. The DTLZ5 and DTLZ6 problems have degenerated Pareto optimal fronts for $m = 3$. In this case not all mentioned differences among the variants can be seen. DTLZ7 is a problem with discontinuous Pareto optimal front. Nevertheless, we could see the specific properties of the approximation sets for each separate portion of the Pareto optimal front. The WFG2 problem has a discontinuous Pareto optimal front, where both DEMO^{IB} achieve lower coverage of the front than DEMO$^{NS\text{-}II}$ and DEMO^{SP2}.

3.6.3 Summary

The comparison of DEMO variants showed that all variants achieve comparable convergence to the Pareto optimal front on all but three out of 16 problems (see Table 3.9), where some variants faced difficulties because of the specific properties of these problems. The visual presentation of the approximation sets helped us compare the configurations obtained by the different environmental selection procedures. It seems that the strength Pareto approach used by DEMO^{SP2} generates approximation sets that are the most similar to our 'grid-like' perception of how should well-distributed objective vectors look like. This encouraged us to use the DEMO^{SP2} variant in the case study presented in the next chapter.

Chapter 4

Optimizing Accuracy and Size of Decision Trees

While the previous chapter presented in detail the DEMO algorithm and its performance on artificial benchmark problems, in this chapter, DEMO is applied to the real-world optimization problem of finding accurate and small decision trees. We start with Section 4.1, which introduces decision trees—popular machine learning models for classification and regression tasks. Section 4.2 shows how DEMO is employed for finding tradeoffs between accurate and small decision trees and presents the results of the experiments performed on some real domains. The chapter concludes with the discussion in Section 4.3.

4.1 Introduction

4.1.1 Decision trees for classification

Let us first introduce some basic terms used in machine learning. Consider a set of instances, described by some attributes and classified into classes (see Table 4.1). *Classification* is the task of determining the class for each possible instance; or in other words, of finding the function, which maps the space of attributes into classes. One of the most popular ways of classifying data is using *decision trees* (Breiman et al., 1984). One of their main advantages is comprehensibility—they are easy to understand and use.

Table 4.1: The weather dataset (Witten and Frank, 2005), with attributes *outlook*, *temperature*, *humidity* and *windy*, where each instance is classified either in the 'yes' or 'no' class of *play*.

outlook	temperature	humidity	windy	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Example 4.1. *The weather dataset presented in Table 4.1 contains a set of weather conditions, which are suitable (or not) for playing an outdoor game. This dataset is characterized by four attributes (*outlook, temperature, humidity *and* windy*) which describe the instances, while the outcome (*play*) can be either 'yes' or 'no'. The decision tree constructed from these instances (see Figure 4.1) describes the weather data set and at the same time defines the outcomes for each possible combination of attribute values.*

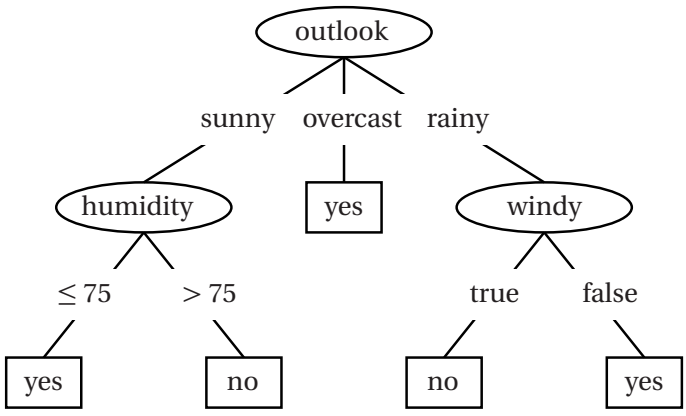


Figure 4.1: A decision tree for the weather dataset. The value in each leaf defines the class for all the instances that satisfy the conditions on the path between the root of the tree and the leaf in question.

Decision trees for classification are usually evaluated with respect to two objectives: *accuracy* and *size*. Accuracy can be estimated by the portion of instances that are correctly classified by the decision tree T:

$$acc_1(T,\mathcal{Y}) = \frac{|\{\boldsymbol{y} \in \mathcal{Y} \mid \boldsymbol{y} \text{ correctly classified by } T\}|}{|\mathcal{Y}|}, \tag{4.1}$$

where $\mathcal{Y}$ is the set of all instances, while size can be simply defined as:

$$size(T) = |\{\text{nodes in } T\}|. \tag{4.2}$$

The principal task of a decision tree is to predict the classes for previously unseen instances. If the tree is constructed using all available instances, the accuracy calculated on the same instances is not representative. Therefore, as a rule, the decision tree is constructed using only a portion of the data $\mathcal{Y}_{\text{train}}$ called *training set*, while the rest of the data $\mathcal{Y}_{\text{test}}$, called *test set*, is used for estimating its accuracy:

$$acc_2(T_{\text{train}},\mathcal{Y}_{\text{test}}) = \frac{|\{\boldsymbol{y} \in \mathcal{Y}_{\text{test}} \mid \boldsymbol{y} \text{ correctly classified by } T_{\text{train}}\}|}{|\mathcal{Y}_{\text{test}}|}.$$

This kind of accuracy estimation fits our requirements better than the previous one, but it is nevertheless highly dependent on the split of data into the training and test sets. It is therefore advisable to repeat this procedure k times—each time reserving a different $1/k$ proportion of instances for the test set and using the remaining instances as the training set. In this way, every instance is used exactly once for testing and $k-1$ times for training:

$$acc_3(T,\mathcal{Y},k) = \frac{1}{k}\sum_{j=1}^{k} \frac{|\{\boldsymbol{y} \in \mathcal{Y}^j_{\text{test}} \mid \boldsymbol{y} \text{ correctly classified by } T^j_{\text{train}}\}|}{|\mathcal{Y}^j_{\text{test}}|}, \tag{4.3}$$

where $\mathcal{Y}^j_{\text{train}} \cup \mathcal{Y}^j_{\text{test}} = \mathcal{Y}$ for each $j = 1,\ldots,k$ and $\bigcup_{j=1}^{k} \mathcal{Y}^j_{\text{test}} = \mathcal{Y}$. This method is called *k-fold cross validation*.

While accuracy assesses the prediction capabilities of a decision tree, its size is closely related to its complexity and can be used to estimate its comprehensibility—smaller trees are usually easier to understand than larger trees. A large tree can be made smaller by *pruning*. Using *prepruning*, the tree stops developing subtrees during the tree-building process, while *postpruning* prunes the subtrees of an already constructed tree. If the decision tree is meant for extracting information from the dataset rather than for prediction, it is very important that the tree is of a controllable size—small enough for people to understand. Moreover, smaller trees are often preferred to larger ones as they do not *overfit* the training set and are less sensitive to noise. On the other hand, trees that are too small can *underfit* the data by not describing the dataset well enough. Consequently, it is very important to be able to find a good tradeoff between accuracy and size of decision trees.

4.1.2 Optimization problem

Our optimization problem consists of finding the parameter settings of machine learning algorithms in order to construct accurate and small trees for a given domain. Accuracy is estimated using the cross validation procedure (4.3), while the size of the tree is calculated with (4.2). Accuracy must be maximized and size minimized. We tackle this problem for the special case of decision trees induced by the C4.5 algorithm (Quinlan, 1993), or more precisely, its Java implementation in the Weka environment (Witten and Frank, 2005) called *J48*.

When building J48 trees, several parameters need to be set (see Table 4.2 for the parameters considered in this study and (Witten and Frank, 2005) for more details on their meaning). For example, the minimum number of instances per leaf must be defined. The user can choose whether to build exclusively binary trees. In addition, the constructed tree can be either pruned or unpruned. If pruned, the confidence factor used for pruning must be set and the choice must be made whether to use subtree raising in postpruning or not. The large amount of possible parameter settings calls for a heuristic method for solving this problem.

Table 4.2: Parameters for building J48 trees.

Name	Possible values	Default value	Description
M – number of instances	$1, 2, \ldots$	2	The minimum number of instances in any leaf (higher values result in smaller trees).
U – unpruned trees	yes/no	no	Use unpruned tree (the default value 'no' means that the tree is pruned).
C – confidence factor	$[10^{-7}, 0.5]$	0.25	The confidence factor used in postpruning (smaller values incur more pruning).
S – subtree raising	yes/no	yes	Whether to consider the subtree raising operation in postpruning.
B – use binary splits	yes/no	no	Whether to use binary splits on nominal attributes when building the tree.

4.1.3 Related work

Kohavi and John (1995) searched for parameter settings of C4.5 decision trees that would result in optimal performance on a particular dataset. They considered four parameters: M, C and S with the same meaning as described in Table 4.2, and G—a binary parameter that determined if the splitting criterion would be information gain or gain ratio. The optimization objective was 'optimal performance' of the tree, i.e., the accuracy measured using 10-fold

cross validation. The problem was tackled as a discrete optimization problem and the best-first search was chosen to explore the parameter space. The trees found by best-first search were compared to the C4.5 trees built with default parameter values on 33 datasets. At a 90% confidence level, best-first search found better parameter settings than the default on nine domains, while on one dataset, default parameter values yielded better trees than the heuristic search.

Similar experiments were performed by Mladenić (1995), who searched for the optimal setting of the m-value in m-estimate postpruning of decision trees (Cestnik and Bratko, 1991). Again, the optimization objective was the accuracy of the decision tree estimated with cross validation. She explored the decision space using several different algorithms: modified greedy search, modified beam search, simulated annealing, a genetic algorithm and enumerative search. The tests on eight datasets showed that the problem is rather simple and that all tested algorithms achieved comparable results.

Both mentioned approaches optimized only the accuracy of the decision trees. To our best knowledge, no work has been done on searching for parameter settings of decision tree building algorithms that would consider accuracy and size of the trees as two optimization objectives.

Bohanec and Bratko (1994) searched for good tradeoffs between accuracy and size of decision trees in a different way. They presented the OPT algorithm which explores the space of all trees that can be derived from a complete ID3 tree (Quinlan, 1986) by pruning, using dynamic programming. The result is an optimal sequence of pruned trees, decreasing in size, such that each tree has the highest accuracy among all possible pruned trees of the same size. While OPT works perfectly for its purpose, it has two serious drawbacks if it was to be applied to serve our needs. The first difficulty is its time complexity, which is quadratic with respect to the number of leaves of the original ID3 tree. The second disadvantage is that the accuracy of the trees is measured on the dataset that was used for constructing these trees by simply counting the additional classification errors made with pruning. If a separate test set would be used for estimating the accuracy, the time needed for building such trees would increase considerably.

A different optimization problem, which still aims at finding accurate and small C4.5 trees was tackled by Pappa et al. (2004). They used a multiobjective evolutionary algorithm for selecting subsets of attributes that would yield C4.5 decision trees of optimal accuracy and size. Their tests on 18 real-world datasets showed that trees constructed on the resulting subsets of attributes often dominated the tree that was built using all attributes and were

dominated by this tree on only two datasets.

4.2 Optimization with DEMO

Since we want to help the users of machine learning algorithms to find good trees without having to search for the right parameter settings manually, we must be careful not to demand from them to set the parameters of DEMO instead. This is why we chose a single parameter setting of DEMO for all the experiments performed on this problem. This setting was in no way fitted to the domains used and should therefore be appropriate for any classification domain.

In this section we present two sets of experiments performed on the optimization problem of finding accurate and small decision trees. The first experiments were made to estimate if DEMO approximates well to the Pareto optimal front on a modified version of this problem, while the second experiments present how DEMO solves the original problem. The results of the original problem are further inspected to see which parameter settings produce the best trees. The section concludes with an additional practical example of optimization with DEMO in an engineering domain.

4.2.1 Experiments on the modified problem

The tackled optimization problem must be simplified in order to enable the comparison between the trees found by DEMO and the trees from the Pareto optimal front, which can be obtained by exhaustively searching the decision space. We also want to see how the Pareto optimal trees compare to the trees found using OPT, since OPT explores a different decision space than DEMO and exhaustive search. Therefore, the accuracy of the induced trees is calculated on the training instances as in (4.1). The decision space contains only three (instead of five) variables:

- number of instances M with possible values $1, 2, \ldots, \left\lceil \frac{|\mathcal{Y}|+1}{2} \right\rceil$, where $|\mathcal{Y}|$ is the number of instances in the dataset;
- unpruned trees U that can take the values *yes* and *no*;
- the confidence factor C, which is discretized to the values $0.01, 0.02, \ldots, 0.50$.

Subtree raising S and binary splits B are both set to *no*, since otherwise J48 could find trees that cannot be obtained with OPT.

On this modified problem, DEMO is compared to the OPT, exhaustive search and random search algorithms. For each domain we also present the tree that was obtained using slightly modified default parameter values of J48—the subtree raising, which is used by default, is not allowed here to enable a fair comparison between the algorithms.

Experimental setup

The OPT algorithm finds the optimal sequence of pruned trees for any tree. While Bohanec and Bratko (1994) used the ID3 algorithm for inducing the original unpruned tree, we use the J48 algorithm instead. If the minimum number of instances in leaves M is greater than 1, the J48 tree is subject to prepruning—this means that for different values of M we can get different trees even if the postpruning is disabled. Therefore, to make a fair comparison between the methods we ran the OPT algorithm several times—once for each possible value of M. The trees found in all runs were combined and only the nondominated ones are shown in the results.

Although OPT finds the best trees possible, they do not form the Pareto optimal front for our problem. In fact, it is possible for OPT to find additional pruned trees that cannot be built using pruning from J48. To obtain the Pareto optimal trees, exhaustive search of the decision space is used. Note that exhaustive search can be very time-consuming[1] and was possible only because we discretized the parameter C.

The modified problem involves two discrete variables, whose values need to be repaired in DEMO in order to be able to evaluate the solutions correctly. We used the Lamarckian repair procedure to round the values of the variable M to the first nearest integer and the Baldwinian repair procedure for the binary variable C, since Lamarckian repair can be too destructive for binary variables. The other parameters of DEMO were set as follows:

- population size = 20,
- number of generations = 25,
- DE selection scheme = DE/rand/1/bin,
- scaling factor $F = 0.6$,
- probability in binomial crossover $CR = 0.6$,
- environmental selection procedure = the strength Pareto approach (as in SPEA2).

[1] Exhaustive search took more than 8 hours on a domain with 8 attributes and 12960 instances using an Intel® Pentium® 4 (3.2 GHz) computer.

Because some datasets are very large and consequently the time to build a single J48 tree very long, we limited the number of all generated trees to 500. While this was often not enough for DEMO to converge, we had to persist in the low number of evaluations to provide the final solutions in a reasonable time. The scaling factor was changed from the 'usual' 0.5 to 0.6 because of the discrete decision variables (for example, if we sum the values 1 and 2 and scale the result with $F = 0.5$, the outcome can depend on the rounding). The probability in binomial crossover was increased to 0.6 (instead of 0.3, which was used in the experiments in Sections 3.5 and 3.6) because the number of variables is small. In the presentation of the results we show all nondominated trees found by DEMO in one run.

Random search built 500 trees in one run (to match the 500 trees by DEMO). Out of all trees, only the nondominated ones are presented in the results. Because DEMO and random search are stochastic algorithms, they were run 10 times on each dataset.

All algorithms were run on six datasets. The first (*EDM*) refers to process parameter selection in electrical discharge machining (Valentinčič and Junkar, 2006), while the other five *(dermatology*, *nursery*, *splice*, *vehicle* and *vowel)* were obtained from the UCI repository (Newman et al., 1998). See Table 4.3 for some basic information on these datasets.

Table 4.3: Properties of the datasets used in the experiments.

Dataset	Number of attributes	Number of classes	Number of instances
EDM	11	2	467
dermatology	34	6	366
nursery	8	5	12960
splice	61	3	3190
vehicle	18	4	846
vowel	12	11	990

Performance assessment

Performance assessment of the considered algorithms was performed in three ways:

- The trees of the deterministic algorithms and those from the best runs of the stochastic algorithms (according to the I_{H} indicator) were plotted in the objective space.
- The best, worst and 50%-attainment surfaces were plotted for DEMO and random search.

- For an additional comparison between DEMO and random search, we assessed these algorithms in the same way as in the experiments presented in Section 3.5. We performed dominance ranking and calculated the $I^1_{\varepsilon+}$, I_{H} and I^1_{R2} quality indicators. The outcomes of these performance metrics were tested for significance. The dominance ranks were subject to the Mann-Whitney rank sum test, while the quality indicators were inspected with Fisher's independent permutation test. Because of multiple testing on the same data, the significance level α_s for each significance test was set to $0.05/4 = 0.0125$ using the Bonferroni correction.

Results

The results of the experiments are presented in Table 4.4 and Figures 4.2 and B.1 (see Appendix B). First, we can see that OPT (which explores a different decision space than the remaining algorithms) finds many trees that cannot be built using pruning from the J48 algorithm. On some datasets, the gap between OPT and exhaustive search is pretty high. For example, on the nursery dataset OPT finds 403 optimal trees, from which only 34 are attained by exhaustive search. This happens for two reasons: (1) the confidence factor used for pruning J48 trees is discretized, and (2) while OPT can prune subtree A and leave subtree B unpruned, the pruning of J48 sometimes causes either the pruning of both subtrees or none of them.

The comparison between exhaustive search and DEMO shows that DEMO attains the Pareto optimal front very well. This holds in part also for random search, which is able to converge close to the Pareto optimal front but has difficulties with covering some parts of the Pareto optimal front, like for example, larger trees on nursery and splice datasets. Nev-

Table 4.4: Outcomes of the statistical tests ($\alpha = 0.05$, $\alpha_s = 0.0125$) on different performance measures for DEMO and random search on the modified problem. The '▲ p-value' under a performance measure means that DEMO is significantly better than random search on this problem regarding the pertinent performance measure, while '-' indicates there are no significant differences between the two algorithms.

	dom_rank		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}	
EDM	▲	3.2×10^{-56}	▲	$<5.4\times10^{-6}$	▲	$<5.4\times10^{-6}$	▲	$<5.4\times10^{-6}$
dermatology	-		-		▲	0.0113	▲	$<5.4\times10^{-6}$
nursery	▲	1.3×10^{-4}	▲	$<5.4\times10^{-6}$	▲	$<5.4\times10^{-6}$	▲	$<5.4\times10^{-6}$
splice	▲	0.0026	▲	4×10^{-5}	▲	0.0105	▲	$<5.4\times10^{-6}$
vehicle	-		▲	$<5.4\times10^{-5}$	▲	$<5.4\times10^{-6}$	▲	0.0062
vowel	-		▲	4×10^{-5}	▲	$<5.4\times10^{-6}$	▲	$<5.4\times10^{-6}$

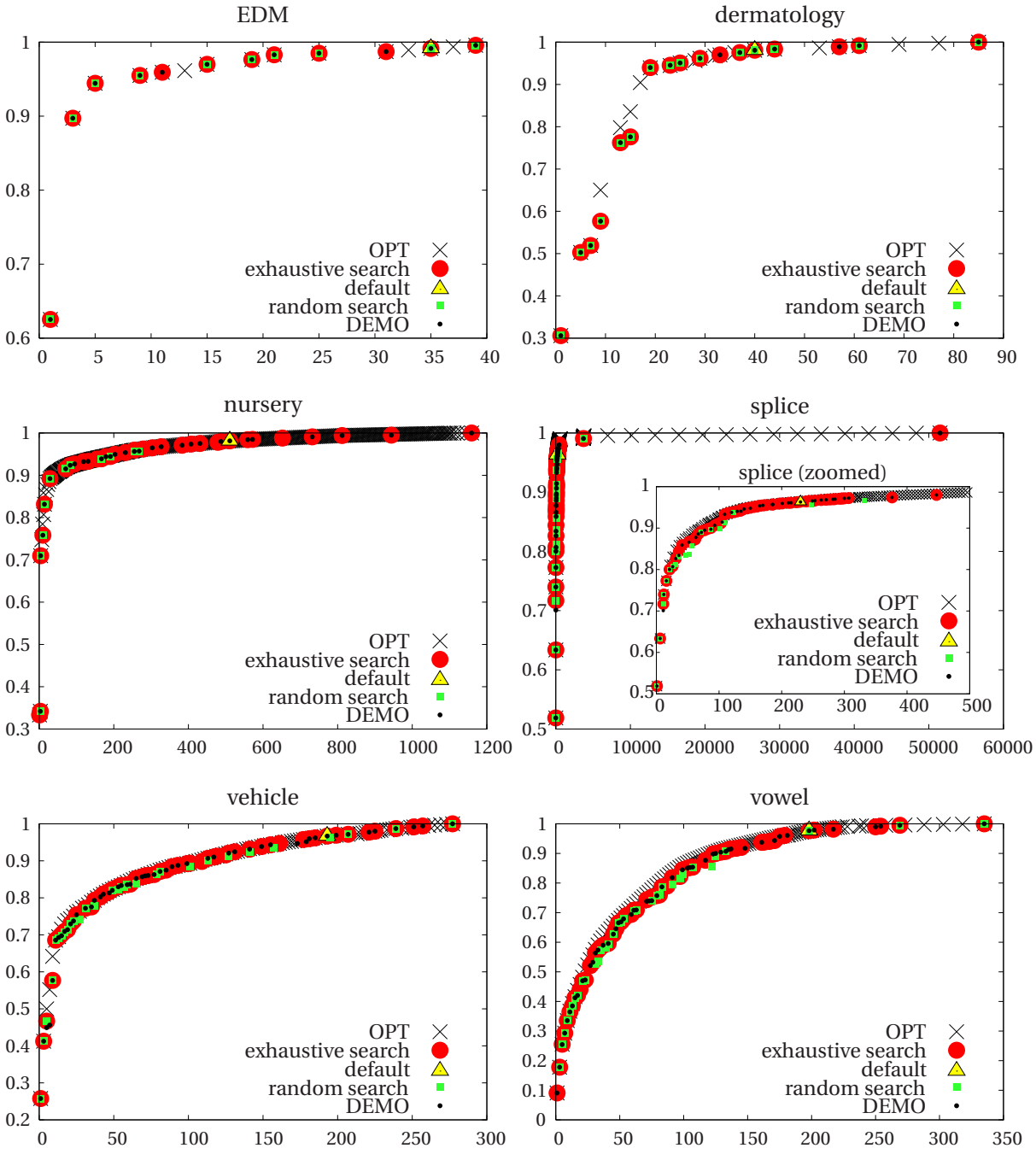


Figure 4.2: Objective values of trees found by the five algorithms on the modified optimization problem for the six datasets. The size of trees is placed on the abscissa, while classification accuracy estimated on the training instances is represented on the ordinate.

ertheless, it seems that many good tradeoffs can be achieved already by randomly searching the decision space.

The trees built using default parameter values of J48 are situated on the Pareto optimal front in all domains. They were found by DEMO in all runs on the EDM, nursery, vehicle and vowel datasets and in more than half of the runs on dermatology and splice datasets.

Statistical tests of the comparison between DEMO and random search show that DEMO outperforms random search on three datasets with regard to dominance ranking. With regard to the other three performance indicators, the differences are significant in all cases but one ($I^1_{\varepsilon+}$ on the dermatology dataset).

4.2.2 Experiments on the original problem

While the modified problem was needed for showing how close to the Pareto optimal front DEMO can get, what we are interested in is the original problem of finding the parameter settings of the J48 algorithm that would result in the best trees. Here, we explore the original decision space, presented in Table 4.2, and estimate the accuracy of trees using 10-fold cross validation. In these experiments, DEMO is compared only to random search and the J48 algorithm with default parameter values as exhaustive search of the space would be too time-consuming, even if the decision space was discretized.

Experimental setup and performance assessment

DEMO and random search used the same parameter settings as in the experiments on the modified problem. They were both ran 10 times on each of the datasets from Table 4.3. Their approximation sets were again assessed by plotting the best approximation sets; best, worst and 50%-attainment surfaces and by applying dominance ranking and the $I^1_{\varepsilon+}$, I_{H} and I^1_{R2} quality indicators to these approximation sets.

Results

Figure 4.3 shows the results of the best run of DEMO and random search (according to the I_{H} indicator) on the original optimization problem and the tree built using the default parameter values of J48. The best, worst and 50%-attainment surfaces for DEMO and random search can be found in Figure B.2 in Appendix B, while Table 4.5 presents the results of the statistical tests performed on dominance ranks and indicator values of approximation sets by DEMO and random search.

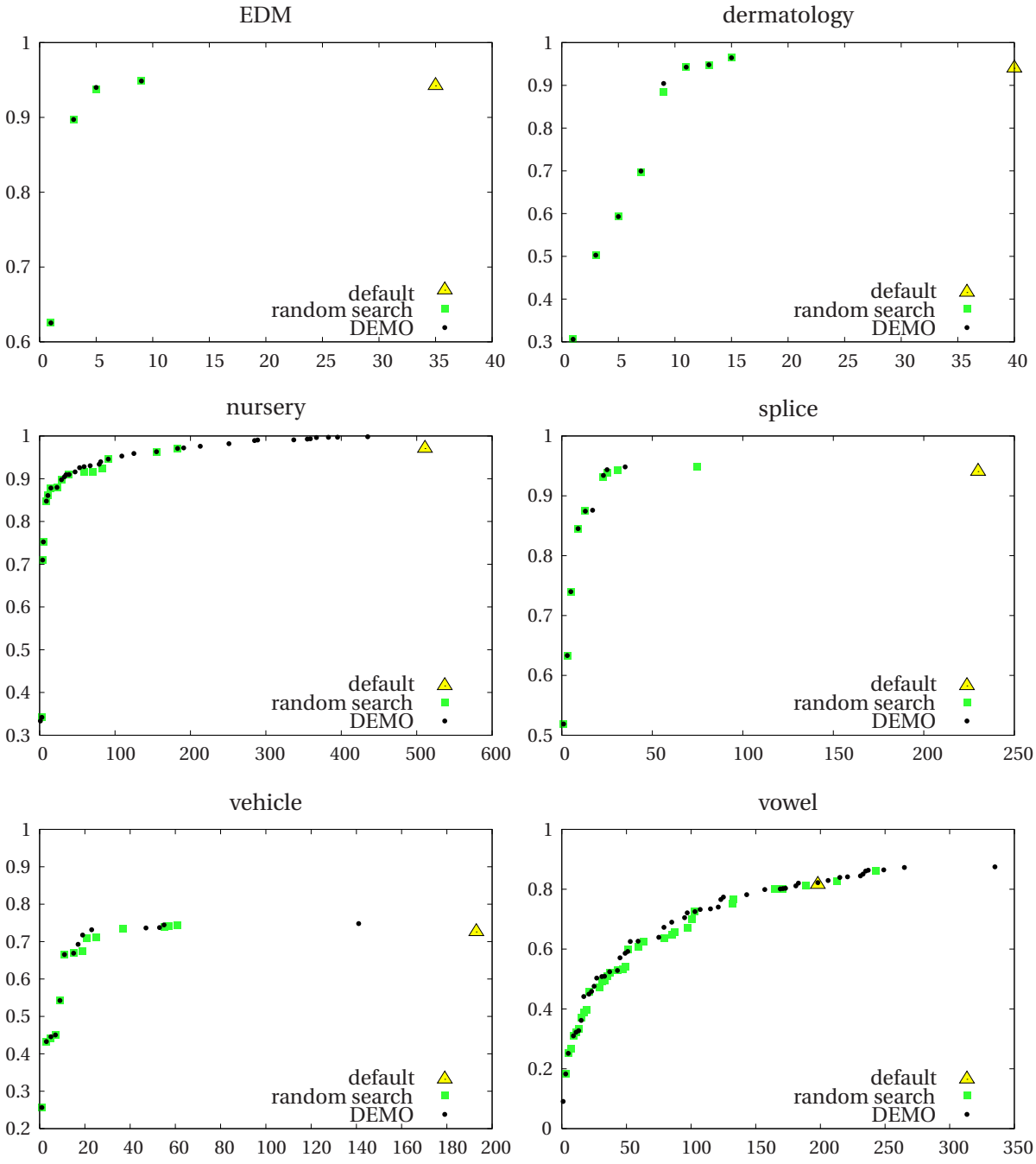


Figure 4.3: Objective values of trees found by the three algorithms on the original optimization problem for the six datasets. The size of trees is placed on the abscissa, while classification accuracy estimated using 10-fold cross validation is represented on the ordinate.

Table 4.5: Outcomes of the statistical tests ($\alpha = 0.05$, $\alpha_s = 0.0125$) on different performance measures for DEMO and random search on the original problem. The '▲ p-value' under a performance measure means that DEMO is significantly better than random search on this problem regarding the pertinent performance measure, while '-' indicates there are no significant differences between the two algorithms.

	dom_rank		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}	
EDM	-		-		-		-	
dermatology	▲	0.0027	▲	0.0017	▲	0.0007	-	
nursery	▲	0.0026	▲	$< 5.4 \times 10^{-6}$	▲	$< 5.4 \times 10^{-6}$	▲	$< 5.4 \times 10^{-6}$
splice	▲	0.0008	▲	0.0025	▲	0.0025	▲	0.0017
vehicle	-		▲	4×10^{-5}	▲	$< 5.4 \times 10^{-6}$	-	
vowel	-		▲	$< 5.4 \times 10^{-6}$	▲	0.0106	-	

Our first observation is that the approximation sets on this problem contain less solutions than the approximation sets on the modified problem. This is a logic consequence of using cross validation for accuracy estimation instead of evaluating accuracy on learning data. Namely, cross validation is able to penalize slightly pruned or unpruned trees for overfitting the data, which is visible as the absence of large trees in approximation sets for the original problem.

The comparison between the trees found by the two heuristic methods and the default tree shows that the default tree is often larger and less accurate than the other trees. On the dermatology, nursery, splice and vehicle datasets DEMO always finds trees that weakly dominate the default tree, while this happens in at least half of the runs on the EDM and vowel datasets. In some of the runs, even trees found by random search dominate the default tree. While DEMO outperforms random search with regard to some performance indicator on the dermatology, nursery, splice, vehicle and vowel datasets, there is no significant difference between the algorithms on the EDM dataset. This proves that it is indeed important for the users of the J48 algorithm to try some other parameter setting beside the default one when building a decision tree model.

4.2.3 Analysis of the decision space

In a single run of DEMO and random search on the original problem, 500 parameter settings were inspected by each algorithm. Since all experiments were repeated 10 times, the algorithms jointly built 10000 decision trees for each dataset. This gives us the possibility to look at the decision space of the original problem and see if there exist specific parameter

settings that induce good decision trees. To this end we gathered all 10000 decision trees for each dataset and denoted which of them are dominated and which are not. Since the decision space of the original problem is five-dimensional, it cannot be simply presented here. Therefore, we only show its projection on the two-dimensional space, defined by the parameters M (minimal number of instances in each leaf) and C (confidence factor used in postpruning), where the parameter U is set to *no* (only trees subject to postpruning are considered).

Inspecting the plots of dominated and nondominated trees in Figure 4.4 we can easily see that the parameter M has a big influence on the size and accuracy of the trees, while the effect of the parameter C seems to be much smaller. This causes the vertical 'stripes' of nondominated trees. When M is large, the trees are subject to heavy prepruning, which leaves little or no room for postpruning. This is why the 'stripes' are so well defined for large values of M. When M is smaller, the quality of trees depends also on the parameter C (see the EDM, vehicle and vowel datasets in Figure 4.4). Note that the quality of trees is always influenced also by the other three parameters (U, S and B), whose values are not shown in these plots.

The plots indicate that nondominated trees can be found at almost any point in the decision space and that their location depends very much on the dataset. Consequently, no general rule for predicting the optimal parameter values can be found. This gives additional evidence that searching for parameters of decision tree building algorithms has to be performed for every dataset separately.

It is important to realize that the decision space was not exhaustively explored, which means that there can exist other optimal trees that are not shown on these plots. Moreover, such optimal trees could dominate some of the trees that were here denoted as nondominated. While this would certainly change the appearance of the plots, the main conclusion would remain the same, i.e., the location of the optimal trees (in view of the two selected parameters) would still differ from dataset to dataset.

4.2.4 A real-world example

Finally, let us show how DEMO can help the users of machine learning algorithms in practice. Consider again the EDM dataset by Valentinčič and Junkar (2006), which was already used in the experiments from Subsections 4.2.1 and 4.2.2. This dataset consists of 467 instances described by 11 numeric attributes and two classes: 'keep' (175 instances) and 'select_lower'

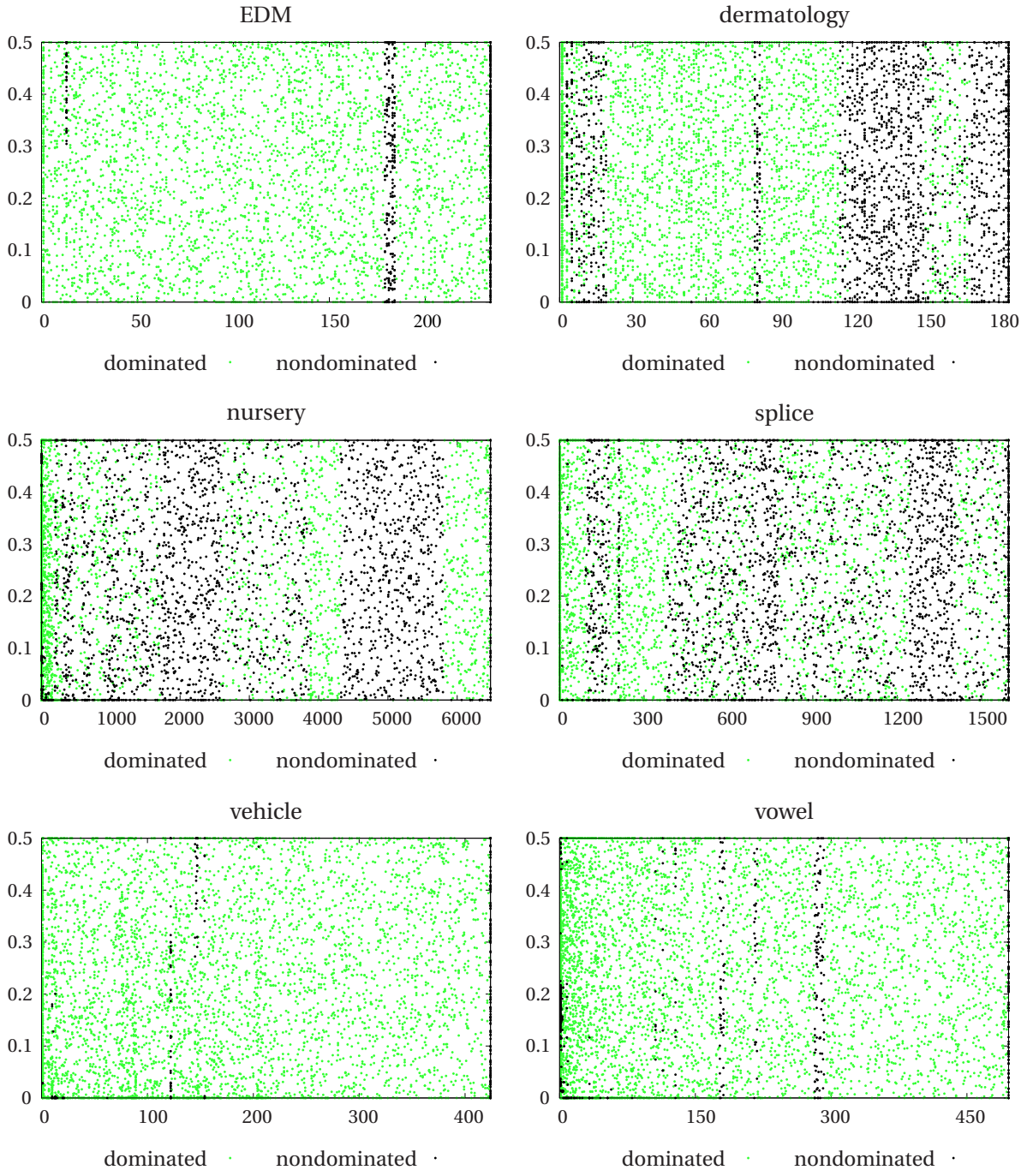


Figure 4.4: Dominated and nondominated trees found by DEMO and random search on the original problem for various values of parameters M (on the abscissa) and C (on the ordinate), while U is set to *no*.

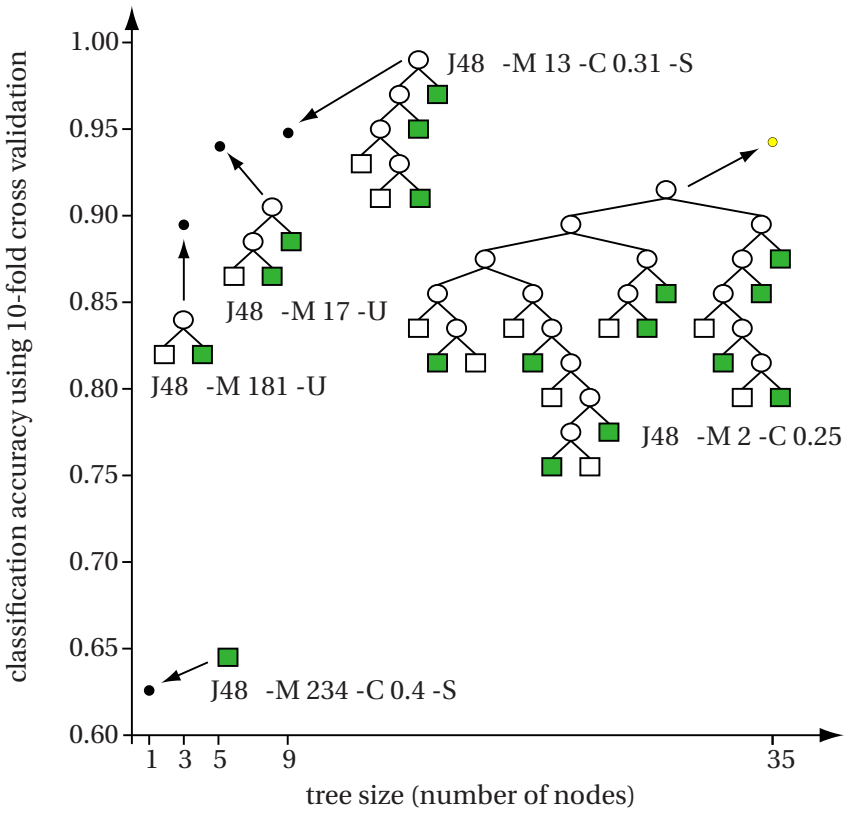


Figure 4.5: Decision trees found by DEMO (black points) and the default J48 tree (yellow point) for the EDM dataset. White leaves predict the class 'keep' and green leaves predict the class 'select_lower'.

(292 instances). The EDM dataset contains the results of experiments in the electrical discharge machining process, where the users need to choose from either keeping the machining parameters as they are ('keep') and selecting lower values ('select_lower'). Since the goal of the authors is to improve automation of EDM rough machining, the model describing this parameter setting problem must be simple.

After a run of DEMO, four mutually incomparable trees were found. These trees, together with the tree constructed using the default parameter values of J48, are presented in Figure 4.5. Each tree in the figure is additionally denoted with the parameter setting of J48 that were used to construct it.

From the set of the four best trees (we can discard the default tree since it is dominated by the largest tree found by DEMO), the users can select the preferred tree. For example, since in this case it is important for the tree to be small, the users might prefer to choose the tree of size 5 instead of the tree of size 9. While the larger tree is more accurate, the loss in

accuracy in quite small (less than 1 percentage point).

4.3 Summary

This chapter presented the problem of finding the parameter settings of algorithms for building decision trees that yield optimal trees—accurate and small. The results of the experiments performed on the original and modified optimization problems have shown that DEMO is capable of efficiently solving this problem, offering the users a wide choice of near-optimal decision trees with different accuracies and sizes in a reasonable time. The set of near-optimal trees helps the users to choose the tree that best suits their needs.

While this chapter was devoted exclusively to decision trees, DEMO could be employed in a similar way also for finding other models built using different machine learning methods with highest prediction accuracy and lowest complexity.

Chapter 5

Conclusion

We presented a multiobjective evolutionary algorithm called DEMO that searches the decision space using DE and is able to incorporate an arbitrary environmental selection procedure. As such, it can be used to compare the performance between GA-based and DE-based algorithms for multiobjective optimization. DEMO variants DEMO$^{\text{NS-II}}$, DEMO$^{\text{SP2}}$, DEMO$^{\text{IB}_{\varepsilon+}}$ and DEMO$^{\text{IB}_{\text{HD}}}$ were compared to their GA-based counterparts NSGA-II, SPEA2, IBEA$_{\varepsilon+}$ and IBEA$_{\text{HD}}$. Extensive experiments on 16 state-of-the-art benchmark problems with three different dimensions showed that on the majority of problems, DE-based algorithms outperform their GA-based equivalents with regard to the applied quality indicators. On the basis of these results we can conclude that DE in general explores the decision space more efficiently than GAs also when multiple objectives need to be optimized. It is important to note, however, that DE and DEMO are limited to vector representation of solutions and can therefore only be used in numerical optimization.

The results of additional comparison among DEMO's different approaches to environmental selection indicate that, in general, all variants achieve comparable convergence to the Pareto optimal front. However, the spread of solutions in the objective space varies from variant to variant. These differences are most visible on the problems with concave parts of Pareto optimal front, where the best distributed solutions (according to our visual perception of how well-distributed solutions look like) were achieved by DEMO$^{\text{SP2}}$. This result encouraged us to use the DEMO$^{\text{SP2}}$ variant in the considered case study.

The proposed real-world optimization problem consists of finding the parameter set-

tings of a machine learning algorithm that result in accurate and small decision trees. The results of the performed experiments on six real domains showed that DEMO is capable of efficiently solving this problem, offering the users a wide choice of near-optimal decision trees with different accuracies and sizes to choose from. Such an approach for searching accurate and simple theories could be extended to other machine learning algorithms.

Another direction for future work is to use DEMO for optimizing more than two objectives for the considered problem. In the case of decision trees, for example, the users might want to optimize also some other objective beside accuracy and size, such as the 'degree of interestingness' of the induced models, estimated in terms of presence or absence of some attributes in the models.

Finally, the presented work could be extended by considering other existing (or possibly new) approaches to environmental selection that were not covered by this thesis.

References

H. A. Abbass, R. Sarker, and C. Newton. PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, volume 2, pages 971–978, May 2001.

S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA – A platform and programming language independent interface for search algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 494–508, April 2003.

M. Bohanec and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilitá. *Pubblicazioni del Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

B. Cestnik and I. Bratko. On estimating probabilities in tree pruning. In *Proceedings of the European working session on learning on Machine Learning (EWSL '91)*, pages 138–150, March 1991.

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

W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, 1999.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

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

K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread Pareto-optimal solutions. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 222–236, April 2003.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. In A. Abraham, R. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 6, pages 105–145. Springer, 2005.

V. D. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 213–225, March 2001.

M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Institute of Mathematical Modelling, Technical University of Denmark, March 1998.

A. G. Hernández-Díaz, L. V. Santana-Quintero, C. Coello Coello, R. Caballero, and J. Molina. A new proposal for multi-objective optimization using differential evolution and rough sets theory. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, volume 1, pages 675–682, July 2006.

S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 280–295, March 2005.

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

J. Ilonen, J.-K. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 7(1):93–105, 2003.

A. W. Iorio and X. Li. Solving rotated multi-objective optimization problems using differential evolution. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004)*, pages 861–872, December 2004.

A. W. Iorio and X. Li. Incorporating directional information within a differential evolution algorithm for multi-objective optimization. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, volume 1, pages 675–682, July 2006.

H. Ishibuchi, S. Kaige, and K. Narukawa. Comparison between Lamarckian and Baldwinian repair on multiobjective 0/1 knapsack problems. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 370–385, March 2005.

J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Department of Computer Science, 2002.

J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report TIK-Report No. 214, Computer Engineering and Networks Laboratory, ETH Zürich, February 2006.

R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 304–312, July 1995.

T. Krink, B. Filipič, G. B. Fogel, and R. Thomsen. Noisy optimization problems – A particular challenge for differential evolution? In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, volume 1, pages 332–339, June 2004.

S. Kukkonen and J. Lampinen. An extension of generalized differential evolution for multi-objective optimization with constraints. In *Proceedings of Parallel Problem Solving from Nature (PPSN VIII)*, pages 752–761, September 2004.

S. Kukkonen and J. Lampinen. GDE3: The third evolution step of generalized differential evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, volume 1, pages 443–450, September 2005.

J. Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, July 2001.

N. K. Madavan. Multiobjective optimization using a Pareto differential evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, volume 2, pages 1145–1150, May 2002.

E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, volume 1, pages 485–492, July 2006.

D. Mladenić. Domain-tailored machine learning. Master's thesis, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, 1995.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

G. L. Pappa, A. A. Freitas, and C. A. A. Kaestner. Multi-objective algorithms for attribute selection in data mining. In C. A. Coello Coello and G. B. Lamont, editors, *Applications of Multi-Objective Evolutionary Algorithms*, chapter 25, pages 603–626. World Scientific, 2004.

K. E. Parsopoulos, D. K. Taoulis, N. G. Pavlidis, V. P. Plagianakos, and M .N. Vrahatis. Vector evaluated differential evolution for multiobjective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, volume 1, pages 204–211, June 2004.

K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer, 2005.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 520–533, March 2005.

T. Rogalsky, R. W. Derksen, and S. Kocabiyik. Differential evolution in aerodynamic optimization. In *Proceedings of the 46th Annual Conference of the Canadian Aeronautics and Space Institute*, pages 29–36, May 1999.

L. V. Santana-Quintero and C. A. Coello Coello. An algorithm based on differential evolution for multiobjective problems. In *Smart Engineering System Design: Neural Networks, Evolutionary Programming and Artificial Life*, volume 15, pages 211–220, November 2005.

J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

R. Storn and K. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

R. Thomsen. Flexible ligand docking using differential evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2354–2361, December 2003.

T. Tušar and B. Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 257–271, March 2007.

R. K. Ursem and P. Vadstrup. Parameter identification of induction motors using differential evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 2, pages 790–796, December 2003.

J. Valentinčič and M. Junkar. Detection of the eroding surface in the EDM process based on the current signal in the gap. *The International Journal of Advanced Manufacturing Technology*, 28(3-4):294–301, 2006.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

F. Xue, A. C. Sanderson, and R. J. Graves. Pareto-based multi-objective differential evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 2, pages 862–869, December 2003.

E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, September 2004.

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

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. In *Proceedings of Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2001)*, pages 95–100, September 2001.

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

Appendix

Abbreviations

The abbreviations used in this thesis can be divided into three groups. The first one consists of the most often used abbreviations, for which we also provide translations into Slovene:

DE	Differential Evolution Diferencialna evolucija
DEMO	Differential Evolution for Multiobjective Optimization Diferencialna evolucija za večkriterijsko optimiranje
EA	Evolutionary Algorithm Evolucijski algoritem
GA	Genetic Algorithm Genetski algoritem
MOEA	Multiobjective Evolutionary Algorithm Večkriterijski evolucijski algoritem
MOP	Multiobjective Optimization Problem Večkriterijski optimizacijski problem

The second group relates to the applied test problems:

EDM	Machine learning domain of Electrical Discharge Machining
DTLZ	Test problems by Deb, Thiele, Laumanns and Zitzler
WFG	Test problems by the Walking Fish Group

The last group contains the acronyms of the related algorithms:

DEMORS	Differential Evolution for Multiobjective Optimization with Rough Sets
GDE	Generalized Differential Evolution
IBEA	Indicator Based Evolutionary Algorithm
MODE	Multiobjective Differential Evolution
NSDE	Non-dominated Sorting Differential Evolution
NSGA	Nondominated Sorting Genetic Algorithm
OPT	Algorithm for finding the optimal sequence of pruned trees
PDE	Pareto-frontier Differential Evolution
PDEA	Pareto Differential Evolution Approach
SPEA	Strength Pareto Evolutionary Algorithm
VEDE	Vector Evaluated Differential Evolution
VEGA	Vector Evaluated Genetic Algorithm

Appendix B

Complete Results

Table B.1: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{NS-II}}$ and NSGA-II on the problems with two objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{NS-II}}$ is significantly better (worse) than NSGA-II regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=2$					
	$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}	
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG4	-		▲	1.2×10^{-4}	-	
WFG5	▽	$<8.5\times10^{-18}$	-		▽	$<8.5\times10^{-18}$
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	▲	0.0073	-		▲	2.0×10^{-5}
WFG9	-		▲	0.0016	-	

Table B.2: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{NS-II}}$ and NSGA-II on the problems with three objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{NS-II}}$ is significantly better (worse) than NSGA-II regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=3$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	0.0027	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	▲	0.0057	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	-		-		-	
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	0.0001	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	-		-		-	
WFG9	-		▲	3.2×10^{-4}	▲	$<8.5\times10^{-18}$

Table B.3: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{NS-II}}$ and NSGA-II on the problems with four objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{NS-II}}$ is significantly better (worse) than NSGA-II regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=4$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	-		▽	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	0.0085	▲	0.0004	▲	0.0018
WFG3	-		▽	8.2×10^{-4}	▽	$<8.5\times10^{-18}$
WFG4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	-		-		▲	$<8.5\times10^{-18}$
WFG6	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	-		▲	$<8.5\times10^{-18}$	-	
WFG8	-		-		▽	$<8.5\times10^{-18}$
WFG9	-		-		-	

Table B.4: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO^{SP2} and SPEA2 on the problems with two objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO^{SP2} is significantly better (worse) than SPEA2 regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=2$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG4	-		▲	0.0026	-	
WFG5	▽	$<8.5\times10^{-18}$	▽	$<8.5\times10^{-18}$	▽	$<8.5\times10^{-18}$
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	▲	$<8.5\times10^{-18}$	-		▲	$<8.5\times10^{-18}$
WFG9	-		-		-	

Table B.5: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO^{SP2} and SPEA2 on the problems with three objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO^{SP2} is significantly better (worse) than SPEA2 regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=3$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▽	0.0049	-		-	
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	-		-		-	
WFG4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	▲	0.0001	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	▲	4.0×10^{-5}	▲	2.0×10^{-5}	▲	$<8.5\times10^{-18}$
WFG9	▲	0.0053	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$

Table B.6: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{SP2}}$ and SPEA2 on the problems with four objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{SP2}}$ is significantly better (worse) than SPEA2 regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=4$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ4	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	▽	6.4×10^{-4}	-		▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	0.0007	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	▽	0.0095	-		-	
WFG4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	▲	0.0022	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG6	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	0.0111	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG9	-		▲	1.2×10^{-4}	▲	2.6×10^{-4}

Table B.7: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$ on the problems with two objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{IB}_{\varepsilon+}}$ is significantly better (worse) than IBEA$_{\varepsilon+}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=2$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	-		-		▲	$<8.5\times10^{-18}$
DTLZ3	▽	$<8.5\times10^{-18}$	▽	$<8.5\times10^{-18}$	▽	$<8.5\times10^{-18}$
DTLZ4	-		-		▲	$<8.5\times10^{-18}$
DTLZ5	-		-		▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	-		-		▲	4.0×10^{-5}
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	▲	0.0010	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG4	▽	$<8.5\times10^{-18}$	▽	0.0053	▽	5.2×10^{-4}
WFG5	▽	$<8.5\times10^{-18}$	▽	1.4×10^{-4}	▽	$<8.5\times10^{-18}$
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	-		-		-	
WFG9	-		▲	0.0007	-	

Table B.8: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$ on the problems with three objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{IB}_{\varepsilon+}}$ is significantly better (worse) than IBEA$_{\varepsilon+}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=3$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ2	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	0.0028
DTLZ4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	-		-		▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	▽	0.0013	-		▲	3.6×10^{-4}
WFG1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	-		-		-	
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	-		-		▲	$<8.5\times10^{-18}$
WFG9	▲	0.0003	-		▲	$<8.5\times10^{-18}$

Table B.9: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{IB}_{\varepsilon+}}$ and IBEA$_{\varepsilon+}$ on the problems with four objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{IB}_{\varepsilon+}}$ is significantly better (worse) than IBEA$_{\varepsilon+}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m=4$					
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	-	
DTLZ2	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ3	▲	$<8.5\times10^{-18}$	▲	0.0122	▲	0.0002
DTLZ4	▲	0.0038	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ5	-		▽	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
DTLZ7	-		-		-	
WFG1	▲	$<8.5\times10^{-18}$	▲	1.2×10^{-4}	▲	0.0016
WFG2	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG3	▽	4.0×10^{-5}	▲	0.0004	▽	9.6×10^{-4}
WFG4	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG5	-		▲	4.0×10^{-5}	▲	8.0×10^{-5}
WFG6	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG7	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$
WFG8	-		-		▲	$<8.5\times10^{-18}$
WFG9	-		▲	$<8.5\times10^{-18}$	▲	$<8.5\times10^{-18}$

Table B.10: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{IB}_{\text{HD}}}$ and IBEA$_{\text{HD}}$ on the problems with two objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{IB}_{\text{HD}}}$ is significantly better (worse) than IBEA$_{\text{HD}}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

			$m = 2$			
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ2	-		-		▲	$< 8.5 \times 10^{-18}$
DTLZ3	-		▽	0.0077	▽	0.0103
DTLZ4	▲	4.4×10^{-4}	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ5	-		-		▲	$< 8.5 \times 10^{-18}$
DTLZ6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ7	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$
WFG1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG2	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG3	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG4	-		▽	0.0107	▽	0.0028
WFG5	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$
WFG6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG7	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG8	-		-		-	
WFG9	▲	0.0092	▲	0.0006	-	

Table B.11: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for DEMO$^{\text{IB}_{\text{HD}}}$ and IBEA$_{\text{HD}}$ on the problems with three objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which DEMO$^{\text{IB}_{\text{HD}}}$ is significantly better (worse) than IBEA$_{\text{HD}}$ regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

			$m = 3$			
		$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}
DTLZ1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ2	-		▲	0.0107	▲	$< 8.5 \times 10^{-18}$
DTLZ3	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ4	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ5	▽	0.0016	▽	0.0039	▲	$< 8.5 \times 10^{-18}$
DTLZ6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ7	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$	▽	2.0×10^{-5}
WFG1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG2	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG3	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG4	-		▲	2.8×10^{-4}	▲	$< 8.5 \times 10^{-18}$
WFG5	-		-		▲	6.6×10^{-4}
WFG6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG7	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG8	-		-		▽	7.4×10^{-4}
WFG9	-		▲	0.0067	▲	$< 8.5 \times 10^{-18}$

Table B.12: Outcomes of the Fisher-independent test ($\alpha = 0.05$ and $\alpha_s = 0.0125$) for $\text{DEMO}^{\text{IB}_{\text{HD}}}$ and IBEA_{HD} on the problems with four objectives. The '▲ p-value' ('▽ p-value') under the indicator I denotes the problems, on which $\text{DEMO}^{\text{IB}_{\text{HD}}}$ is significantly better (worse) than IBEA_{HD} regarding indicator I, while '-' indicates there are no significant differences between the two algorithms regarding indicator I.

	$m = 4$					
	$I^1_{\varepsilon+}$		I_{H}		I^1_{R2}	
DTLZ1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	-	
DTLZ2	-		-		▲	$< 8.5 \times 10^{-18}$
DTLZ3	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ4	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ5	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
DTLZ7	▽	7.6×10^{-4}	▽	0.0032	-	
WFG1	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG2	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG3	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$	▽	$< 8.5 \times 10^{-18}$
WFG4	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG5	-		▽	2.0×10^{-5}	▲	2.0×10^{-5}
WFG6	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG7	-		▲	$< 8.5 \times 10^{-18}$	▲	$< 8.5 \times 10^{-18}$
WFG8	-		-		▽	8.6×10^{-4}
WFG9	-		-		▲	$< 8.5 \times 10^{-18}$

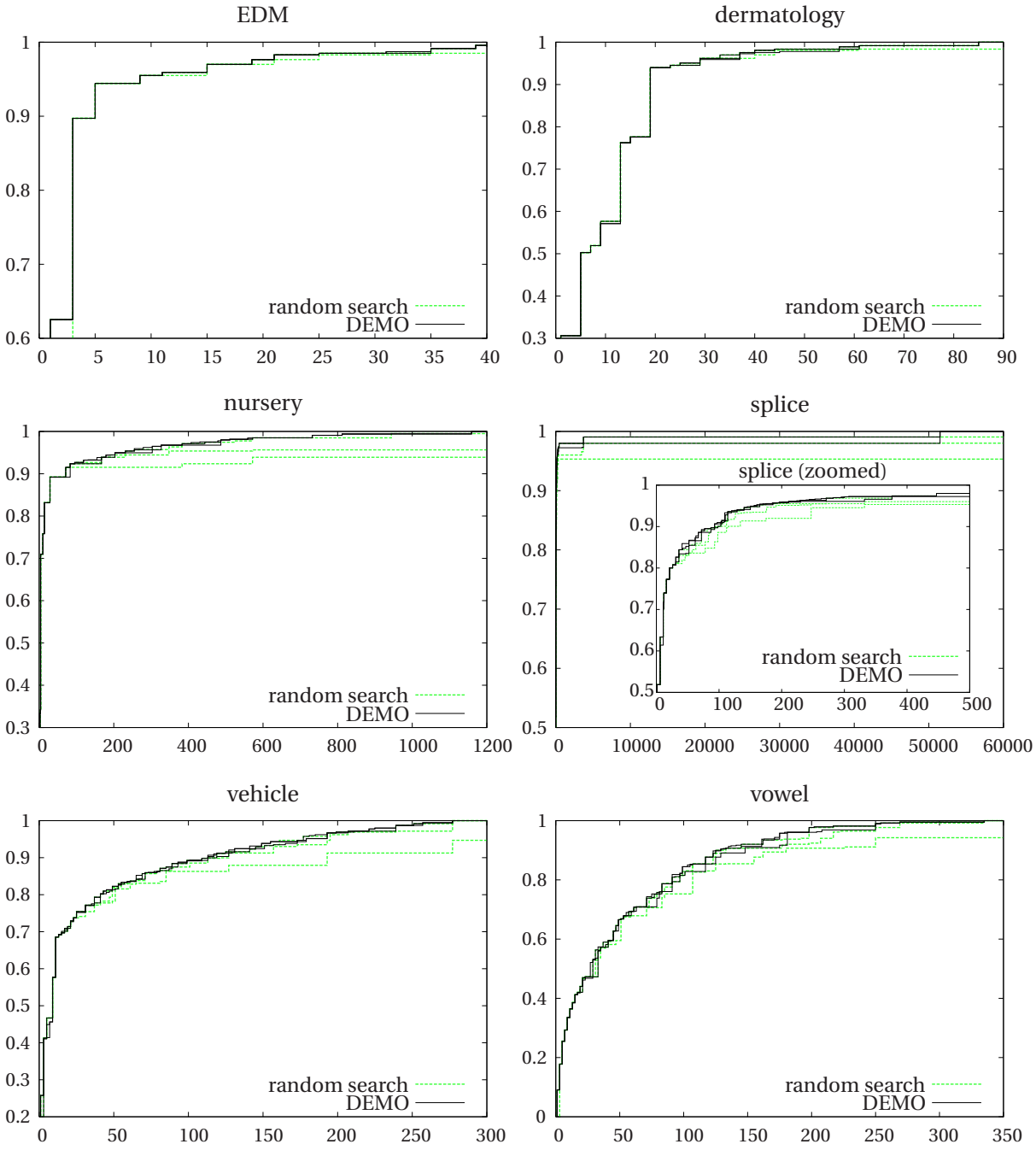


Figure B.1: Best, worst and 50% attainment surfaces of DEMO and random search on the modified optimization problem for the six datasets. The size of trees is placed on the abscissa, while classification accuracy estimated estimated on the training instances is represented on the ordinate.

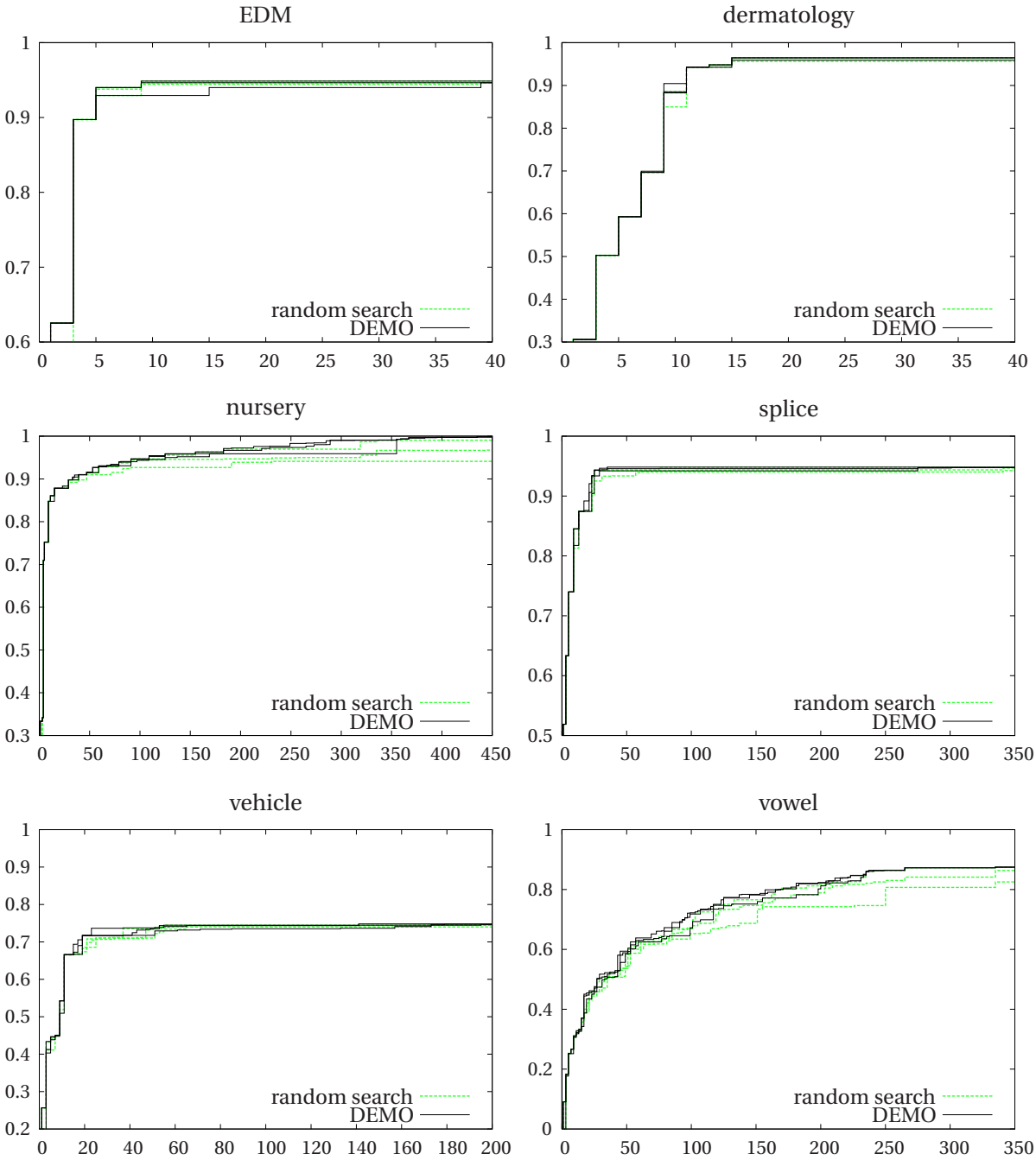


Figure B.2: Best, worst and 50%-attainment surfaces of DEMO and random search on the original optimization problem for the six datasets. The size of trees is placed on the abscissa, while classification accuracy estimated using 10-fold cross validation is represented on the ordinate.

Appendix C

Razširjen povzetek v slovenskem jeziku

C.1 Uvod

V praksi se pogosto srečujemo z zahtevo po sočasnem optimiranju po več kriterijih. V primeru, ko si kriteriji nasprotujejo, nimamo opravka samo z eno optimalno rešitvijo, temveč z množico t. i. *Pareto optimalnih rešitev*, kjer vsaka rešitev predstavlja nek kompromis med kriteriji. Brez dodatne informacije o prednosti kriterijev tako ne moremo trditi, da je katera izmed teh rešitev boljša od druge. Zato pogosto želimo poiskati čim več Pareto optimalnih rešitev hkrati – v enem samem zagonu algoritma. To nam omogočajo evolucijskimi algoritmi, ki so populacijska preiskovalna metoda in zato lahko sočasno in po več kriterijih optimirajo množico rešitev.

Pri večkriterijskem optimiranju obstaja poleg *prostora spremenljivk*, kjer iščemo rešitve, še *prostor kriterijev*, kjer dobljene rešitve vrednotimo. Evolucijski algoritmi, ki se uporabljajo za večkriterijsko optimiranje (t. i. večkriterijski evolucijski algoritmi), zato delujejo na dveh nivojih: uporabljajo neko *preiskovalno metodo* za usmerjanje iskanja rešitev v prostoru spremenljivk in *metodo za kriterijsko selekcijo*, s katero izberejo najboljše rešitve glede na njihov položaj v prostoru kriterijev. Čeprav je bilo v zadnjih dvajsetih letih predstavljenih mnogo večkriterijskih evolucijskih algoritmov, večina izmed njih – tudi popularni NSGA-II (Deb in sod., 2002), SPEA2 (Zitzler in sod., 2001) in IBEA (Zitzler in Künzli, 2004) – uporablja isto preiskovalno metodo (genetski algoritem) in se razlikuje le v uporabljenem pristopu za kri-

terijsko selekcijo.

V magistrski nalogi predstavljamo algoritem DEMO (angl. Diferential Evolution for Multiobjective Optimization), ki lahko poljuben pristop za kriterijsko selekcijo kombinira z diferencialno evolucijo (Storn in Price, 1997) – preiskovalno metodo, ki v reševanju enokriterijskih problemov pogosto doseže boljše rezultate kot genetski algoritmi. V pričujoči magistrski nalogi smo želeli raziskati ali to drži tudi za večkriterijsko optimiranje. Poleg tega želimo algoritem DEMO uporabiti na realnem primeru optimiranja točnosti in velikosti odločitvenih dreves.

C.2 Posebnosti večkriterijskega optimiranja

C.2.1 Osnovne definicije

Večkriterijski optimizacijski problem je definiran kot problem iskanja optimuma funkcije

$$\begin{aligned} \boldsymbol{f} &: X \to Z \\ \boldsymbol{f} &: (x_1, \ldots, x_n) \mapsto (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)), \end{aligned}$$

kjer je X n-dimenzionalni prostor spremenljivk in Z m-dimenzionalni prostor kriterijev (za $m \geq 2$).

Definicija C.1 (**Pareto dominiranost vektorjev**)**.** Vektor $\boldsymbol{z}^1$ *dominira* vektor $\boldsymbol{z}^2$ ($\boldsymbol{z}^1 \prec \boldsymbol{z}^2$) $\overset{def}{\iff}$ $z_j^1 \leq z_j^2$ za vse $j \in \{1, \ldots, m\}$ in $z_k^1 < z_k^2$ vsaj za en $k \in \{1, \ldots, m\}$.

Za različna vektorja $\boldsymbol{z}^1$ in $\boldsymbol{z}^2$ v prostoru kriterijev lahko velja, da $\boldsymbol{z}^1 \not\prec \boldsymbol{z}^2$ in $\boldsymbol{z}^2 \not\prec \boldsymbol{z}^1$. Takrat pravimo, da sta $\boldsymbol{z}^1$ in $\boldsymbol{z}^2$ *neprimerljiva*.

Definicija C.2 (**Pareto optimalnost**)**.** Rešitev $\boldsymbol{x}^*$ in njej pripadajoči vektor v prostoru kriterijev $\boldsymbol{z}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ sta *Pareto optimalna* $\overset{def}{\iff}$ ne obstaja $\boldsymbol{z} \in Z$, tako da $\boldsymbol{z} \prec \boldsymbol{z}^*$.

Vsi Pareto optimalni vektorji v prostoru kriterijev sestavljajo *Pareto optimalno fronto*. Vsak vektor iz Pareto optimalne fronte predstavlja kompromis med kriteriji. Brez dodatne informacije o pomembnosti posameznih kriterijev so vse Pareto optimalne rešitve enakovredne.

C.2.2 Dva pristopa

Podobno kot v enokriterijskem optimiranju, je tudi v večkriterijskem končni cilj najti eno samo optimalno rešitev. Dobimo jo z uporabo bodisi prednostnega bodisi idealnega pristopa.

Po *prednostnemu pristopu* večkriterijski optimizacijski problem na začetku prevedemo na enokriterijskega tako, da kriterijem dodelimo prednosti. To lahko dosežemo na primer z uporabo utežene vsote kriterijev. Eno samo rešitev potem poiščemo tako, da rešimo dobljeni enokriterijski optimizacijski problem. Uporaba prednostnega pristopa takrat, ko prednosti kriterijev ne poznamo vnaprej, ima več slabosti. Rešitev, ki jo dobimo na ta način, je odvisna od funkcije, ki je bila uporabljena za transformacijo več kriterijev v enega. Z drugačno funkcijo lahko dobimo drugačno rešitev. Poleg tega pri nekaterih najpogosteje uporabljenih transformacijah (kot je na primer utežena vsota) ni mogoče najti vektorjev, ki ležijo na konkavnih delih Pareto optimalne fronte.

Z *idealnim pristopom* pa najprej rešimo večkriterijski problem in šele nato uporabimo informacijo o prednosti kriterijev, da izberemo eno samo rešitev iz množice optimalnih. Idealni pristop od uporabnika ne zahteva poznavanja prednosti kriterijev pred optimizacijo. Šele ko je znanih več kompromisnih rešitev, uporabnik potrebuje poznavanje prednosti kriterijev, da se odloči za eno izmed njih.

C.2.3 Aproksimacijske množice

Rezultat algoritma, ki rešuje večkriterijske optimizacijske probleme po idealnem pristopu, je navadno množica nedominiranih rešitev. Kriterijski vektorji teh rešitev sestavljajo t. i. *aproksimacijsko množico*, ki aproksimira Pareto optimalno fronto.

Da bi uporabniku omogočili čim boljšo izbiro, algoritmi za večkriterijsko optimizacijo pogosto iščejo aproksimacijske množice, v katerih so vektorji čim bolj raznoliki in enakomerno razporejeni vzdolž Pareto optimalne fronte. Čeprav je to v praktičnih problemih mnogokrat zaželena lastnost algoritmov, ni skladna z relacijo Pareto dominiranosti. To pomeni, da je doseganje vektorjev v bližini Pareto optimalne fronte edini cilj večkriterijskega optimiranja, medtem ko je enakomerna razporeditev vektorjev samo zaželena lastnost in ne more biti formalno šteta kot enakovredni drugi cilj večkriterijskega optimiranja. To nasprotuje konceptu dveh ciljev večkriterijskega optimiranja, ki ga lahko pogosto najdemo v literaturi (Deb, 2001).

C.3 Diferencialna evolucija za večkriterijsko optimiranje

Diferencialna evolucija je evolucijski algoritem, ki uporablja vektorsko predstavitev rešitev in nove rešitve gradi s pomočjo vektorskega seštevanja, množenja s skalarjem in križanja. Vsaka tako dobljena nova rešitev, imenovana tudi kandidat, je sprejeta v populacijo le, če je enakovredna ali boljša od svojega starša. Diferencialna evolucija na številnih enokriterijskih optimizacijskih problemih dosega boljše rezultate kot genetski algoritmi (Price in sod., 2005). Zato so se v zadnjih letih pojavili različni algoritmi za večkriterijsko optimiranje, ki temeljijo na diferencialni evoluciji (Abbass in sod., 2001; Lampinen, 2001; Madavan, 2002; Xue in sod., 2003; Parsopoulos in sod., 2004; Kukkonen in Lampinen, 2004). Vendar pa so ti algoritmi bodisi zaobšli primerjanje med kandidati in starši bodisi je bila primerjava prestroga za uspešno uporabo pri več kriterijih. Poleg tega je bil edini uporabljeni pristop za kriterijsko selekcijo v teh algoritmih kombinacija nedominiranega sortiranja in metrike nakopičenosti (kot pri NSGA-II).

V magistrski nalogi predstavljamo algoritem DEMO, ki preiskuje prostor spremenljivk z diferencialno evolucijo in lahko za izbiro najboljših rešitev za naslednjo generacijo uporablja poljuben pristop za kriterijsko selekcijo.

C.3.1 Algoritem DEMO

DEMO tvori novega kandidata na enak način kot diferencialna evolucija. Primerjavo med kandidatom in njegovim staršem izvede po naslednjem postopku. Če kandidat dominira starša, potem ga zamenja v populaciji. Če je kandidat dominiran s strani starša, potem kandidata zavržemo. Če pa sta med seboj neprimerljiva, DEMO kandidata doda v populacijo. Ko se ta postopek ponovi za vse kandidate v trenutni generaciji, se je populacija lahko povečala. V takem primeru jo moramo zmanjšati na prvotno velikost, kar naredimo s pomočjo poljubnega pristopa za kriterijsko selekcijo. V magistrski nalogi predstavljamo štiri različice algoritma DEMO, označene z $\text{DEMO}^{\text{NS-II}}$, DEMO^{SP2}, $\text{DEMO}^{\text{IB}_{\varepsilon+}}$ in $\text{DEMO}^{\text{IB}_{\text{HD}}}$, ki uporabljajo enake pristope za kriterijsko selekcijo kot algoritmi NSGA-II, SPEA2, $\text{IBEA}_{\varepsilon+}$ in IBEA_{HD}.

Največja prednost diferencialne evolucije in algoritma DEMO je v njuni enostavnosti za razumevanje in implementacijo, največja slabost pa omejenost uporabe na numerične optimizacijske probleme. Rešitve morajo biti zaradi vektorskih operacij namreč zapisane kot vektorji, tega pa za kombinatorične optimizacijske probleme ne moremo narediti.

C.3.2 Primerjava algoritma DEMO z večkriterijskimi genetskimi algoritmi

Najprej želimo primerjati diferencialno evolucijo in genetske algoritme kot dve preiskovalni metodi za večkriterijsko optimiranje neodvisno od uporabljenega pristopa za kriterijsko selekcijo. Zato med sabo paroma primerjamo algoritme DEMO$^{\text{NS-II}}$ in NSGA-II, DEMO$^{\text{SP2}}$ in SPEA2, DEMO$^{\text{IB}_{\varepsilon+}}$ in IBEA$_{\varepsilon+}$ ter DEMO$^{\text{IB}_{\text{HD}}}$ in IBEA$_{\text{HD}}$.

Vse algoritme smo pognali 30-krat na 16 testnih problemih (uporabili smo 7 problemov DTLZ (Deb in sod., 2005) in 9 problemov WFG (Huband in sod., 2005)) z dvema, tremi in štirimi kriteriji. Algoritme smo po priporočilu (Knowles in sod., 2006) ocenili na podlagi štirih mer: rangiranja po dominiranosti (Knowles in sod., 2006) in indikatorjev kakovosti $I^1_{\varepsilon+}$ (Zitzler in sod., 2003), I_{H} (Zitzler in Thiele, 1999) ter I^1_{R2} (Hansen in Jaszkiewicz, 1998). Signifikantnost dobljenih rezultatov smo preverili s statističnimi testi. Poleg tega smo aproksimacijske množice algoritmov primerjali še glede na njihove površine dosega (Grunert da Fonseca in sod., 2001).

Rezultati testov rangiranja po dominiranosti so pokazali, da je DEMO v 19% vseh primerov signifikantno boljši in le v 2% primerov signifikantno slabši od pristopov, ki uporabljajo genetske algoritme. Če opazujemo indikatorje kakovosti, pa je DEMO signifikantno boljši kar v 81% in signifikantno slabši le v 9% vseh primerov. Iz teh rezultatov lahko sklepamo, da diferencialna evolucija preiskuje prostor spremenljivk učinkoviteje kot genetski algoritmi.

C.3.3 Primerjava različic algoritma DEMO

Ker bi se radi odločili za različico algoritma DEMO, ki najbolj ustreza našim zahtevam, smo naredili dodatno primerjavo med vsemi štirimi različicami algoritma. Zanjo smo uporabili rezultate prejšnjih poskusov, tako da smo jih ponovno rangirali po dominiranosti. Posamezne različice algoritma DEMO so se statistično signifikantno razlikovale samo na nekaterih primerih, kjer sta DEMO$^{\text{IB}_{\varepsilon+}}$ in DEMO$^{\text{IB}_{\text{HD}}}$ praviloma dosegla slabše rezultate kot DEMO$^{\text{NS-II}}$ in DEMO$^{\text{SP2}}$.

Ker je edina razlika med različicami algoritma DEMO v uporabljenem pristopu za kriterijsko selekcijo, smo različice primerjali tudi glede dobljene razporeditve vektorjev v prostoru kriterijev. Prikazi aproksimacijskih množic na primerih z dvema in tremi kriteriji so razkrili, da algoritem DEMO$^{\text{SP2}}$ doseže razporeditev vektorjev, ki se najbolje ujema z intuitivno predstavo o dobri razporejenosti. Zato smo se odločili, da bomo različico DEMO$^{\text{SP2}}$ uporabili za reševanje praktičnega problema optimiranja točnosti in velikosti odločitvenih dreves.

C.4 Optimiranje točnosti in velikosti odločitvenih dreves

Algoritem DEMO želimo uporabiti v strojnem učenju za iskanje parametrov učnih algoritmov z namenom, da bodo dobljene teorije čim bolj točne in enostavne. S tem želimo olajšati delo uporabnikom, ki se ne spoznajo na algoritme strojnega učenja in ne vedo, kako nastaviti parametre teh algoritmov, da bi dobili primerne teorije. Poleg tega uporabniki mnogokrat vnaprej sploh ne vedo, kakšno teorijo iščejo. DEMO jim pri tem lahko pomaga tako, da preišče prostor parametrov učnega algoritma in uporabniku vrne množico (skoraj) optimalnih teorij, ki se razlikujejo v kompromisu med točnostjo in enostavnostjo. Na ta način uporabnik dobi boljši vpogled v svojo problemsko domeno in ima na voljo več teorij, med katerimi lahko izbere najustreznejšo.

C.4.1 Opis problema

V magistrski nalogi se omejimo na algoritem za gradnjo odločitvenih dreves C4.5 (Quinlan, 1986), oz. na njegovo implementacijo v okolju Weka, imenovano J48 (Witten in Frank, 2005). Pri optimiranju klasifikacijske točnosti in velikosti dobljenih odločitvenih dreves preiskujemo prostor naslednjih parametrov algoritma J48:

- najmanjše število primerov v listih (M),
- gradnja izključno neporezanih dreves (U),
- faktor zaupanja pri naknadnem rezanju dreves (C),
- dvigovanje poddreves pri naknadnem rezanju (S),
- gradnja izključno dvojiških dreves (B).

C.4.2 Optimiranje z algoritmom DEMO

Ker je na velikih domenah gradnja odločitvenih dreves lahko časovno potratna, postopek preiskovanja omejimo tako, da pregleda le 500 rešitev. Kot rečeno, smo se problema lotili z različico DEMO^{SP2}.

Za optimiranje točnosti in velikosti odločitvenih dreves smo opravili dve seriji poskusov. V prvi smo želeli videti, kako blizu Pareto optimalne fronte ležijo vektorji, ki jih dobimo z algoritmom DEMO^{SP2}. V ta namen smo okrnili originalni problem tako, da smo DEMO^{SP2} lahko primerjali z izčrpnim preiskovanjem, algoritmom OPT (Bohanec in Bratko,

1994), ki iz danega drevesa dobi vsa optimalna rezana poddrevesa, in naključnim preiskovanjem. Vse algoritme smo preizkusili na šestih domenah strojnega učenja (domeni EDM (Valentinčič in Junkar, 2006) in petih domenah iz zbirke UCI Machine Learning Repository (Newman in sod., 1998)).

Na okrnjenem problemu se je DEMO^{SP2} izkazal zelo dobro. Podobno kot naključno preiskovanje je našel mnogo vektorjev na Pareto optimalni fronti, ki pa so bili veliko bolje razporejeni. Statistični testi so potrdili, da so rezultati algoritma DEMO^{SP2} signifikantno boljši od rezultatov dobljenih z naključnim preiskovanjem.

Vendar smo okrnjeni problem uporabili le zato, ker nam je omogočil primerjavo s Pareto optimalno fronto. Naš cilj je uporabiti algoritem DEMO za reševanje originalnega problema. Zato smo DEMO^{SP2} in naključno preiskovanje preizkusili tudi na tem problemu. Poleg tega nas je zanimalo, kakšna so dobljena drevesa v primerjavi z drevesom, ki ga dobimo, če za algoritem J48 uporabimo privzeto nastavitev parametrov.

Na domeni EDM, kjer je malo optimalnih dreves, se rezultati algoritma DEMO^{SP2} in naključnega preiskovanja ne razlikujejo signifikantno. Po drugi strani pa so razlike na ostalih problemih med algoritmom DEMO^{SP2} in naključnim preiskovanjem signifikantne. Opazimo tudi, da so drevesa, zgrajena s privzetimi parametri algoritma J48, navadno večja od dreves, ki jih dobimo z algoritmom DEMO^{SP2} in naključnim preiskovanjem. Drevesa, dobljena z algoritmom DEMO^{SP2}, večinoma dominirajo privzeta drevesa.

Dodaten vpogled v prostor spremenljivk je pokazal, da ni pravila, ki bi lahko za poljubno domeno napovedalo, kakšne nastavitve parametrov bodo dale optimalna drevesa. Te nastavitve so namreč v veliki meri odvisne od izbrane domene.

C.5 Sklep

Predstavili smo algoritem DEMO, ki prostor spremenljivk preiskuje z diferencialno evolucijo in izbira najboljše rešitve s poljubnim pristopom za kriterijsko selekcijo. DEMO smo implementirali v štirih različicah (DEMO$^{NS\text{-}II}$, DEMO^{SP2}, DEMO$^{IB_{\varepsilon+}}$ in DEMO$^{IB_{HD}}$) in jih paroma primerjali z algoritmi z enakim pristopom za kriterijsko selekcijo, ki za preiskovanje uporabljajo genetske algoritme. Primerjava na 16-ih testnih večkriterijskih optimizacijskih problemih je pokazala, da na obravnavanih problemih diferencialna evolucija bolje preišče prostor spremenljivk kot genetski algoritmi. Žal pa sta tako diferencialna evolucija kot DEMO zaradi vektorske predstavitve rešitev primerna le za reševanje numeričnih optimizacijskih problemov.

Dodatna primerjava med štirimi različicami algoritma DEMO je pokazala, da različica DEMO^{SP2} doseže razporeditev vektorjev v prostoru kriterijev, ki uporabniku običajno najbolj ustreza. Zato smo to različico uporabili za reševanje realnega problema optimizacije točnosti in velikosti odločitvenih dreves. Na tem problemu je DEMO našel dobre kompromise med točnimi in majhnimi drevesi, med katerimi lahko uporabnik izbira najbolj zaželenega.

Na podoben način bi DEMO lahko uporabili tudi za optimiranje teorij, grajenih z drugimi algoritmi strojnega učenja. Poleg tega bi lahko h kriterijema točnosti in velikosti lahko dodali še kakšen dodaten kriterij, na primer 'zanimivost' zgrajene teorije, ki bi jo lahko merili s prisotnostjo ali odsotnostjo določenih atributov v teoriji.

Izjava o avtorstvu

Spodaj podpisana Tea Tušar izjavljam, da sem avtorica magistrske naloge z naslovom *Razvoj algoritma za večkriterijsko optimiranje z diferencialno evolucijo*, oziroma angleškim naslovom *Design of an Algorithm for Multiobjective Optimization with Differential Evolution*. Magistrsko nalogo sem izdelala samostojno pod mentorstvom akad. prof. dr. Ivana Bratka in somentorstvom doc. dr. Bogdana Filipiča. Izkazano pomoč drugih sodelavcev sem v celoti navedla v zahvali.

Tea Tušar, univ. dipl. mat.