Eero Hakavuori

Objective reduction in multiobjective optimization

Minor Subject Thesis in Information Technology

February 18, 2015

University of Jyväskylä

Department of Mathematical Information Technology

Author: Eero Hakavuori

Contact information: eero.j.hakavuori@jyu.fi

Supervisors: Jussi Hakanen, and Markus Hartikainen

Title: Objective reduction in multiobjective optimization

Työn nimi: Tavoitteiden vähentäminen monitavoiteoptimoinnissa

Project: Minor Subject Thesis

Study line: Computational science

Page count: 41+4

Abstract: The aim of this thesis is to study methods that have been created to avoid some of the problems related to solving multiobjective optimization problems with a large number of objectives. Multiple methods are presented covering various assumptions on the optimization problem, such as linearity or convexity, and the strengths and weaknesses of the methods are discussed. One of the methods is looked at in a more practical fashion, by presenting a Python code implementation of the abstract algorithm of the method in question and studying its behavior for some examples. Additionally, some criteria for classifying methods of objective reduction in multiobjective optimization are defined.

Keywords: multiobjective optimization, Pareto optimality, redundant objectives, objective reduction

Suomenkielinen tiivistelmä: Tämän tutkielman tavoitteena on tarkastella menetelmiä, jotka pyrkivät ratkaisemaan ongelmia, jotka ilmenevät monitavoiteoptimointitehtävissä tavoitteiden määrän kasvaessa suureksi. Työssä esitellään useita menetelmiä kattaen erilaisia oletuksia optimointitehtävän luonteelta, kuten esimerkiksi lineaaristen tai konveksien tehtävien tapaukset. Lisäksi kommentoidaan menetelmien vahvuuksia ja heikkouksia. Yksi menetelmistä otetaan käytännönläheisempään tarkasteluun toteuttamalla siihen liittyvä abstrakti algoritmi Python-koodina ja tarkastelemalla algoritmin käyttäytymistä esimerkkien avulla. Lisäksi esitetään muutamia tapoja luokitella monitavoiteoptimointitehtävien tavoitteiden vähentämisen

menetelmiä.

Avainsanat: monitavoiteoptimointi, Pareto-tehokkuus, tavoitteiden vähentäminen

Contents

1 Introduction

Optimization problems occur naturally in many different fields of work, for instance in engineering. In practice, a usual optimization problem will have multiple objectives to be minimized or maximized, for example cost, durability, reliability, etc. The often conflicting nature of multiple objectives, e.g. not being able to simultaneously minimize cost and maximize durability, is the reason that a solution to a multiobjective optimization problem is usually not a single point, but instead consists of a set of points none of which can be chosen as the best without some additional information (see for example the discussion in Chankong and Haimes 1983, 1.2.6). As the number of objectives in the problem grows, in general so does the number of conflicting pairs of objectives. Usually this also means an increase in the computational complexity in calculating the solution of the optimization problem.

On the practical side of things, it is often computationally infeasible to solve problems with a very high number of objectives. The most common evolutionary algorithms in particular are mostly well suited to at most a couple of objectives. Due to this reason a natural question is if it is possible to solve the optimization problem by either first simplifying the problem at hand, or by solving another, simpler, related problem, which would also give the solutions of the original problem.

The most extreme simplification is some form of scalarization, i.e. turning the multiobjective optimization problem into a single objective problem by some method. For a large number of objectives such an extreme simplification might however lose so much information that it is impractical, if not impossible, to deduce the solutions of the original problem. A less extreme simplification is simply omitting one or more objectives from the problem entirely. Then the question becomes which objectives should be omitted so as to lose as little information about the problem as possible. In some cases it is even possible to omit some objectives without any loss of information. In these cases solving the entire problem is unnecessary as solving the reduced problem would already give exactly the same solutions.

This problem of objective reduction in multiobjective optimization is the focus of this thesis. An overview of several different methods for various types of problems found in the liter-

ature will be presented. In addition, a few simple ways to classify the different methods is discussed. Finally, a sample algorithm of one of the methods will be presented as a Python implementation.

The structure of this thesis is as follows: chapter 2 will go through most of the general terminology and notations, as well as presenting multiple criteria for classifying the methods presented. Chapter 3 will look at methods for the simplest class of optimization problems: the linear case. Chapter 4 will present methods for convex problems under various types of convexity assumptions. Chapter 5 will cover methods under no particular assumptions on the problem itself. Finally, chapter 6 will present the implementation of an algorithm in chapter 4. The code and a test program for this implementation can be found in appendices A and B.

2 Overview of approaches and terminology

2.1 The multiobjective optimization problem

The multiobjective optimization problems that will be examined in this thesis will be of the form

$$\min_{x \in S} F(x), \tag{2.1}$$

where F is a vector valued function

$$F : S \to \mathbb{R}^m, \quad F(x) = (f_1(x), \ldots, f_m(x))$$

and the minimization is taken to mean minimization in all objectives simultaneously. Here S is called the *feasible region*, $x \in S$ is called the decision variable and the functions $f_1, \ldots, f_m : S \to \mathbb{R}$ are called the *objective functions*. The vector valued function F is called the *objective vector function*. For simplicity, it will be assumed that all the objective functions are to be minimized, since a maximization problem in some objective can always be changed into a minimization problem by replacing the objective f_j with the objective $-f_j$.

Since, in general, it is not possible to minimize all the functions $f_1, \ldots, f_m$ at once, a solution to the above problem will consist of a set of *Pareto optimal solutions*. A Pareto optimal solution is a point for which no objective function value can be decreased without increasing some other objective function value. In other words, a point $x \in S$ is Pareto optimal, if there exists no other point $y \in S$ for which $f_j(y) \leq f_j(x)$ for all indices j and the inequality is strict for at least one index. The value $F(x)$ corresponding to a Pareto optimal point x will be called a *Pareto optimal value*. The set of Pareto optimal points will be denoted by E. The set of Pareto optimal values is called the *Pareto front* and will be denoted by $F(E)$.

A related concept is the *Pareto dominance relation* $\leq_{\text{Par}}$, which is defined by letting $x \leq_{\text{Par}} y$, if $f_j(x) \leq f_j(y)$ for all indices j and the inequality is strict for at least one index. Using this relation we can rephrase the condition of Pareto optimality, by saying that a point $x \in S$ is Pareto optimal, if it is not Pareto dominated by any point $y \in S$, i.e. $y \leq_{\text{Par}} x$ is never satisfied in S.

Some results presented in this thesis will also require some more specific notions of Pareto

optimality, namely *strong Pareto optimality* and *weak Pareto optimality*. A point x in the feasible set S is strongly Pareto optimal, if there exists no other point $y \in S$, $y \neq x$, for which $F(y) \leq F(x)$. A point $x \in S$ is weakly Pareto optimal if there exists no point $y \in S$ for which $F(y) < F(x)$. Here the inequality on vectors $F(y) \leq F(x)$ is understood as the component-wise inequality $f_j(y) \leq f_j(x)$ for all j. The inequality $F(y) < F(x)$ has a similar meaning.

Sometimes it can be useful to characterize these notions of Pareto optimality by using *level sets* and *sublevel sets* (see Ehrgott and Nickel 2002). The level set of a function $f : S \to \mathbb{R}$ is defined as

$$L_c(f) = f^{-1}(\{c\}) = \{x \in S : \quad f(x) = c\},$$

the sublevel set as

$$L_c^-(f) = f^{-1}((-\infty, c]) = \{x \in S : \quad f(x) \leq c\}$$

and the strict sublevel set as

$$L_c^-(f) \setminus L_c(f) = f^{-1}((-\infty, c)) = \{x \in S : \quad f(x) < c\},$$

where $c \in \mathbb{R}$ is some constant.

Using these sets, we have the following characterizations: Let $x \in S$ and denote by $z_j = f_j(x)$ the value of the j:th objective at x.

(1) The point x is Pareto optimal if and only if

$$\bigcap_{j=1}^{m} L_{z_j}^-(f_j) = \bigcap_{j=1}^{m} L_{z_j}(f_j).$$

(2) The point x is strongly Pareto optimal if and only if

$$\bigcap_{j=1}^{m} L_{z_j}^-(f_j) = \{x\}.$$

(3) The point x is weakly Pareto optimal if and only if

$$\bigcap_{j=1}^{m} L_{z_j}^-(f_j) \setminus L_{z_j}(f_j) = \emptyset.$$

2.2 Redundant objectives

Consider the multiobjective optimization problem (2.1) with one added objective f_{m+1}. Denote the set of Pareto optimal solutions of this extended problem

$$\min_{x \in S}(f_1(x), \ldots, f_{m+1}(x)) \tag{2.2}$$

by E^+.

Even if the objective function f_{m+1} is different from all the other objectives $f_1, \ldots, f_m$, it might not contribute anything to the solutions of the problem. For example, consider the single objective optimization problem

$$\min_{x \in \mathbb{R}} x^2.$$

If we add to the objective set $\{x \mapsto x^2\}$ another objective $x \mapsto x^2 + 2$, then even though the new objective is different from the first, the solutions of the extended problem are exactly the same as the solutions of the original problem regardless of the feasible set. An objective f_{m+1} that does not change the set of solutions of the problem, i.e. for which $E = E^+$, is called *redundant* or *nonessential* (Gal and Leberling 1977).

2.3 Method classification

2.3.1 Feature selection vs. feature extraction

Trying to simplify a multiobjective optimization problem by removing redundant objectives is just one way to make multiobjective optimization problems computationally easier to solve. Another approach to dimensionality reduction is to instead attempt to combine or modify the objectives in a suitable way to find a new, preferably smaller, set of objectives, while still not changing the solutions of the problem. Methods of the former type are called *feature selection* approaches, while methods of the latter type are called *feature extraction* approaches (Brockhoff et al. 2008).

In this thesis, all of the methods presented will be of the feature selection type as these are much more common. Often they are also simpler to formulate as they are necessary finite problems in the sense that they are concerned with finding a specific subset of objectives.

Contrast this to the less limited problem of finding suitable, a priori arbitrary, new objectives. The one method presented in this thesis that could be considered a feature extraction approach is Deb and Saxena's principal component analysis based procedure (Deb and Saxena 2005) presented in section 5.2. Even though this procedure uses the feature extraction method of principal component analysis, its use is limited to deciding which objectives to choose. Thus even this method can be considered a feature selection method as the end result gives a subset of the original objectives.

2.3.2 Exact vs. approximate

In addition to the previously mentioned feature selection vs. extraction classification for methods of objective reduction in multiobjective optimization problems, it is possible to classify methods based on their consideration of error. In its simplest form, objective reduction in multiobjective optimization problems is concerned with finding the smallest subset of the original objectives for which the Pareto front remains the same. However, in some cases it may be possible to find an even smaller subset of objectives for which the new Pareto front is a reasonable approximation of the Pareto front of the original problem. Thus it is useful to classify objective reduction methods based on whether they attempt to exactly determine or only approximate the Pareto front of the original problem.

The latter approach is particularly common in evolutionary methods, such as the ones that will be presented in sections 5.1 and 5.2. This is mostly due to the fact that solving a multiobjective optimization problem with an evolutionary method usually means finding a good enough approximation of the Pareto front. Thus since the solutions themselves are not guaranteed to match the true Pareto front, trying to find an error-free reduction in the objectives may not even be necessary.

To contrast with this, most of the methods for more specific (or simpler) types of optimization problems, such as the linear and convex cases discussed in chapters 3 and 4, are looking for an error-free solution to the problem of objective reduction.

2.3.3 Problem reducing vs. problem solving

Another way to classify methods is to look at their end goals. From this point of view, methods can be split into two types. The first are the problem reducing methods. These methods attempt to take a multiobjective optimization problem and produce another problem that should be easier to solve. An example of such a method is given by Malinowska's algorithm (Malinowska 2006) presented in chapter 3, which determines whether a single additional objective is redundant.

The second type is made up of the problem solving methods. These methods aim to find the solutions of the problem, but instead of directly solving the original problem, they solve one or more reduced problems and combine the results suitably to get the solution to the original problem. An example of such a method is given by Malivert and Boissard's study of the structure of the weak Pareto front (Malivert and Boissard 1994) presented in section 4.1.2.

Both types of methods naturally have their advantages and it is not always clear which type is more suitable to solve the problem at hand. One advantage (or sometimes disadvantage) of methods of the first type is that they don't fix the actual method of solving the optimization problem itself. They merely modify the problem and leave one to select a suitable optimization method for the reduced problem, which may be readily solvable by a wider variety of methods than the original problem. This decoupling of the method of objective reduction from the problem solution itself provides a lot of flexibility for the use of these types of methods.

The main advantage of methods of the second type comes in the potential for efficiency. By integrating the problem reduction directly into the solution process, the methods gain the possibility of optimizing away unnecessary work. For instance, when using a method of the first type combined with another method to actually solve the problem, it is possible that the reduction method converts the data into a convenient output format for general use, which then gets reconverted back into its original form by the problem solver. This is completely unnecessary work from the point of view of the entire solution process, but due to the decoupling of the solution from the reduction, this can not be foreseen as it requires knowledge about the reduction method *and* solution process.

2.3.4 Pre-optimization vs. post-optimization reduction

One further point of view to objective reduction in multiobjective optimization is given by looking at *who* the method is intended to help. In the previous classifications the focus has been mainly on how the problem is solved in a way that is computationally efficient. However, in the study of multiobjective problems, a high number of objectives does not cause problems merely during the solution process. Indeed, even if the Pareto front were always easy to solve, as the number of objectives increases, it also becomes increasingly difficult for a decision maker to choose a preferred solution from all possible Pareto optimal solutions. This increase in complexity is the reason why it may in certain situations be a good idea to attempt to reduce the number of objectives even after the optimization process has been completed.

A series of examples of methods reducing the number of objectives after the Pareto optimal solutions have been found are given by Brockhoff and Zitzler's algorithms to solve the MOSS problem (Brockhoff and Zitzler 2006b) presented in section 5.1. Deb and Saxena's PCA based method on the other hand combines both pre- and post-optimization reduction by iterating the optimization process and the reduction process one after another.

3 Linear problems

A linear multiobjective optimization problem is a problem of the form (2.1), where all the objective functions are linear and the feasible region S is a subset of a vector space V that is determined by some linear functions $g_1, \ldots, g_k : V \to \mathbb{R}$ and constants $c_1, \ldots, c_k \in \mathbb{R}$ as

$$S = \bigcap_{j=1}^{k} \{x \in V : g_j(x) \leq c_j\}.$$

In this section, the problems will only be considered in the finite dimensional case with $S \subset \mathbb{R}^n$. In this setting, we can formulate the linear multiobjective optimization problem in matrix notation as

$$\min_{Gx \leq C} Ax, \tag{3.1}$$

where

$$A = \begin{bmatrix} f_1 & f_2 & \ldots & f_m \end{bmatrix}^T$$

is the matrix given by the objective functions,

$$G = \begin{bmatrix} g_1 & g_2 & \ldots & g_k \end{bmatrix}^T$$

is the matrix given by the constraint functions and $C = (c_1, \ldots, c_k) \in \mathbb{R}^k$ is a constant vector. Note that here the linear functions $f_j : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$ are being identified with vectors in $\mathbb{R}^n$, which makes A a $m \times n$-matrix and G a $k \times n$-matrix.

Gal and Leberling (1977) presented a simple way to determine if an objective function is redundant based on linear relations between the functions. In the classification criteria of section 2.3 this method is an exact, feature selection, problem reduction method.

Theorem 3.1. *The objective function f_m is redundant, if it is a linear combination of the other objective functions with non-negative coefficients, i.e. if there exist $\lambda_1, \ldots \lambda_{m-1} \geq 0$ such that*

$$f_m = \lambda_1 f_1 + \cdots + \lambda_{m-1} f_{m-1}.$$

Proof. See Gal and Leberling (1977, Thm. 2.3). □

This theorem is not in fact restricted to only the case of linear problems, but instead holds for any optimization problem for which the weight factor method finds all Pareto optimal solutions, see (Steuer 1986, Ch. 7) for the linear case and (Chankong and Haimes 1983, 4.3.3 and 6.2) for a more general setting. In the case of linear problems though, there is a straightforward way to verify the condition of the theorem. Using Gaussian elimination on the matrix A it is possible to find the linear relations within the objective functions. After this, all that is necessary to verify if the condition holds, is to look at the signs of the coefficients in the found linear relations.

A somewhat more in-depth result on objective redundancy in linear problems can be found in an article of Malinowska (2006). In this article, Malinowska presents an algorithm that completely determines whether an objective function is necessary or not. Although slightly more intricate, similarly to the previous method, this method is still an exact, feature selection, problem reduction method.

For the formulation of the method, consider a linear multiobjective optimization problem (3.1) extended with one added objective f_{m+1}. Denote the extended objective function matrix by

$$A^+ = \begin{bmatrix} f_1 & f_2 & \dots & f_m & f_{m+1} \end{bmatrix}^T$$

and let E^+ be the set of Pareto optimal solutions of this extended problem.

An important part of Malinowska's algorithm is determining whether the sets

$$\begin{aligned} U &= \{x \in \mathbb{R}^n : \quad Ax \leq_{\text{Par}} 0\} \quad \text{and} \\ U^+ &= \{x \in \mathbb{R}^n : \quad A^+x \leq_{\text{Par}} 0\} \end{aligned}$$

are empty or not. The algorithm itself boils down to the following theorems:

Theorem 3.2. *Let $M = \operatorname{argmin}_{x \in S} f_{m+1}(x)$ be the set of minimizers for the single objective f_{m+1}. For the extended multiobjective optimization problem (2.2), $E^+ \subset E$ if and only if*

$$E \cap M \neq \emptyset \quad \textit{and} \quad U^+ \neq \emptyset.$$

Proof. See Malinowska (2002, Theorem 1) and Malinowska (2006, Remark 1). □

Theorem 3.3. *If $U = \emptyset$, then $E = S$. Also, if $\operatorname{int} S \neq \emptyset$ then $E = S$ implies $U = \emptyset$.*

Proof. Malinowska refers the reader to a book by Galas, Nykowski, and Żółkiewski (1987). □

Theorem 3.2 is a slight modification of Theorem 1 in (Malinowska 2006). The statement of the theorem in Malinowska's article is concerned with determining a necessary and sufficient criterion for the added objective f_{n+1} to be redundant. By definition, the objective f_{n+1} is redundant exactly when $E = E^+$, which can be proved by showing the inclusions $E \subset E^+$ and $E^+ \subset E$. In the theorem of the article the first inclusion $E \subset E^+$ is one of the conditions required for the criterion. By removing this condition, the theorem changes to a necessary and sufficient condition for the inclusion $E^+ \subset E$, which is exactly the statement of Theorem 3.2.

Using these theorems, the algorithm goes as follows:

(1) Check whether $U^+ = \emptyset$. If not, go to (5).
(2) Check whether $U = \emptyset$. If so, then by Theorem 3.3 $E^+ = S = E$ and the objective f_{m+1} is redundant.
(3) Check whether $\operatorname{int} S = \emptyset$. If so, then by Theorem 3.3 $E^+ = S \neq E$ and the objective f_{m+1} is not redundant.
(4) Check if $E = S$. If so, the objective f_{m+1} is redundant, otherwise it is not.
(5) Determine the solutions M of the single objective problem $\min_{x \in S} f_{m+1}(x)$.
(6) Check whether $E \cap M = \emptyset$. If so, then by Theorem 3.2 $E^+ \not\subset E$ and hence the objective f_{m+1} is not redundant. Otherwise $E^+ \subset E$.
(7) Determine if $E \subset E^+$. If so, then $E^+ = E$ and the objective f_{m+1} is redundant. Otherwise it is not redundant.

As noted in Malinowska's article, the problematic steps in the above algorithm are (4) and (7). The other steps only require solving single objective linear optimization problems. For examples on the use of this algorithm, see Malinowska (2006, Section 5).

4 Convex problems

A subset S of a vector space is convex if for all $x, y \in S$ and $\lambda \in [0,1]$

$$\lambda x + (1-\lambda)y \in S.$$

Let $f : S \to \mathbb{R}$ be a function on a convex set S. The function f is

(1) *convex*, if for all $x, y \in S$ and $\lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y),$$

(2) *quasiconvex*, if for all $x, y \in S$ and $\lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)y) \leq \max\{f(x), f(y)\},$$

(3) *strictly quasiconvex*, if for all $x, y \in S$ and $\lambda \in (0,1)$

$$\begin{aligned} &f(\lambda x + (1-\lambda)y) < \max\{f(x), f(y)\} \quad \text{if } f(x) \neq f(y) \text{ and} \\ &f(\lambda x + (1-\lambda)y) \leq f(x), \quad \text{if } f(x) = f(y) \end{aligned}$$

Note that a convex function is always strictly quasiconvex and a strictly quasiconvex function is always quasiconvex. In this section, it will be assumed that in the multiobjective optimization problem (2.1), the feasible region S is a closed convex set. If in addition all the objective functions f_j are convex, the problem will be called a convex multiobjective optimization problem, or in short, a convex problem. Similarly, the problem will be called (strictly) quasiconvex, if all the objective functions are (strictly) quasiconvex.

4.1 Convex and strictly quasiconvex case

4.1.1 A redundancy criterion

A similar result to that of Theorem 3.2 holds also for certain convex multiobjective optimization problems. As the result is virtually identical, the classification of the problem is also identical, i.e. the method is an exact, feature selection, problem reduction method. Due

to the generalization from the linear result to the convex case, a few additional assumptions about the problem are needed.

Consider the extended multiobjective optimization problem (2.2) and assume that the problem is convex. Assume in addition, that the feasible region $S \subset \mathbb{R}^n$ is bounded and that the objective functions f_j are continuous convex functions defined on all of $\mathbb{R}^n$. Let E_∞^+ denote the set of Pareto optimal solutions of the unbounded problem

$$\min_{x \in \mathbb{R}^n} (f_1(x), \ldots, f_{m+1}(x)).$$

Under these assumptions, the following theorem (Malinowska 2002) holds:

Theorem 4.1. *Let $M = \operatorname{argmin}_{x \in S} f_{m+1}(x)$ be the set of minimizers for the single objective f_{m+1}. Then, $E^+ \subset E$ if and only if*

$$E \cap M \neq \emptyset \quad \textit{and} \quad S \setminus E \subset \mathbb{R}^n \setminus E_\infty^+.$$

The difference between this theorem and Theorem 3.2 is the use of the condition $S \setminus E \subset \mathbb{R}^n \setminus E_\infty^+$ as opposed to the condition $U^+ \neq \emptyset$ in the linear case. In fact, this theorem is more general, since the condition $S \setminus E \subset \mathbb{R}^n \setminus E_\infty^+$ implies $U^+ \neq \emptyset$ in the linear case. The latter condition is however simpler to check, which is why it is preferred in the linear case.

It is worth noting that this last condition $S \setminus E \subset \mathbb{R}^n \setminus E_\infty^+$ is why the objective functions were assumed to be defined on all of $\mathbb{R}^n$ even if solutions to the original optimization problem are only looked for in some subset $S \subset \mathbb{R}^n$. This however limits the applications of the above theorem as not all convex functions can be extended to convex functions in a strictly larger set. This is the case for example with the convex continuous function

$$[0, \infty) \to \mathbb{R}, \quad x \mapsto -\sqrt{x}$$

as there exists no convex function $f : \mathbb{R} \to \mathbb{R}$, with $f(x) = -\sqrt{x}$ when $x \geq 0$.

4.1.2 Weakly Pareto optimal solutions

In the following, assume that the multiobjective optimization problem (2.1) is a strictly quasiconvex problem and that the feasible region is contained in the n-dimensional space $\mathbb{R}^n$.

Ward (1989) showed that for convex problems, the set of weakly Pareto optimal solutions can be determined by looking at the sets of Pareto optimal solutions of subproblems with at most $n+1$ objectives. Malivert and Boissard (1994) later extended this result to the strictly quasiconvex case. They also showed that in the case of continuous strictly quasiconvex objectives, with some further calculations, the weak Pareto solutions can be determined using the Pareto solutions of subproblems with at most n objectives. Both of the following results can be classified as exact, feature selection, problem solving methods.

For these results let us fix some more notation. As in (2.1), let the objective functions f_j be indexed by $j \in \{1,\ldots,m\}$. For any subset of the index set $I \subset \{1,\ldots,m\}$ let $E(I)$ denote the set of Pareto solutions of the subproblem with the objectives $\{f_j : j \in I\}$. Denote by $|I|$ the number of elements of I. Under this notation, the first result of Malivert and Boissard (1994) is as follows:

Theorem 4.2. *If all the objective functions f_j are upper semicontinuous along line segments, then the set of weak Pareto solutions of the original problem is given by the union of the sets of Pareto solutions of all subproblems with at most $n+1$ objectives. In other words,*

$$E_W = \bigcup_{\substack{I\subset\{1,\ldots,m\}\\ 0<|I|\leq n+1}} E(I).$$

With the additional assumption that the objective functions are continuous and that the feasible region is bounded the evaluation of subproblems of size $n+1$ can be swapped out for an alternate condition. Let R be the set of points $x \in S$, for which any half-line emanating from x meets a Pareto optimal point of some subproblem of at most n problems. Precisely, let

$$R = \{x \in S : \quad \exists I \subset \{1,\ldots,m\}, 0 < \#I \leq n,\ \forall v \neq 0,\ \exists t \geq 0,\ x+tv \in E(I)\}.$$

Using this set R, Malivert and Boissard showed the following representation for the set of weak Pareto solutions of the original problem:

Theorem 4.3. *If all the objective functions are continuous and the feasible region is bounded, then*

$$E_W = R \cup \bigcup_{\substack{I\subset\{1,\ldots,m\}\\ 0<|I|\leq n}} E(I).$$

4.1.3 Pareto optimal solutions for problems with 2 variables

The previous results are concerned with determining the set of weak Pareto solutions of the original problem using the Pareto solutions of subproblems with at most $n+1$ or n objectives. Ehrgott and Nickel (2002) considered the related problem of distinguishing weakly Pareto optimal points from Pareto optimal points, while still holding on to the criteria of not solving problems with more than $n+1$ objectives. The main result of their article is a Pareto optimality criterion in the case where the domain of the objectives is 2-dimensional. Additionally, they also presented an algorithm based on this result for checking whether a given point is Pareto optimal or not. In the classification of section 2.3, this algorithm is an exact, feature selection, problem solving method.

The result is formulated as follows (Ehrgott and Nickel 2002, Theorem 4.3):

Theorem 4.4. *Let the multiobjective optimization problem (2.1) be a strictly quasiconvex problem with upper semi-continuous objectives and with $S \subset \mathbb{R}^2$. Then a point $x \in S$ is Pareto optimal, if either*

(1) the point x is strongly Pareto optimal for some subproblem with at most 3 objectives, or

(2) there exists subsets $I_1, \ldots, I_k \subset \{1, \ldots, m\}$, $|I_j| \leq 3$, such that $I_1 \cup \cdots \cup I_k = \{1, \ldots, m\}$ and $x \in E(I_j)$ for all $j \in \{1, \ldots, k\}$.

The algorithm based on this theorem uses the characterization of strong Pareto optimality based on level sets presented at the end of section 2.1, i.e. that a point $x \in S$ is strongly Pareto optimal, if and only if

$$\bigcap_{j=1}^{m} L_{z_j}^{-}(f_j) = \{x\},$$

where $z_j = f_j(x)$.

Assume that the assumptions of Theorem 4.4 are satisfied. Then Ehrgott and Nickel's algorithm to determine whether a point $x \in S$ is Pareto optimal goes as follows:

(1) Set $k = 0$ and $\mathcal{Q} = \{I \subset \{1, \ldots, m\}, \quad |I| \leq 3\}$.

(2) Select an index set $I \in \mathcal{Q}$. Set $\mathcal{Q} = \mathcal{Q} \setminus \{I\}$.

(3) If $\bigcap_{j \in I} L_{z_j}^{-}(f_j) = \{x\}$, the point x is strongly Pareto optimal for the subproblem with

objectives f_j, $j \in I$. By Theorem 4.4, the point x is then Pareto optimal for the original problem.

(4) Check whether $x \in E(I)$. If so, set $k = k+1$ and $I_k = I$. If now $\bigcup_{j=1}^{k} I_k = \{1,\ldots,m\}$, then by Theorem 4.4 the point x is Pareto optimal also for the original problem.

(5) If $\mathcal{Q}$ is nonempty, go to (2). Otherwise the point x is not Pareto optimal.

4.2 Quasiconvex case

Ehrgott and Nickel (2002) showed that the result presented in Theorem 4.2 can not be extended as is to the case where the objectives are merely quasiconvex, as opposed to *strictly* quasiconvex. The counterexample (Ehrgott and Nickel 2002, Example 3.1) consisted of the three continuous piecewise linear objectives

$$f_1 : \mathbb{R} \to \mathbb{R}, \quad f_1(x) = \begin{cases} 4, & \text{if } x < 0 \\ -\frac{1}{3}x + 4, & \text{if } 0 \le x < \frac{9}{2} \\ \frac{5}{2}, & \text{if } \frac{9}{2} \le x \end{cases}$$

$$f_2 : \mathbb{R} \to \mathbb{R}, \quad f_2(x) = \begin{cases} \frac{3}{2}, & \text{if } x < 1 \\ \frac{6}{5}x + \frac{3}{10}, & \text{if } 1 \le x < \frac{7}{2} \\ \frac{9}{2}, & \text{if } \frac{9}{2} \le x \end{cases} \quad \text{and}$$

$$f_3 : \mathbb{R} \to \mathbb{R}, \quad f_3(x) = \begin{cases} x + \frac{9}{4}, & \text{if } x < \frac{1}{2} \\ \frac{11}{4}, & \text{if } \frac{1}{2} \le x < \frac{9}{2} \\ x - \frac{7}{4}, & \text{if } \frac{9}{2} \le x \end{cases}$$

The observation made in the article was that for the multiobjective optimization problem defined by these objective functions,

$$E_W \not\subset \bigcup_{I \subset \{1,\ldots,3\}} E(I).$$

Indeed, any point in $\mathbb{R}$ is weakly Pareto optimal for the entire problem, but any point on the half-open interval $[\frac{7}{2}, \frac{9}{2})$ is not a Pareto optimal for the problem or any subproblem.

To fix this issue in the case of merely quasiconvex objectives, the union needs to be taken over the *weak* Pareto solutions of all subproblems of limited size. Then the following result (Ehrgott and Nickel 2002, Corollary 3.1) holds:

Theorem 4.5. *Let the multiobjective optimization problem (2.1) be a quasiconvex problem. Then the set of weak Pareto solutions of the original problem is given by the weak Pareto solutions of all subproblems with at most $n+1$ objectives. In other words,*

$$E_W = \bigcup_{\substack{I \subset \{1,\ldots,m\} \\ 0<|I|\leq n+1}} E_W(I).$$

5 General problems

5.1 MOSS

The point of this section will be to look at a series of algorithms by Brockhoff and Zitzler. All of these algorithms revolve around the Minimum Objective Subset problem (MOSS) presented in (Brockhoff and Zitzler 2006b). The Minimum Objective Subset problem is the problem of finding the smallest subset of objectives of the multiobjective optimization problem 2.1, such that the weak Pareto dominance relation is left unchanged on a given set $X \subset S$, see (Sawaragi, Nakayama, and Tanino 1985, 2.3) for more information on dominance structures. More precisely, the problem is concerned with finding the smallest possible index set $I \subset \{1,\ldots,m\}$ such that for any $x,y \in X$, $f_j(x) \leq f_j(y)$ for all $j \in \{1\ldots,m\}$ if and only if $f_j(x) \leq f_j(y)$ for all $j \in I$.

The usual setting for applications of this problem is when X is an approximation of the set of Pareto optimal solutions by a finite number of points, possibly obtained by some optimization process. For this situation, it is possible to determine the solution to the MOSS problem exactly via computational methods. Note that it is always possible that the only index set preserving the weak Pareto dominance relation is the whole index set $I = \{1,\ldots,m\}$, in which case no reduction in objectives through the MOSS problem can be found.

5.1.1 MOSS

Brockhoff and Zitzler (2006b) showed that the MOSS problem is $\mathcal{NP}$-hard, which implies that for a large number of objectives or a large set X, the problem becomes infeasible to solve perfectly computationally. For this reason Brockhoff and Zitzler presented both an exact algorithm and a greedy algorithm to solve the problem. In the classification of section 2.3, the algorithms both fall into the feature selection and problem reducing categories. Naturally, the exact algorithm is an exact method, whereas the greedy algorithm is an approximate one.

The greedy version of the algorithm is based on two observations about the MOSS problem.

The first is that the weak Pareto dominance relation $\leq_{\text{W-Par}} \subset X^2$,

$$x \leq_{\text{W-Par}} y \quad \text{if and only if} \quad F(x) \leq F(y)$$

is the intersection of the m relations $\leq_j \subset X^2$ defined by single objectives as

$$x \leq_j y \quad \text{if and only if} \quad f_j(x) \leq f_j(y),$$

i.e.

$$\leq_{\text{W-Par}} = \bigcap_{j=1}^{m} \leq_j .$$

The second observation is that when considering the complements of these relations, the intersection in the previous formula becomes a union, i.e.

$$\leq^C_{\text{W-Par}} = \bigcup_{j=1}^{m} \leq^C_j ,$$

where $\leq^C_{\text{W-Par}}$ is the complement relation

$$\leq^C_{\text{W-Par}} = X^2 \setminus \leq_{\text{W-Par}}$$

and similarly $\leq^C_j$ is the complement relation for $\leq_j$. Using these observations the greedy algorithm amounts to selecting indices $j \in \{1,\ldots,m\}$ one after another such that $\leq^C_j$ covers as much as possible of the part of $\leq^C_{\text{W-Par}}$ not yet covered.

Precisely, the greedy algorithm for the MOSS problem is as follows:

(1) Set $R = \leq^C_F$ and $I = \emptyset$.
(2) If $R = \emptyset$, the index set I is the solution.
(3) Select an index $j \in \{1,\ldots,m\} \setminus I$ such that $|\leq^C_j \cap R|$ is maximal.
(4) Set $R = R \setminus \leq^C_j$ and $I = I \cup \{j\}$. Go to step (2).

Brockhoff and Zitzler (2006b, Theorem 4) showed that this algorithm has a runtime of order $O(|X|^2 m)$ and an approximation ratio of $\Theta(\log|X|)$.

For the presentation of the exact algorithm, it is convenient to define some additional notation. The algorithm works using sets of index sets $S \subset \mathcal{P}(\{1,\ldots,m\})$, for which 2 types

of set operations will be needed. The first is the pairwise union on elements, $\sqcup$, defined by setting

$$S_1 \sqcup S_2 = \{I_1 \cup I_2 : I_1 \in S_1,\, I_2 \in S_2\}.$$

The second operation is the minimal set operation, min, which creates a new set out of those index sets in S, which do not contain as proper subsets any other index set in S. In other words,

$$\min S = \{I \in S : \nexists J \in S,\, J \subsetneq I\}.$$

In Brockhoff and Zitzler's article these two operations are combined into the notation $\sqcup$, so the operation $(S_1, S_2) \mapsto S_1 \sqcup S_2$ in their article corresponds to the operation $(S_1, S_2) \mapsto \min(S_1 \sqcup S_2)$ in this thesis.

The exact algorithm for the MOSS problem:

Set $S = \emptyset$.
for $(x, y) \in X^2$ **do**
  Set $S_x = \{\{j\} : f_j(x) < f_j(y)\}$.
  Set $S_y = \{\{j\} : f_j(x) > f_j(y)\}$.
  Set $S_{xy} = S_x \sqcup S_y$.
  if $S_{xy} = \emptyset$ **then**
    Set $S_{xy} = \{\{1\}, \ldots, \{m\}\}$.
  end if
  Set $S = \min(S \sqcup S_{xy})$.
end for
The smallest index set $I \in S$ is the solution.

Brockhoff and Zitzler (2006b, Theorem 5) proved that this algorithm indeed gives the exact solution to the MOSS problem, and also showed that its run time is of order $O(|X|^2 m 2^m)$. The main idea of the algorithm is to find all index sets for which no objective can be removed without changing the weak Pareto dominance relation on the set X. The smallest such index set is then the solution of the MOSS problem.

The algorithm works by keeping track of the smallest possible index sets which explain the weak Pareto dominance relation on the pair of points considered so far. Indeed at every

step of the for-loop, the sets S_{xy} contain as elements the smallest possible index sets, which explain the relation on the single pair of points (x,y). That is, for any index set $I \in S_{xy}$, $F(x) \leq F(y)$ if and only if $f_j(x) \leq f_j(y)$ for all $j \in I$ and $F(x) \geq F(y)$ if and only if $f_j(x) \geq f_j(y)$ for all $j \in I$.

Notice that if an index set I explains the relation on some pairs of points $(x_1,y_1),\ldots,(x_k,y_k)$ and another index set J explains the relation on another bunch of pairs of points $(x_k+1,y_k+1),\ldots,(x_l,y_l)$, then the union of the index sets $I \cup J$ explains the relation on all the pairs $(x_1,y_1),\ldots,(x_l,y_l)$. This is why after every step the set S contains the smallest possible index sets explaining the relation on all pairs so far. Thus, at the end of the algorithm, the smallest index set in S is the solution to the MOSS problem.

Notice also, that the operation $(S_1,S_2) \mapsto \min(S_1 \sqcup S_2)$ is associative. That is, when chaining multiple operations, the order does not matter. In fact,

$$\min(S_1 \sqcup \min(S_2 \sqcup S_3)) = \min(S_1 \sqcup S_2 \sqcup S_3) = \min(\min(S_1 \sqcup S_2) \sqcup S_3).$$

If the for-loop in the exact algorithm for MOSS is unraveled, the algorithm amounts to calculating the set

$$\min(\ldots(\min(\min S_1 \sqcup S_2) \sqcup S_3)) \cdots \sqcup S_l),$$

where $S_1,\ldots,S_l$ are the sets S_{xy} for all pairs of points $x,y \in X$. By associativity, this operation does not have to be done sequentially. This means that the algorithm can actually be parallelized by splitting the sets $S_1,\ldots,S_l$ into suitable segments and combining the results with the same $\min(\cdot \sqcup \cdot)$ operation. The main problem limiting the usefulness of this parallelization is that the runtime of the algorithm is exponential with respect to the number of objectives, whereas it is merely quadratic with respect to the size of the set X, which is where this parallelization might save time.

5.1.2 δ-MOSS and k-EMOSS

The two variants of the MOSS problem that were studied by Brockhoff and Zitzler are δ-MOSS and k-EMOSS (Brockhoff and Zitzler 2006a). Both of these modify the MOSS problem to include a consideration for error, but the problems have different goals. Brockhoff and Zitzler presented both an exact and an approximate algorithm for both methods. All of the

algorithms presented in their article fall into the category of approximate, feature selection, problem reducing methods.

The consideration for error is handled by replacing the weak Pareto dominance relation with the relation $\leq_\varepsilon$, defined by setting $x \leq_\varepsilon y$ if

$$f_j(x) - \varepsilon \leq f_j(y) \quad \text{for all } j \in \{1, \ldots, m\},$$

where $\varepsilon > 0$ is some constant error term.

The δ-MOSS problem is concerned with finding the smallest subset of objectives, such that the relation $\leq_\delta$ is left unchanged on a given subset $X \subset S$. The k-EMOSS problem, on the other hand, is concerned with finding a subset of at most k objectives, such that the relation $\leq_\delta$ that is left unchanged on a given $X \subset S$ has the smallest possible error $\delta \geq 0$. In other words, if (I, δ) is a solution to the k-EMOSS problem, then for the subset of objectives $I \subset \{1, \ldots, m\}$, $|I| \leq k$ and the relation $\leq_\delta$ is left unchanged on X. Furthermore, if $J \subset \{1, \ldots, m\}$ is a subset of objectives with $|J| \leq k$ that leaves a relation $\leq_\varepsilon$ unchanged on X, then $\varepsilon \geq \delta$.

The exact algorithm for the δ-MOSS and k-EMOSS problems works fundamentally in a similar way to the exact algorithm for the original MOSS problem. That is, by sequentially considering all pairs of points in X one after another and keeping track of all possible solutions that preserve the $\leq_\delta$ relation on the pairs of points considered so far. The greedy versions of the algorithms also work essentially similarly to how the greedy algorithm for the original MOSS problem worked. However, due to the need to try to either optimize the error term in k-EMOSS or satisfy the error constraint in δ-MOSS, the selection of the best index set is slightly more nuanced.

5.2 PCA

5.2.1 PCA-NSGA-II

Deb and Saxena's PCA-NSGA-II procedure (Deb and Saxena 2005) combines the principal component analysis (PCA) method with the NSGA-II evolutionary multiobjective optimization method presented by Deb, Pratap, Agarwal and Meyarivan in an earlier article (Deb et al. 2002). The idea is to supplement the shortcomings of the NSGA-II method in prob-

lems with a large number of objectives by iteratively reducing the number of objectives. The method can be classified as an approximate, feature selection, problem solving approach. This method however straddles the line between feature selection and feature extraction, as the way in which objectives are selected is based on the feature extraction method of PCA.

As usual, the setting is the multiobjective optimization problem (2.1) with m objectives. The main steps of the PCA-NSGA-II procedure are as follows (Deb and Saxena 2005, 5.3):

(1) Set $t = 0$ and $I_0 = \{1, \ldots, m\}$.
(2) Initialize a random population. Let P_t be the population obtained after running the NSGA-II evolutionary optimization method for the problem with objectives I_t.
(3) Define a reduced set of objectives I_{t+1} via principal component analysis on the population P_t.
(4) If there was no change in objectives, i.e. $I_{t+1} = I_t$, end the procedure. The population P_t gives the approximation of the Pareto front. Otherwise set $t = t + 1$ and go to (2).

From the objective reduction point of view, the interesting step in the above algorithm is step (3). The main idea here is to determine if and how the objectives are correlated, and to select the reduced set of objectives based on this knowledge. Thus, instead of working with the objective vectors $F(x) \in \mathbb{R}^m$ given by points x in the population P_t, the principal component analysis will use the vectors of single objective values

$$X_j = \{f_j(x_1), \ldots, f_j(x_k)\} \in \mathbb{R}^k,$$

where $k = |P_t|$ is the number of points in the population.

First the vectors X_j are standardized to have zero means and unit variance. Then the correlations of these standardized vectors are calculated. Next the principal components, i.e. the eigenvectors of correlation matrix, and the corresponding eigenvalues are calculated.

The principal components are by assumption sorted in order of decreasing magnitude of the corresponding eigenvalue. This means that the first principal component is the most significant and explains the largest amount of variance in the data. Let $l \in \mathbb{N}$ be the smallest index such that the first l principal components $W_1, \ldots, W_l$ explain at least 95% of the variance in the data. These l principal components are then used to determine which objectives to

select for the next iteration.

For the first (most significant) principal component W_1, select the objectives corresponding to the largest positive and smallest negative element of the vector. The other principal components determine which objectives to select via the following process:

If all elements of the principal component vector are positive, select the objective corresponding to the largest positive element. If all elements are negative, select all objectives. Otherwise consider the ratio of the magnitudes of the largest positive element and the smallest negative element

$$r = \left| \frac{\max_i a_i}{\min_j a_j} \right|,$$

where the principal component in question is the vector $(a_1, \ldots, a_{\tilde{m}}) \in \mathbb{R}^{\tilde{m}}$. If the ratio is too small or large, then the objective corresponding to either the largest positive or smallest negative element is considered insignificant and only the more significant objective is chosen. In Deb and Saxena's implementation an upper bound of $1,25$ and a lower bound of $0,9$ are used. Thus, if $r > 1,25$, select only the objective corresponding to the largest element, if $r < 0,9$, select only the one corresponding to the smallest element, and otherwise select the objectives corresponding to the largest and smallest elements.

The last step in the reduction process is to look at the reduced correlation matrix, that is, the correlation matrix of only the objectives that were selected in the previous steps. If there exists a pair of objectives with a positive correlation amongst themselves and identical correlations with the other objectives, then one objective from the pair is deemed redundant. When no such pairs exist, the final reduced list of objectives is found.

5.2.2 C-PCA-NSGA-II and MVU-PCA-NSGA-II

In a later article (Saxena and Deb 2007), Deb and Saxena presented two modifications of the original PCA-NSGA-II procedure, the C-PCA-NSGA-II and MVU-PCA-NSGA-II methods. Both of the variants follow the same outline as the original PCA-NSGA-II procedure, differing only in how the reduced set of objectives is determined. As such they still fall into the classification of approximate, feature selection, problem solving methods.

Both of the variants were designed to be able to avoid one of the main shortcoming of the original PCA based method. As PCA looks at the data as a whole and looks for the linear relations therein, it is not well suited to problems where the data is contained in some nonlinear submanifold. As mentioned in the article of Deb and Saxena, the variants of the original method are based on the strategy of embedding the data into a feature space where the data can be more readily analyzed with essentially linear methods such as PCA.

The two variants differ in how they choose to embed the data into a feature space. In the C-PCA-NSGA-II procedure Deb and Saxena utilize the correntropy PCA technique presented by Xu et al. (2006). Here correntropy is defined as a generalization of the standard correlation function which is used in standard PCA. Deb and Saxena note however, that using the correntropy PCA technique with the PCA-NSGA-II procedure is problematic due to the need to select a suitable kernel function for the embedding into a feature space, which naturally varies depending on the problem at hand.

The MVU-PCA-NSGA-II procedure is a further refinement to avoid the previously mentioned problem of choosing a suitable kernel function. The problem is avoided by essentially formulating the selection of the kernel as a convex optimization problem. It is based on the maximum variance unfolding technique presented by Weinberger and Saul (2006). The main idea is to find a locally distance preserving representation of the data that also maximizes the sum of pairwise distances between the points. Weinberger and Saul give an intuitive description of this method by describing a beaded necklace that has been coiled up: “By pulling the necklace taut, the beads are arranged in a line, a nonlinear dimensionality reduction from $\mathbb{R}^3$ to $\mathbb{R}^1$”.

6 An implementation of Ehrgott and Nickel's Pareto optimality algorithm

6.1 Overview of the implementation

In this chapter, an implementation of a variant of Ehrgott and Nickel's Pareto optimality checking algorithm (see section 4.1.3) will be presented. Due to practical matters of the computations involved, the actual algorithm does not directly reflect the more abstract algorithm presented earlier. Namely step (3), the checking for strong Pareto optimality, has been completely ignored due to the infeasibility of numerically confirming whether the Pareto optimality of a point is strong or not.

The main problem here is that even for convex problems, confirming strong Pareto optimality for a point x in the domain would require checking that a neighborhood of the point contains no point y which is at least as good as x, i.e. $f(y) \leq f(x)$. If there is a better point, i.e. a point y for which $y \leq_{\text{Par}} x$, then the numerical solution should arrive at the result that x is in fact not Pareto optimal. However, in the case that there is a point $x \neq y$ with $f(y) = f(x)$, there is no guarantee that a numerical optimization will find such a point y when minimizing in the neighborhood of the point x.

Dropping step (3) causes the algorithm to lose its "if and only if" guarantee, and it will instead give a sufficient, but not always necessary, criterion for the Pareto optimality of a given point x. However, since strong Pareto optimality implies Pareto optimality it is possible to determine the cases where a Pareto optimal point might slip through unconfirmed due to the omission of step (3). If there are no subproblems, for which the point is Pareto optimal, then there are also no subproblems for which it is strongly Pareto optimal. In these cases the implemented algorithm can confirm that the point is in fact not Pareto optimal. Because of this, the implementation of the algorithm has three possible results for the point: Pareto optimal, not Pareto optimal, and uncertain about Pareto optimality.

The implementation has been coded in Python 2.7 (Python Software Foundation 2015) utilizing assorted default libraries, and using the SciPy and NumPy libraries (Jones, Oliphant,

Peterson, et al. 2001–), see also (Oliphant 2007), to handle the optimization of single objective problems. The NumPy library is used mainly for convenience and integration with the SciPy optimization framework, which works with NumPy arrays. The SciPy library is used to provide the downhill simplex algorithm to solve a single objective minimization problem. Although the implementation has not been created in a directly modular way, it would be straightforward to swap out this minimizer for some other arbitrary optimizer provided by another library.

6.2 Details on the functionality

The code for the implementation itself can be found in appendix A and an example of its use in the case of the problem described in section 6.3.1 can be found in appendix B. The code has not been tested for any other version of Python other than 2.7 and is thus not guaranteed to work for any other version.

In the code, the main algorithm is contained in the function *isPareto*, which takes as arguments a function to calculate objective vector values, the point whose Pareto optimality is being checked, a parameter for the achievement scalarizing function (see below), and a boolean value that determines if all subproblems should be checked even if Pareto optimality has already been confirmed. The return value of the function is 1, 0 or -1, where 1 means that the point is Pareto optimal, -1 means it is not Pareto optimal, and 0 means it is unknown whether the point is Pareto optimal or not.

The first step of the algorithm is to generate a list of all index sets which correspond to the subproblems for which the Pareto optimality of the given point needs to be checked. This is done by using the powerful Python default library of *itertools* to generate all subsets of the entire index set with 1,2 or 3 elements.

After this, the algorithm loops through the index sets one by one checking whether the point is Pareto optimal for the subproblem where only the objectives in the current index set are considered. This check itself is done via converting the 1,2 or 3 objective optimization problem into a single objective problem by considering the minimization of the achievement

scalarizing function

$$g : S \to \mathbb{R}, \quad g(x) = \max_{j \in I}(f_j(x) - f_j(x_0)) + \rho \sum_{j \in I} f_j(x),$$

see (Wierzbicki 1981) and (Miettinen 1999). Here x_0 is the point whose Pareto optimality is being checked, I is the current index set and ρ is a suitable small constant. This constant ρ is the third parameter of the function *isPareto* mentioned above. The key point in the way this method works is that the point x_0 is Pareto optimal for the subset of objectives I if and only if it is the minimizer for the achievement scalarizing function for some small enough constant ρ.

As mentioned earlier, the optimization of the single objective problem is done by using the downhill simplex algorithm (see Steuer 1986, 3.4) provided by the SciPy library. This algorithm was selected as a reasonable optimizer for a generic (convex) optimization problem. It is worth noting that this is an unconstrained optimizer, whereas the optimization problems being solved may be constrained and it is even possible that the objectives are not defined outside these constraints. However, due to the assumption of strict quasiconvexity of the objectives, any globally Pareto optimal point is also locally Pareto optimal, and vice versa. Therefore even the downhill simplex algorithm works well enough when the point being tested is not on the boundary of the feasible region.

The COBYLA method provided by the SciPy library was also tried as an alternative optimizer, with the hope of being able to properly handle constraints. The problem with this approach is however that the linear approximations used in this method are not suitable for minimizing the achievement scalarizing function. This is due to the presence of the max operation, which may cause edges to appear on the graph of the function, where the linear approximations on either side of the edge are wildly different. This type of behaviour can be seen for example with the function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x,y) = x + |y|.$$

Using the default parameters and an initial guess of the point $x_0 = (0,0)$, the COBYLA implementation of the SciPy library returns the same point $(0,0)$ as the minimizer of the function f. This is not, however, a minimizer of f, since the value of f can be decreased simply by decreasing the value of the variable x.

Finally, due to numerical accuracy issues, an additional tolerance parameter was implemented that determines when the point x_0 is considered to be the minimizer of the achievement scalarizing function. This means that as long as the point $x_{\min}$ given by the optimization procedure is close to the point x_0 in both distance and in value of the function, the point x_0 is assumed to be a minimizer. In other words, if

$$|x_{\min} - x_0| < \delta \quad \text{and} \quad g(x_0) \leq g(x_{\min}) + \delta$$

for the tolerance parameter δ, then x_0 is assumed to be a minimizer and thus it is assumed that x_0 is Pareto optimal for the subset of objectives in question.

6.3 Examples

The implemented algorithm was tested on two main problems checking Pareto optimality for a few different points known to be (or known not to be) Pareto optimal. This section will present the results from these tests.

6.3.1 A simple linear problem

The first testcase is essentially merely check on the functionality of the algorithm. It involves 4 linear functions chosen in such a way as to make *every* point a (strongly) Pareto optimal solution. The objectives were defined as

$$\begin{aligned}
&f_1 : \mathbb{R}^2 \to \mathbb{R}, \quad f_1(x,y) = x, \\
&f_2 : \mathbb{R}^2 \to \mathbb{R}, \quad f_2(x,y) = y, \\
&f_3 : \mathbb{R}^2 \to \mathbb{R}, \quad f_3(x,y) = -2x - y \quad \text{and} \\
&f_4 : \mathbb{R}^2 \to \mathbb{R}, \quad f_4(x,y) = x + y.
\end{aligned}$$

As all the objectives are linear, they are immediately continuous and strictly quasiconvex, so the assumptions of the method are immediately verified. To see that any point is strongly Pareto optimal, one can use the criterion mentioned in section 2.1, i.e. that a point (x,y) is strongly Pareto optimal if and only if

$$\bigcap_{j=1}^{m} L^-_{z_j}(f_j) = \{(x,y)\},$$

where $z_j = f_j(x)$ is the value of the j:th objective at (x,y).

Here, the objectives are all linear, so the sublevel sets involved are all closed halfplanes with a common boundary point at (x,y). It is easily verified that no other point is contained in the intersection of the halfplanes. For example, at the origin $(0,0)$, there is no point (x,y) other than $(0,0)$ itself that satisfies

$$x \leq 0, \quad y \leq 0 \quad \text{and} \quad -2x - y \leq 0.$$

The above equations imply that the origin is a (strongly) Pareto optimal point also for the subproblem given by the objective subset $\{1,2,3\}$. Similarly one can see that it is a (strongly) Pareto optimal point for the subproblem with objectives $\{1,3,4\}$. For any other subproblem with at most 3 objectives, the origin is not Pareto optimal.

The same is true in general, i.e. any point x is (strongly) Pareto optimal for the subproblems with objectives $\{1,2,3\}$ or $\{1,3,4\}$ and not Pareto optimal for any other subproblem with at most 3 objectives. This was indeed correctly verified by the algorithm implementation, i.e. the algorithm gave a positive result for Pareto optimality of an arbitrary point. This was tested with 100 random points in the square $[-10,10]^2 \subset \mathbb{R}^2$. The implementation of this example can be found in appendix B.

6.3.2 A more varied problem

The second problem tested has a few more varied types of objectives and, more importantly, has a nontrivial Pareto front. The objectives for this problem are

$$\begin{aligned}
f_1 &: S \to \mathbb{R}, \quad f_1(x,y) = (x-1)^2 + \frac{1}{2}y^2, \\
f_2 &: S \to \mathbb{R}, \quad f_2(x,y) = 0.15y, \\
f_3 &: S \to \mathbb{R}, \quad f_3(x,y) = 1/(4-x^2-y^2), \\
f_4 &: S \to \mathbb{R}, \quad f_4(x,y) = \max\{-x-y, -x\} \quad \text{and} \\
f_5 &: S \to \mathbb{R}, \quad f_5(x,y) = (x+2)^2 + (y+2)^2.
\end{aligned}$$

where $S = B(0,2) \subset \mathbb{R}^2$. Again, these functions are all continuous and strictly quasiconvex, so the method works for these objectives. For this problem any point in the set

$$\{(x,y) \in S, \quad y \leq 0,\, x - y \geq 0\}$$

is a Pareto optimal solution and no other Pareto optimal solutions exist.

The behavior of the objectives can be seen from their level sets. For the first objective, the level sets are ellipses centered at $(1,0)$. For the second objective, lines in the direction of the x-axis. For the third, circles centered at the origin. The fourth objective has as its level sets bent lines made up by two halflines emanating in the directions $(0,1)$ and $(1,-1)$ from points on the x-axis. For the last objective, the level sets are arcs of circles centered outside the feasible region S at $(-2,-2)$.

For a representative sample of the behavior of the algorithm, the feasible region $S = B(0,2)$ was split into parts based on these level sets and a representative point from each part was selected. The results of the algorithm will be looked at in depth for the following 6 points:

$$p_1 = (-1,1), \qquad p_2 = (0.5,0.8), \qquad p_3 = (-1,-0.5),$$
$$p_4 = (-0.5,-1.5), \qquad p_5 = (0.3,-1.2) \quad \text{and} \quad p_6 = (1.1,-1.6).$$

Of these points, the last three are Pareto optimal solutions, while the first three are not. This result was correctly confirmed by the algorithm. For this problem, the algorithm even confirmed that the first three points are indeed not Pareto optimal (as opposed to giving the result that they are *possibly* not Pareto optimal). This was possible since in this case the points were not Pareto optimal for any subproblem, which gives the desired result, as discussed in the overview of the implementation.

Running the program showed that the latter three points were also not Pareto optimal for any subproblem with only 1 or 2 objectives. For subproblems with 3 objectives, the results are tabled below. In the table, the objective sets for which the point was Pareto optimal are marked with a $\times$-symbol. Note that the index sets $\{1,3,4\}$, $\{1,3,5\}$, $\{1,4,5\}$ and $\{3,4,5\}$ are absent from the table, since no point was Pareto optimal for any of these subproblems.

	$\{1,2,3\}$	$\{1,2,4\}$	$\{1,2,5\}$	$\{2,3,4\}$	$\{2,3,5\}$	$\{2,4,5\}$
p_4			$\times$		$\times$	$\times$
p_5	$\times$		$\times$	$\times$		$\times$
p_6		$\times$		$\times$		$\times$

Perhaps the most significant observation that is immediately observed from the table is the

column of marks for the subproblem with objectives $\{2,4,5\}$. The points p_4, p_5 and p_6 are all Pareto optimal for this subproblem. On the other hand, as previously mentioned, none of the points p_1, p_2 or p_3 are Pareto optimal for any subproblem, including this one. Thus the subproblem $\{2,4,5\}$ correctly classifies the Pareto optimality of all the points $p_1,\ldots,p_6$ considered. For this problem, this is also true in general. That is, for any point it is sufficient to solve only the subproblem with objectives $\{2,4,5\}$ to determine whether or not a point is Pareto optimal. Therefore, the objectives f_1 and f_3 are redundant.

7 Conclusions

The methods presented in this thesis approached the problem of dimension reduction in multiobjective optimization in a few different ways. Malinowska (2002, 2006) presented a method for the linear and convex cases to determine whether a single objective is redundant by studying the sets of Pareto optimal solutions. Malivert and Boissard (1994), and later Ehrgott and Nickel (2002), presented methods to find the Pareto front by solving only the Pareto fronts of subproblems with a limited number of objectives. Deb and Saxena (2005), Saxena and Deb (2007), and Brockhoff and Zitzler (2006b, 2006a), presented methods that can be used to determine which objectives to choose for further study when the values of the objectives on a finite set of points are known.

What is mainly lacking, is however the approach, where prior to the optimization process some objectives are deemed redundant. The only method of this type is found in the linear case, where Gal and Leberling (1977) presented a criterion based on linear relations within the objectives. It can of course be argued, that if in the optimization problem there are redundant objectives, then the problem has not been formulated properly. However, in practical problems, it is probably unreasonable to assume that the model for the problem is always minimal from the optimization point of view.

This observation naturally leads into the iterative structure of dimension reduction present in both Brockhoff and Zitzler's, and Deb and Saxena's, methods:

(1) Find some finite approximation for the set of Pareto optimal solutions.
(2) Perform some dimension reduction on the data given by the values of the objectives on this finite set of points to find a reduced set of objectives.
(3) Either repeat from step (1), considering only the new reduced set of objectives, or present the reduced data as the solution.

There are two main benefits to this type of approach. First, the computationally expensive optimizations of a high number of objectives need not be performed with the same level of detail as with a single-optimization approach. Second, since it may be difficult to select a suitable solution when the number of objectives is large, it is useful to reduce unnec-

essary complexity in the decision making process even if no further optimizations will be performed. The main drawback to this type of approach is possibly losing some relevant data in the reduction process, for example due to the imperfect nature of the approximation of the set of Pareto optimal solutions.

Even the implementation of Ehrgott and Nickel's algorithm presented in chapter 6 could be applied in this way, as seen from the example discussed in 6.3.2. In this example it was found that the original 5 objectives could be reduced to 3 objectives, which were observed to correctly determine Pareto optimality for any point. This means that when presenting the data, the remaining 2 objectives can safely be omitted and if further optimization is necessary, the calculations would be much simpler.

During the implementation of the aforementioned algorithm, it also became apparent that the algorithm never gave the result that it is unknown whether or not a point is Pareto optimal. In fact, the geometry of convex sets on the plane seem to imply that it may in fact not be possible that, for a (quasi)convex problem, a point is strongly Pareto optimal for a subset of three objectives, while not being Pareto optimal for any subproblem with a single fixed objective and two other arbitrary objectives. This is a problem for further research, since if this were indeed the case, then step (3) in the algorithm of Ehrgott and Nickel (see section 4.1.3) could be omitted without losing any generality. This would avoid the problem in the algorithm of determining when a Pareto optimal point is strongly Pareto optimal, which is computationally difficult.

Bibliography

Brockhoff, Dimo, Dhish Kumar Saxena, Kalyanmoy Deb, and Eckart Zitzler. 2008. “On handling a large number of objectives a posteriori and during optimization”. In *Multiobjective problem solving from nature,* 377–403. Springer.

Brockhoff, Dimo, and Eckart Zitzler. 2006a. “Dimensionality reduction in multiobjective optimization with (partial) dominance structure preservation: Generalized minimum objective subset problems”. *TIK-Report* 247.

———. 2006b. “On objective conflicts and objective reduction in multiple criteria optimization”. *TIK-Report* 243.

Chankong, Vira, and Yacov Y. Haimes. 1983. *Multiobjective decision making: theory and methodology.* North-Holland.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. *IEEE transactions on Evolutionary Computation* 6 (2): 182–197.

Deb, Kalyanmoy, and Dhish Kumar Saxena. 2005. *On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems.* KanGAL report 2005011. Indian Institute of Technology Kanpur.

Ehrgott, Matthias, and Stefan Nickel. 2002. “On the number of criteria needed to decide Pareto optimality”. *Mathematical Methods of Operations Research* 55 (3): 329–345.

Gal, Tomas, and Heiner Leberling. 1977. “Redundant objective functions in linear vector maximum problems and their determination”. *European Journal of Operational Research* 1 (3): 176–184.

Galas, Zenon, Ireneusz Nykowski, and Zbigniew Żółkiewski. 1987. *Programowanie wielokryterialne.* Państwowe Wydawnictwo Ekonomiczne.

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. 2001–. “SciPy: Open source scientific tools for Python”. `http://www.scipy.org/`.

Malinowska, Agnieszka Barbara. 2002. “Changes of the set of efficient solutions by extending the number of objectives and its evaluation”. *Control and Cybernetics* 31 (4): 965–974.

———. 2006. “Nonessential objective functions in linear multiobjective optimization problems”. *Control and Cybernetics* 35 (4): 873–880.

Malivert, C., and N. Boissard. 1994. “Structure of efficient sets for strictly quasi-convex objectives”. *Journal of Convex Analysis* 1 (2): 143–150.

Miettinen, Kaisa. 1999. *Nonlinear multiobjective optimization.* Kluwer academic publishers.

Oliphant, Travis E. 2007. “Python for scientific computing”. *Computing in Science & Engineering* 9 (3): 10–20.

Python Software Foundation. 2015. “The python language reference, version 2.7”. `http://www.python.org`.

Sawaragi, Yoshikazu, Hirotaka Nakayama, and Tetsuzo Tanino. 1985. *Theory of multiobjective optimization.* Academic Press.

Saxena, Dhish Kumar, and Kalyanmoy Deb. 2007. “Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: employing correntropy and a novel maximum variance unfolding”. In *Evolutionary Multi-Criterion Optimization,* 4403:772–787. Lecture Notes in Computer Science. Springer.

Steuer, Ralph E. 1986. *Multiple criteria optimization: theory, computation, and application.* Wiley.

Ward, James. 1989. “Structure of efficient sets for convex objectives”. *Mathematics of Operations Research* 14 (2): 249–257.

Weinberger, Kilian Q., and Lawrence K. Saul. 2006. “An introduction to nonlinear dimensionality reduction by maximum variance unfolding”. In *Proceedings of the 21st National Conference on Artificial Intelligence,* 1683–1686. AAAI.

Wierzbicki, Andrzej P. 1981. “A mathematical basis for satisficing decision making”. In *Organizations: Multiple Agents with Multiple Criteria,* 190:465–486. Lecture Notes in Economics and Mathematical Systems. Springer.

Xu, Jian-Wu, Puskal P. Pokharel, António R. C. Paiva, and José C. Príncipe. 2006. “Nonlinear component analysis based on correntropy”. In *International Joint Conference on Neural Networks,* 1851–1855. IEEE.

Appendices

A Python code of the implemented algorithm

```
from itertools import chain,combinations
from scipy import optimize
import numpy
from math import sqrt

def achscalar(f,x,x_0,rho,debug=False):
    """The achievement scalarizing function evaluated at x for the
    vector valued function f, using the basepoint x_0."""
    return max(f(x)-f(x_0))+rho*sum(f(x))

def minmaxopt(f, x_0, maxiter, tol, rho):
    """Find argmin_x max_i f_i(x)-f_i(x_0)
    for a function f=(f_1,...,f_n)."""
    def objf(x):
        return achscalar(f,x,x_0,rho)
    xmin = optimize.fmin(objf, x_0,
        maxfun=maxiter, xtol=tol, ftol=tol, disp=False)
    return xmin

def limitedPowerset(items, minsize, maxsize):
    """Returns a list containing all the subsets of the set 'items'
    that have between minsize and maxsize elements (inclusive)."""
    return list(chain.from_iterable(combinations(items, r)
        for r in range(minsize,maxsize+1)))

def isPareto(f,x,rho,checkall=False):
    """An implementation of Ehrgott and Nickel's algorithm to
    check when a point is Pareto optimal from
    Matthias Ehrgott and Stefan Nickel. On the number of criteria
```

```
needed to decide pareto optimality. Mathematical Methods of
Operations Research, 55(3):329-345, 2002.

f: The objective vector function of arbitrary dimension
    accepting a 2 dimensional variable.
x: The point whose Pareto optimality is being checked.
rho: The parameter used in the achievement scalarizing function
    to distinguish Pareto optimal points from weakly Pareto
    optimal points.
checkall: If True, will check Pareto optimality for all
    subproblems. Otherwise ends as soon as point is known
    to be Pareto optimal.

Returns 1, if the point is Pareto optimal for the problem,
        0, if it is unknown whether the point is Pareto optimal
        -1, if the point is not Pareto optimal.
"""
maxiter = 1000
tol = 10**-6
f0 = f(x)

# Select all the indexsets of objectives for which the Pareto
# optimality of x may need to be checked.
S = set(range(len(f0)))
Q = limitedPowerset(S,1,3)

# Initialize helper variables.
Isets = []
Iunion = set()
paretofound = False

# Loop through the indexsets.
for I in Q:
```

```
        # Define a new objective vector function from the reduced
        # problem corresponding to the objectives in I.
        def fI(y):
            fy=f(y)
            return numpy.array([fy[j] for j in I])

        # Optimize the subproblem.
        xmin = minmaxopt(fI, x, maxiter, tol, rho)
        paropt = \
            achscalar(fI,xmin,x,rho) >= achscalar(fI,x,x,rho) - tol \
            and sqrt(sum((xmin - x)**2)) < tol
        if paropt:
            print "Pareto optimality for subproblem",I,":",paropt
        if paropt:
            paretofound = True
            Isets.append(I)
            Iunion = Iunion | set(I)
            if (not checkall) and Iunion == S:
                # All objectives accounted for.
                # x is Pareto optimal for the entire problem.
                return 1
    if checkall and Iunion == S:
        return 1

    # Some objectives have not been accounted for.

    if paretofound:
        # x was Pareto optimal for at least 1 subproblem
        # if it was also strongly Pareto optimal, then
        # x may not be Pareto optimal for the entire problem.
        return 0

    # x was not Pareto optimal for any subproblem,
```

```
    # thus it is not Pareto optimal for the entire problem.
    return -1
```

B Usage example for the implementation

```
from datetime import datetime
import numpy
import random

if __name__ == '__main__':
    # Define the objective vector function.
    def testfunc(x):
        return numpy.array([x[0],x[1],-2*x[0]-x[1],x[0]+x[1]])

    # parameter for the achievement scalarizing function
    rho = 10**-6

    # The points to be checked.
    xlist = [numpy.array([(random.random()-0.5)*20,(random.random()-0.5)*20]
             for j in range(100)]

    for i,x in enumerate(xlist):
        print "\n---\nTest %d: Pareto optimality of %s\n---" % (i,x)
        prev = datetime.now()
        res = isPareto(testfunc,x,rho)
        cur = datetime.now()
        print "\nAlgorithm complete in %f seconds."\
            %((cur-prev).total_seconds())
        print "The point",x,("is" if res == 1 else\
            ("might not be" if res == 0 else "is not")),\
            "Pareto optimal."
```