Diss. ETH № 20631

Exploring Structural Diversity in Evolutionary Algorithms

A dissertation submitted to
ETH Zurich

for the degree of

Doctor of Sciences

presented by

Tamara Ulrich

MSc ETH in Electrical Engineering and Information Technology
born September 2, 1983
citizen of Küssnacht, SZ

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

2012

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 133

Tamara Ulrich

Exploring Structural Diversity in Evolutionary Algorithms

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to
ETH Zurich
for the degree of Doctor of Sciences

Diss. ETH № 20631

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

Examination date: September 27, 2012

Contents

Abstract

Optimization problems arise in many different contexts and applications. For each optimization problem, there is a so-called decision space that contains all feasible solutions to the problem. Additionally there are one or several objective functions that quantify how well each solution satisfies the given objectives. The goal of optimization algorithms for single-objective problems is to find the global optimum, i.e. one or several solutions that have the best objective value. In multi-objective problems, on the other hand, there is no single best solution, but a set of tradeoff solutions, the so-called Pareto-front. Multi-objective optimizers therefore aim at finding that front, or a subset of it.

To find the global optimum or the Pareto-front, either analytical methods or exhaustive search can be employed. Sometimes though, the decision space is too large for exhaustive search, and the type of problem is not suitable for analytical methods. In such cases, Evolutionary Algorithms (EAs) are often used to approximate the best solutions. EAs mimic natural evolution by evolving sets of solutions in iterations, where in each iteration, new solutions are created by combining or modifying the current solutions, and the best solutions are kept and enter the next iteration.

When optimizing real-world problems, a model is needed that presents the optimization problem in such a way that an EA can optimize it. Often, there are simplifications and uncertainties in these models. Therefore, not only optimal, but also close-to-optimal solutions are of interest. Moreover, a user may not be satisfied with a single solution, but instead wants to gain insights into the problem. In this case it is advantageous to present the user with structurally diverse solutions, i.e. solutions which are diverse in decision space. Therefore, this thesis tackles the problem of generating a set of solutions which has a high structural diversity, but whose solutions at the same time have acceptable objective values.

Also, it is useful to have methods that support analyzing the optimized set, in order to help the user to identify the characteristics that lead to high

quality solutions, and in the case of multi-objective problems, the characteristics that cause the solutions to lie in a certain region in objective space. This thesis provides methods to analyze these optimized sets by clustering the solutions and highlighting the similarities of the solutions of each cluster.

Finally, this thesis investigates the effectiveness of the hypervolume indicator, which in this thesis is the main measure of objective space goodness in multi-objective problems. The hypervolume indicator basically measures how well a set of solutions approximates the Pareto-front. An algorithm is effective if it can reach the set with the optimal hypervolume on any optimization problem. If there are optimization problems where the algorithm cannot reach the best set, the question arises how far the best achievable hypervolume is from the theoretically optimal hypervolume.

More precisely, this thesis makes the following main contributions:

- It proposes three diversity-optimizing EAs, one for single-objective problems and two for multi-objective problems, and compares their performance on different problems, including a bridge optimization problem.
- It proposes two methods to analyze optimized sets of solutions, one specifically designed to tackle binary, biobjective problems, and the other designed to tackle problems with an arbitrary number of objectives and general decision spaces.
- It derives upper and lower bounds on how far the best hypervolume achieved by an EA is from the theoretically optimal hypervolume. These bounds hold for any optimization problem, and are tighter than the bounds previously known in the literature.

Zusammenfassung

Optimierungsprobleme treten in verschiedensten Gebieten auf. Zu jedem Optimierungsproblem gehört ein Entscheidungsraum, welcher die gültigen Lösungen des Problems enthält. Zusätzlich gibt es eine oder mehrere Zielfunktionen, welche ausdrücken wie gut eine einzelne Lösung die gegebenen Optimierungsziele erfüllt. In Einzielproblemen soll ein Optimierungsalgorithmus das globale Optimum finden, d.h eine oder mehrere Lösungen, welche den besten Zielfunktionswert haben. In Mehrzielproblemen hingegen gibt es nicht eine beste Lösung, sondern eine Menge von Kompromisslösungen, welche zusammen die so genannte Pareto-Front bilden. Optimierer für Mehrzielprobleme sollen deshalb diese Front, oder eine Teilmenge davon, finden.

Um das globale Optimum oder die Pareto-Front zu finden können entweder analytische Methoden oder eine vollständige Suche verwendet werden, wobei die vollständige Suche alle Lösungen im Entscheidungsraum vergleicht. Manchmal ist der Entscheidungsraum jedoch zu gross für eine vollständige Suche, und die Art des Problems lässt keine analytischen Methoden zu. In solchen Fällen werden oft Evolutionäre Algorithmen (EAs) verwendet, um die besten Lösungen zu approximieren. EAs imitieren die natürliche Evolution indem sie Mengen von Lösungen in Schritten optimieren, wobei in jedem Schritt neue Lösungen aus den bisherigen erzeugt werden, indem die bisherigen Lösungen kombiniert oder leicht verändert werden. Dabei werden jeweils die besten Lösungen ausgewählt, um die bisherigen zu ersetzen.

Bei der Optimierung von realen Problemen wird meist ein Modell benötigt, welches das Optimierungsproblem so darstellt, dass es mittels eines EAs optimiert werden kann. Oft gibt es in einem solchen Modell Vereinfachungen und Unsicherheiten. Deshalb sind nicht nur optimale, sondern auch leicht suboptimale Lösungen von Interesse. Zusätzlich kann es sein, dass der Benutzer nicht nur an einer einzelnen Lösung interessiert ist, sondern dass er durch die Optimierung einen Einblick in das Problem selbst erhalten möchte. In einem solchen Fall ist es von Vorteil, wenn der Algorithmus dem Benutzer strukturell unterschiedliche Lösungen präsentieren kann, also Lö-

sungen, welche im Entscheidungsraum divers sind. Diese Arbeit untersucht deshalb das Problem, Lösungen zu finden, welche strukturell divers sind, aber trotzdem gute Zielfunktionswerte besitzen.

Nützlich sind auch Methoden welche es erlauben, eine optimierte Menge von Lösungen zu untersuchen. So sollen Charakteristiken gefunden werden, welche zu guten Lösungen führen, oder im Fall von Mehrzielproblemen Charakteristiken, welche einen Einfluss auf die Position der Lösung im Zielfunktionsraum haben. Diese Arbeit stellt deshalb Methoden vor, welche optimierte Lösungsmengen analysieren indem die Lösungen gruppiert und die Gemeinsamkeiten der Lösungen in einer Gruppe hervorgehoben werden.

Die vorliegende Arbeit untersucht auch die Effektivität des Hypervolumenindikators. Der Hypervolumenindikator ist in dieser Arbeit das hauptsächlich verwendete Mass, um die Güte einer Lösungsmenge im Zielfunktionsraum zu berechnen. Im Wesentlichen misst der Indikator die Grösse des dominierten Bereiches in Mehrzielproblemen. Ein Algorithmus wird effektiv genannt, wenn er die Lösungsmenge mit dem besten Hypervolumen in jedem beliebigen Optimierungsproblem erreichen kann. Falls es Optimierungsprobleme gibt, in welchen der Algorithmus die beste Menge nicht erreichen kann stellt sich die Frage, wie nah das beste erreichbare Hypervolumen dem theoretisch besten Hypervolumen kommt.

Konkret liefert die vorliegende Arbeit die folgenden Beiträge:

- Sie schlägt drei Diversitäts-optimierende EAs vor, einer für Einzielprobleme und zwei für Mehrzielprobleme, und vergleicht diese auf verschiedenen Optimierungsproblemen, insbesondere auf einem Brückenproblem.
- Sie schlägt zwei Methoden vor um Lösungsmengen zu analysieren, eine für binäre Entscheidungsräume und zweidimensionale Zielfunktionsräume, und die andere für beliebige Entscheidungsräume und beliebig viele Zielfunktionen.
- Sie findet eine Unter- und eine Obergrenze dafür, wie weit das beste erreichbare Hypervolumen eines EA vom theoretisch bestmöglichen Hypervolumen entfernt ist. Diese Grenzen gelten für alle Optimierungsprobleme, und sind enger als die bisher in der Literatur bekannten Grenzen.

Statement of Contributions

Parts of this thesis has been already published in journal articles and conference proceedings. However, some approaches, experiments and results have not yet been published. Additionally, the whole content of the thesis has been revised and partly rewritten.

With the exception of the hill climber leading to Figure 2.1, all the experiments and implementation work in the original papers and the thesis were done by myself. Also, all illustrations in this thesis have been created by myself (with the exception of Figure 2.1), and have been adapted to fit the style of this thesis. My contribution to the writing of a each contributing paper was at least $1/n$, n denoting the number of authors.

In detail, the publications behind the individual chapters of this thesis are as follows:

Chapter 1 The entire chapter including the illustrations have been created from scratch for this thesis.

Chapter 2 This chapter is based on the work published in [107, 110, 111]. All experiments have been redone for this thesis. Also, the thesis contains more extensive tests than the original papers. Algorithmic changes with respect to the original papers are as follows: DIOP is used without the weighted sum as its fitness measure (as indicated in the corresponding paragraph), and DIVA has been adapted to include the Solow-Polasky diversity measure.

Chapter 3 This chapter is based on [109] and [106].

Chapter 4 This chapter is based on [108].

Tamara Ulrich

Acknowledgments

First of all, I thank my advisors, Lothar Thiele and Eckart Zitzler for the valuable inputs and ideas, as well as for the honest evaluation of my work. I also thank my co-examiner, Kalyanmoy Deb, for taking the time to correct this thesis. Further thanks go to my coauthors Dimo Brockhoff, Johannes Bader, Eckart Zitzler and Lothar Thiele, for teaching and supporting me in writing my papers.

Furthermore I thank all my colleagues from TIK, especially the former SOPsies Dimo Brockhoff, Johannes Bader, and Tim Hohm, for countless discussions about problems I encountered during my thesis. I especially thank Dimo Brockhoff and Eckart Zitzler for introducing me to SOP by supervising my master thesis.

Thanks also go to my colleagues from Bosch, Ralph Moritz, Susanne Bürklen, Robert Kornhaas and Markus Behle, for giving me the opportunity to work on a real-world problem. Many of the topics covered in this thesis were motivated by this optimization problem.

Finally, I thank my own moral support team, my parents, my brother, and Roman, who encouraged me in challenging times and celebrated with me the highlights of my time at ETH.

List of Symbols and Abbreviations

The following notation and abbreviations are used in this thesis:

Pareto-Set Analysis

$\mathcal{M}_{n,d}(\{0,1\})$	Set of binary matrices with n rows and d columns, as used in MANA, page 94
Υ	Module matrix as used in MANA, page 95
Ξ	Decision matrix as used in MANA, page 94
C	Partitioning, i.e. a set of clusters, page 114
c_i	Cluster, i.e. a subset of solutions, page 114
d_D	Distance in decision space, page 113
d_O	Distance in objective space, page 113
d_{H}	Hamming distance, page 99
$e_{\mathrm{dec}}, e_{\mathrm{obj}}$	Error functions as used in MANA, page 97
G	Group, i.e. a subset of solutions as used in MANA, page 99
k	Number of clusters, page 114
$m(c_i)$	Medoid of cluster c_i, page 118
S_i	Module as used in MANA, page 94
$T_{\Upsilon\to\Xi}$	Transformation function between module matrix and decision matrix as used in MANA, page 96
$T_{\Xi\to\Upsilon}$	Transformation function between decision matrix and module matrix as used in MANA, page 95
V	Validity index as used in PAN, page 114
x^i	Decision vector as used in MANA, page 94
y^r	Module vector as used in MANA, page 96

Diversity

$\mathcal{A}$	Archive population used in DIOP, page 47
$\mathrm{D}^{\mathcal{A}}(\mathcal{T},\varepsilon)$	Constrained diversity measure used in DIOP, page 48
$\mathrm{dom}_A(z)$	Subset of $\mathcal{A}$ dominating the objective vector z, as used in DIVA, page 63
$\preccurlyeq_D$	Diversity preference as used in DIVA, page 63
$\mathcal{T}$	Target population used in DIOP, page 46

θ	Normalization parameter for Solow-Polasky diversity measure, page 26
b	Bound used in NOAH, page 34
c	Termination criterion for diversity optimization in NOAH, page 35
D	Diversity measure, page 21
d	Distance measure, page 21
e^T	Transpose of vector e, page 26
I_H^{D}	Diversity integrating hypervolume, as used in DIVA, page 64
M	Matrix of normalized pairwise distances, page 26
$q_{\mathcal{X}^*}$	Measure that quantifies distance to Pareto-optimal set, page 46
r	Number of solutions being kept in the population during BOUNDCHANGE in NOAH, page 35
v	Barrier value used in NOAH, page 32

General

$\emptyset$	Empty set, page 62
$\mathbb{N}$	Set of natural numbers, page 153
$\mathbb{R}$	Set of real numbers, page 4
$\leq$	Comparing scalars, left scalar is smaller or equal to right scalar, page 4
$\leqslant$	Comparing vectors, all elements of left vector are smaller or equal to corresponding element of right vector, page 4
$\mathcal{P}, \mathcal{A}, \mathcal{B}$	Sets of solutions, page 154

Hypervolume Effectiveness

$\mathcal{R}$	Reference set, page 4
$A(\mathcal{P}, \mathcal{O})$	Archiving algorithm, page 152
$A_{\mathcal{P}}(y)$	Attainment function of set $\mathcal{P}$, page 4
I_H	Hypervolume indicator, page 4
$I_H^{\max}$	Maximum achievable hypervolume, page 149
$I_{H,\mu}^{\max}$	Maximum achievable hypervolume for a population of size μ, page 153

Multiobjective Optimization

$2^{\mathcal{X}}$	Powerset of the decision space, page 5
λ	Offspring size, page 11
μ	Population size, page 11
$\prec$	Pareto dominance, page 4
$\preceq$	Weak Pareto dominance, page 4

$\preceq_\varepsilon$	Weak ε-Pareto dominance, page 46
$\mathcal{X}$	Decision space, page 3
$\mathcal{X}^*$	Pareto-optimal set, page 46
$\mathcal{Y}$	Objective space, page 4
d	Dimensionality of decision space if decision variables are used, page 50
f	Vector of m objective functions, page 3
f_i	i-th objective function, page 3
g	Maximum number of generations, page 34
m	Number of objective functions, page 3
p_R	Recombination probability, page 129

EA Evolutionary Algorithm
NOAH the Diversity-optimizing Single-objective Evolutionary Algorithm
DIVA the Diversity-integrating Multi-objective Evolutionary Algorithm
DIOP the Diversity-optimizing Multi-objective Evolutionary Algorithm
sMOEA the Standard Multi-objective Evolutionary Algorithm
OMNI the Omni-Optimizer
NSGA-II the Nondominated Sorting Genetic Algorithm
WFG Walking Fish Group
ECU electronic control unit
PAN the Pareto-Front Analyzer
MANA the Module-Annotating Hierarchical Clustering Algorithm
SPEA2 the Modified Strength Pareto Evolutionary Algorithm
IBEA the Indicator-Based Evolutionary Algorithm

1

Introduction

Many real-world problems aim at optimizing several conflicting objectives. Consider for example a car manufacturing problem, where the task is to design the car's electric and electronic (E/E) architecture. Designing an E/E-architecture consists of multiple steps, ranging from assigning a given set of components (sensors, actuators and software) to electronic control units (ECUs), to placing those ECUs in the car and connecting them via busses, selecting gateways to connect the busses, selecting microcontrollers for the ECUs, and realizing the bus structure using physical wires. The goal is to find an architecture which is as cheap as possible but at the same time has a complexity which is as low as possible. Complexity here is defined as the average number of components per ECU, whereas the cost is governed mainly by the wiring cost. Cost and complexity are also called *objectives*, and in this case, they are conflicting, as an architecture containing only a few ECUs with many components has most communication between components handled within the ECUs which reduces cost, whereas an architecture with many ECUs containing only one component has a low complexity.

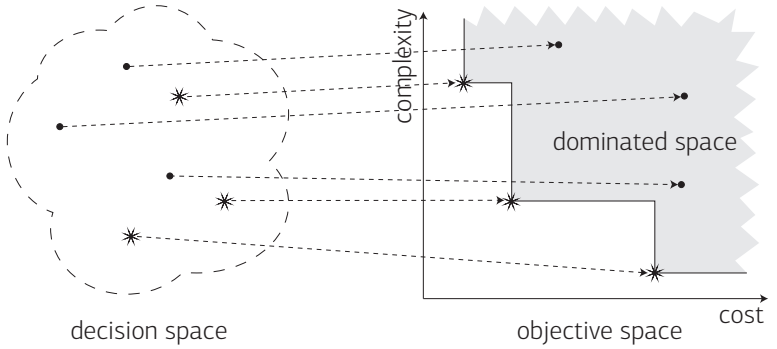


Figure 1.1 Biobjective optimization problem where the objectives to be minimized are cost and complexity. Both the decision and the objective space are shown. Pareto-optimal solutions are indicated with a star, dominated solutions with a circle. Also, the area dominated by the Pareto-optimal solutions is shown (gray area).

This E/E-architecture design problem is an example of a *multi-objective optimization problem*, see Figure 1.1 for an illustration. A specific architecture is also called a *solution* to the car manufacturing problem. The space of all possible architectures is called the *decision space*. The image of the decision space under the given objectives is called the *objective space*, which in the car manufacturing problem is a two dimensional real-valued space, where each architecture is assigned a vector corresponding to its values in the two objectives. As these objectives are conflicting, there is no single best architecture minimizing both objectives. Instead, there is a set of tradeoff architectures, the so-called *Pareto-optimal set* or *Pareto set*. The image of the Pareto set under the objectives is called the *Pareto-optimal front* or *Pareto front*. It is usually not possible to find the exact Pareto front, mostly because the problems are complex such that analytical solvers cannot be used, and the size of the decision space does not allow complete enumeration. Therefore, most approaches aim at approximating the Pareto front.

One way to approximate the Pareto front is to use an Evolutionary Algorithm (EA). They mimic biological evolution, one part of which is that they do not evolve single solutions, but sets of solutions. In accordance

with biology, these sets are called *populations*. Because EAs optimize sets of solutions, they are well suited to solve multi-objective optimization problems, where an approximation of the Pareto front is sought. Furthermore, they are very general in nature and can therefore be applied to a variety of problems.

During this thesis, an application scenario of an E/E-architecture problem was jointly investigated with BOSCH. A short overview of the problem including a description of the model we developed for the EA is presented in [72]. Several research questions considered in this thesis were inspired by this project.

This chapter is organized as follows: First, there will be a more detailed introduction into multi-objective optimization, including the notation used throughout the thesis. Then, a description of EAs will be given. And finally, the research questions of this thesis will be posed and the contributions will be outlined.

1.1 · Multi-objective Optimization

Consider a two-objective car manufacturing problem, where the first goal is to minimize the cost, and the second goal is to minimize the complexity, as illustrated in Figure 1.1. As these objectives are conflicting, there is not one single car being best in both objectives. Instead, there are three Pareto-optimal cars. The first one is the cheapest car, the second one is a bit more expensive, but has a lower complexity than the cheapest car, and the third one has the lowest complexity. All three cars are *non-dominated*, meaning that there is no car which is both cheaper and less complex. The set of all non-dominated solutions is also called the Pareto-optimal front, or Pareto front. Multi-objective optimizers aim at finding that Pareto front, or, if finding it is not possible, approximating it in a suitable manner.

The notation used throughout this thesis is as follows: We are considering the minimization of m objective functions $f : \mathcal{X} \to \mathcal{Y}$, $f = \{f_1, ..., f_m\}$. Here, $\mathcal{X}$ denotes the feasible set of solutions in the decision space, i.e. the

set of alternatives of the optimization problem. In the car problem, this would be the set of all possible E/E-architectures. Note that without loss of generality, every maximization problem can be transformed into a minimization problem. A single alternative $x \in \mathcal{X}$, i.e. a single E/E-architecture will be denoted as a solution x. The image of $\mathcal{X}$ under f is denoted as the feasible set in the objective space $\mathcal{Y} = f(\mathcal{X}) = \{y \in \mathbb{R}^m \mid \exists x \in \mathcal{X} : y = f(x)\}$. Therefore, the objective vector of a single solution x is $f(x) = \{f_1(x), ..., f_m(x)\}$.

The underlying preference relation is weak Pareto dominance, where a solution $a \in \mathcal{X}$ weakly dominates another solution $b \in \mathcal{X}$, denoted $a \preceq b$, if and only if solution a is better or equal than b in all objectives, i.e., $a \preceq b$ iff $f(a) \leqslant f(b)$ or equivalently, iff $f_i(a) \leq f_i(b)$, $\forall\ i \in \{1, ..., m\}$. A solution $a \in \mathcal{X}$ strictly dominates another solution $b \in \mathcal{X}$, denoted $a \prec b$, if and only if solution a is better or equal than b in all objectives and strictly better in at least one objective, i.e., $a \prec b$ iff $f(a) \leqslant f(b) \wedge \exists i \in \{1, ..., m\}$ s.t. $f_i(a) < f_i(b)$.

Ideally, a multi-objective optimizer finds the whole Pareto-front, i.e. all solutions which are not dominated by any other solution in $\mathcal{X}$. However, as the size of the Pareto-front is not bounded in general, a subset of fixed size of the whole Pareto-front is usually sought. The question therefore arises which subset of the Pareto-front is the best. One approach to solve this problem is to use a quality indicator $I : 2^{\mathcal{X}} \rightarrow \mathbb{R}$, which assigns each subset of the decision space a real number that indicates the quality of the set. The indicator used in this thesis is the hypervolume indicator, where the hypervolume indicator of a given set $\mathcal{P} \subseteq \mathcal{X}$ is the volume of all points in $\mathbb{R}^m$ which are dominated by at least one point in $\mathcal{P}$ and which dominate at least one point of a reference set $\mathcal{R} \subset \mathbb{R}^m$. Roughly speaking, the hypervolume measures the size of the dominated space of a given set. Sets with a larger hypervolume are considered better. More formally, the hypervolume indicator can be written as $I_H(\mathcal{A}) = \int_{y \in \mathbb{R}^m} A_{\mathcal{P}}(y) dy$, where $A_{\mathcal{P}}(y)$ is called the attainment function of set $\mathcal{P}$ with respect to a given reference set $\mathcal{R}$, and it holds that $A_{\mathcal{P}}(y) = 1$ iff $\exists p \in \mathcal{P}, r \in \mathcal{R} : f(p) \leqslant y \leqslant r$, else $A_{\mathcal{P}}(y) = 0$.

Chapter 2 of this thesis proposes methods to optimize sets of solutions which are both Pareto-optimal and structurally diverse. Structural in this case means that the diversity is measured in decision space, not in objective space. We therefore need a measure to quantify diversity in decision space. Most diversity measures require that some measure of dissimilarity between two arbitrary solutions is provided. Therefore, we assume that a symmetric distance measure $d : \mathcal{X}^2 \to \mathbb{R}$ between two solutions is given. Based on the distance measure we define a diversity measure $D : 2^{\mathcal{X}} \to \mathbb{R}$, where $2^{\mathcal{X}}$ is the powerset of the decision space, i.e. all possible subsets of the decision space. The measure determines for a subset of the decision space its corresponding diversity, which in turn directly depends on the chosen distance measure. Unless otherwise stated, we do not make any other assumptions about the decision space.

1.2 · Evolutionary Algorithms

In this thesis we are dealing with problems which are neither convex nor linear, and cannot be solved optimally. Instead, we would like to find an approximation of the unknown Pareto front. One family of algorithms designed to do exactly that are EAs. EAs have been invented in the 60s and early 70s [10, 51, 86, 94], and have since been applied successfully to a variety of problems, from car manufacturing [47] to aircraft layout [78], antenna design [2] and space mission design [93].

EAs work by mimicking biological evolution, see Figure 1.2. They operate in cycles called *generations*, and maintain a set of solutions, called *population*. At first, the population is initialized with random solutions. In each generation, new offspring are generated from the current population, a process called *variation*, and then, a subset of the current population and the offspring is selected to survive into the next generation, a process called *environmental selection*. During variation, first the solutions which will be used for reproduction have to be selected, a process called *mating selection*. The selected solutions are first *recombined* using a *crossover operator*, and then *mutated* using a *mutation operator*. The goal of recombination is to

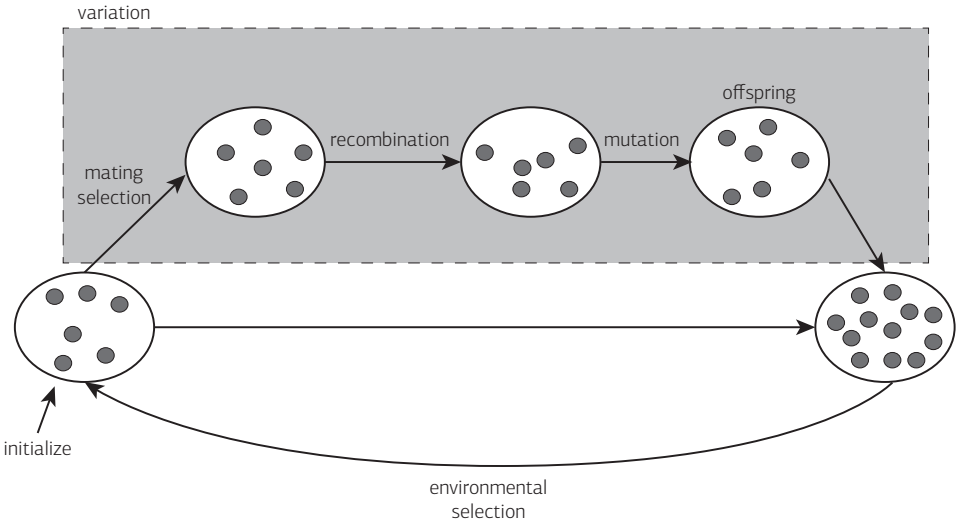


Figure 1.2 General principle of an evolutionary algorithm. They evolve populations (large white circles) of solutions (small gray circles), and alternate between variation and environmental selection. Both the current population (leftmost population) and the offspring are considered for environmental selection

find better solutions by mixing traits of good solutions. The goal of mutation is finding better solutions in the vicinity of good solutions. While the variation phase has to be chosen appropriately for the problem at hand, the selection process is problem independent.

EAs can be used to solve single- as well as multi-objective problems. In single-objective problems, each solution is assigned a *fitness* value, which corresponds to the objective function value of that solution. Unless two solutions have the same fitness, it is always clear which one of two solutions is better. A straight-forward approach therefore would be to always prefer better solutions to worse ones during selection. However, it has been found that following this approach leads to a population containing only copies or slight variations of the best solution, which in turn often leads the algorithm into a local optimum from which it cannot escape. One of the challenges

in designing an EA for a single-objective problem therefore is to maintain diversity in the population while still progressing towards better solutions.

In multi-objective problems [23], on the other hand, there is not a single best solution. Instead, there is a whole set of Pareto-optimal solutions. An algorithm that approximates the Pareto front will therefore inherently be able to maintain some diversity in its population. However, in contrast to single-objective problems, the objective values do not yield a total order of the solutions anymore. The concept of Pareto-dominance implies a partial order on the solutions, but there can be solutions which are *incomparable*, where no solution dominates any other. In this case, it is not quite clear which solutions should be chosen during the selection process.

1.3 · Research Questions

As stated before, it is usually not possible to find the exact Pareto front, therefore we are interested in a good approximation of the front. However, it is not intuitively clear what makes a good approximation. Usually, the quality of a set is determined in objective space, in which case the goal is to have solutions which are (a) close to the true Pareto front, and (b) well distributed in terms of their objective values. One way to measure both of these goals for a given set in one single goodness value is to use the hypervolume indicator [121]. The hypervolume indicator basically measures the size of the dominated objective space, see Figure 1.1. The higher the hypervolume, the better the set.

However, if the results are going to be used in a real-world scenario, e.g. when an engineer is using an EA to determine a good set of E/E-architectures in an automotive design problem, it is not only useful to have solutions that cover the front well, but it is also beneficial to have solutions which are structurally diverse. Sometimes, it would be even more interesting for an engineer to find an architecture which e.g. is a bit more expensive and has a slightly higher complexity than another architecture, but at the same time

has a completely different bus structure from all the other architectures which have already been found.

Therefore, the first research question of this thesis focuses on not only approximating the Pareto front well, but also optimizing the structural diversity of the population. When starting an EA, the randomly chosen initial population has a high structural diversity. During the run, the population converges to Pareto-optimal regions of the decision space, and therefore, diversity decreases. This leads to a tradeoff between converging to the Pareto-optimal front, and optimizing structural diversity. The question is how to find a population with both an acceptable quality in objective space and a high structural diversity, and how to decide on a tradeoff between these two conflicting goals. Consider the situation depicted in Figure 1.3, where you have to decide between two solutions b and c, both being incomparable to the two already selected solutions a and d. There is a tradeoff, as b dominates c, but adding c would increase the diversity more. Up to now, the quality in objective space has been the priority, although some approaches do use structural diversity as a selection criterion if they have to select between two solutions that do not dominate each other. If the quality in objective space is more important than the diversity, b will be chosen, although for an engineer, knowing about c might be more interesting if b is very similar to a and d, and c, on the other hand, is very dissimilar. Therefore, if neither the quality in objective space nor the diversity is given strict priority, it is not clear how the tradeoff should be set.

The second research question tackles the *a posteriori* problem of learning, once an approximation of the Pareto front is found. Interpreting a given set of solutions can take a lot of time, especially if there are many objectives, many solutions, and if a single solution is difficult to visualize. For example in the E/E-architecture problem, it is difficult to visualize a solution, as this includes plotting between 80 and 150 ECUs, showing how they are connected via busses, how these busses are wired, etc. Still, a decision maker might want to learn about the problem, such as what type of architectures lead to Pareto-optimal solutions, and what architectures lead to what region in objective space. Depending on the problem at hand, it could even be

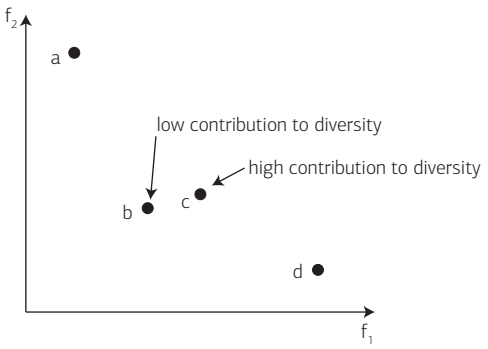


Figure 1.3 Tradeoff between quality in objective space and diversity. Four solutions a, b, c, and d are shown, where b dominates c, but c has a higher contribution to diversity.

possible to extract certain characteristics in decision space which influence in which objective space region a solution lies. For example it might be possible that architectures which contain Flexray busses are less complex but more expensive than cars containing only CAN busses. Therefore, we need ways to represent the final solutions to the engineer such that it is easier for him to learn from the optimization problem.

All EAs proposed in this thesis use the hypervolume indicator to quantify the quality of a set of solutions in objective space. By using the hypervolume indicator, the multi-objective problem of selecting a set of solutions which is close to the Pareto front and well distributed in objective space transforms to a set problem, where the set of solutions which maximizes the hypervolume indicator is sought. Ideally, an algorithm will find the set of a given size n with the maximum hypervolume value. In practice, however, the best hypervolume which can be achieved by a specific EA might be lower than the maximal hypervolume over all possible sets of size n. Therefore, the third research question is about determining the *effectiveness* of hypervolume-based algorithms. An algorithm is called effective if it can be proven that the algorithm can find the set of size n with the maximum hypervolume. An algorithm is called α-approximate, if it can be proven that it can find a set of size n with a hypervolume which is at least $1/\alpha$ times the maximum hypervolume. If an algorithm is not effective, we would

like to find optimization problems where the algorithm cannot reach the set with maximal hypervolume, and we would like to derive upper and lower bounds on α, i.e. on how close the hypervolume of the best set of solutions found by the algorithm will be to the maximum hypervolume.

1.4 · Contributions and Overview

This thesis proposes methods and answers to the above three problems. First, three approaches are presented that maintain structural diversity during the search and find a tradeoff between quality in objective space and structural diversity. The first one tackles single-objective problems. It only optimizes according to structural diversity, but at the same time has a constraint on the minimum quality in terms of the objective function. This constraint is loose at first, and becomes tighter during the search to ensure a good quality in objective space. The second one follows a similar approach, but for multi-objective problems. To do so, two populations are evolved simultaneously, one which is optimized according to the hypervolume indicator, and the other which is optimized according to structural diversity, but with the constraint of staying within a certain distance in objective space to the solutions in the first population. This constraint also becomes tighter during the run, as the first population progresses towards the Pareto front. The third approach is also aimed at multi-objective optimization, but instead of simply having a constraint on the quality in objective space, and optimizing for structural diversity of feasible solutions, this approach aims at having solutions well distributed in both decision and objective space, as well as being close to the front. It does so by integrating the diversity of the solutions into the hypervolume indicator.

To answer the second research question of how to present the solutions to the decision maker, we propose two approaches, one aimed at binary biobjective problems, and the other aimed at general multi-objective problems. The first method uses biclustering to find so-called *modules* of similar settings of the binary decision variables. Those modules are then used to cluster the solutions in objective space. As a result, the decision maker can learn about

what decision variable setting leads to what region in objective space. This approach tackles the two problems of interpreting large sets of solutions, and of interpreting solutions which are difficult to visualize because there are many decision variables, as each solution can be expressed in terms of the chosen modules. The second approach does not make any assumptions about the decision space, only that a distance measure between solutions is given. It then simultaneously clusters the solutions in decision and objective space, such that the resulting clusters contain similar solutions (according to the chosen distance measure in decision space) which lie close in objective space. For each cluster, a representative solution is selected. In order to learn about what types of solutions lead to what regions in objective space, the decision maker only has to look at one solution per cluster, knowing that the other solutions in the cluster are similar to the representative solution. It therefore helps solving the problem of interpreting a large population in a general setting.

For the third research question, we consider general $(\mu + \lambda)$-evolutionary algorithms, where μ is the population size and λ is the number of offspring. The $+$ here means that both the current population and the offspring generated from it are considered during selection. We investigate how far the achievable hypervolume is from the theoretically optimal hypervolume, and derive upper and lower bounds for the achievable hypervolume. The bounds presented in this thesis are tighter than the bounds previously known in the literature.

2

Maintaining Structural Diversity During Optimization

Typically, optimization attempts to find a solution that minimizes one or possibly several given objective functions. But often, it might also be useful to obtain a set of structurally diverse solutions which all have acceptable objective values. With such a set, a decision maker would be given a choice of diverse solutions to select from. In addition, the decision maker can learn about the optimization problem at hand by inspecting the diverse close-to-optimal solutions.

This chapter addresses the problem of simultaneously optimizing the structural diversity and the objective values of a set of solutions. Section 2.1 gives some motivation of why structural diversity optimization can be useful. Section 2.2 formally states the requirements a diversity measure should fulfill, and it reviews some of the more commonly used diversity measures. Finally, Sections 2.3, 2.4, and 2.5 discuss three approaches, one aimed at single-objective optimization and two aimed at multi-objective optimization.

2.1 · Motivation and Background

Consider the case that an engineer wants to design the electronic system in a car. The engineer is given a fixed cap on the cost that must be satisfied. There are a few standard designs which the engineer could use, e.g. a centralized system where each subsystem is controlled by a central processor, or a distributed design where each subsystem has its own processor. Nevertheless, the engineer would like to know whether there are any other, possibly non-standard designs that satisfy the cost cap, such that the engineer can then select the design which can best be integrated into the given car family. To this end, an algorithm is required that returns a set of designs (i.e. solutions) that are structurally as diverse as possible, but still satisfy the cost cap.

Another engineer might want to create a truss bridge that is as cheap as possible but at the same time can carry as much load as possible. The engineer used a multi-objective optimizer to approximate the Pareto-front, and found that the optimized set consists of bridges with similar truss structures, but different truss thicknesses. The whole Pareto-front can be covered by such designs, as thickening the trusses increases both the load the bridge can carry as well as its cost. Nevertheless, the engineer might want to know whether there are some other, dissimilar truss structures with acceptable objective values.

In single-objective evolutionary optimization, the maintenance of structural diversity has played an important role since the beginning. Without any diversity maintenance, the population will quickly converge to a set containing only copies of a single solution, an effect called premature convergence [37]. In multi-objective optimization, the problem is less obvious, as the existence of conflicting objectives ensures that there is a set of Pareto-optimal solutions instead of a single optimal solution. Algorithms that aim at achieving a good distribution of the solutions across the Pareto-front usually also achieve a certain degree of structural diversity at the same time. Note that in multi-objective optimization, the term *diversity* is usually used in the context of objective space diversity, where solutions should (a) be as close

to the front as possible, and (b) well distributed (i.e. diverse) in objective space. In this thesis, however, the term *diversity* refers to the structural diversity, i.e. the diversity of solutions in decision space. Many notions of diversity exist, and it is important to formally define what is meant by the term *diversity*, as it is done in Section 2.2.

Maintaining multiple solutions that cover different parts of the decision space, e.g. different designs, offers many advantages: First, it enables the decision maker to choose among different designs with the same or at least equally preferable objective values. In certain applications, it may even be more important to find a diverse set of close-to-optimal solutions than to identify a set of optimal but structurally similar solutions. Second, it helps the decision maker to gather information about the problem structure; and third, it can speed up search—for instance by improving exploration and preventing premature convergence. Finally, when simultaneously optimizing diversity and objective function values, large gains in diversity can be achieved with little to no loss in objective space values. The next two sections elaborate on when and why diversity optimization might be helpful, and they give an overview over related work.

2.1.1 · Single-objective Problems

In a standard single-objective problem, the optimization goal is to find the one solution with the best objective function value. Nevertheless, there may be several reasons for an optimization scenario where not a single best solution is of interest but a set of diverse high-quality solutions. At first, the result of the optimization may be only a single step in a complex design process, as in the engineering examples above. Due to unknowns in the whole decision process, one would rather be interested in various possible options that explore the solution space and can be evaluated further (maybe based on additional criteria). Secondly, a set of diverse (almost) optimal solutions as the result of an optimization may be used to learn more about the system to be optimized. Finally, optimizations are usually based on a suitable abstraction of the problem, for example in form of an analytic model or a simulation. These models typically contain simplifications and

need appropriate parameterizations. This modeling process introduces uncertainties in the objective function. Other reasons for such uncertainties are unknown or time-varying system parameters. An optimization process which yields a single solution may not be sufficient in this case as it reflects only a single possible problem instance. Rather, one would be interested in a diverse set of solutions that provide appropriate decision support.

There exists a large body of methods that integrate diversity preservation into evolutionary search methods, see for example [32, 118] for an overview. Most of these methods try to maintain diverse solutions in order to fight the problem of premature convergence during the optimization. However, known approaches do not directly optimize diversity as a set measure, but rather have some implicit diversity preservation, e.g. through the maintenance of different niches, see also [32]. In the following, a more detailed overview about comparable approaches is given, including methods that determine solutions that are robust towards uncertainties in the objective function or solutions that reflect sets of local minima.

As stated before, one of the main reasons why a diverse, close-to-optimal set of solutions is beneficial is that there are uncertainties in the design process and in the modeling of a system. The handling of uncertainties during optimization has been treated before, for an overview see e.g. [58]. Four different categories of uncertainties are distinguished: (1) Subsequent evaluations of the same individual yield different objective values. (2) There are uncertainties in the decision variables. Both categories are usually treated by repeatedly evaluating a single individual in order to get an estimation of its fitness. Further categories are: (3) Uncertainties introduced by the usage of a simplified model of a real-world problem. (4) Objective functions that change over time. Methods dealing with dynamically changing objective functions usually try to introduce or maintain a certain degree of diversity, which is discussed next.

There are many algorithms that attempt to preserve diversity during an optimization run. The motivation for these methods usually comes from optimizing multi-modal problems, where evolutionary algorithms can get stuck in local optima due to genetic drift, see e.g. [32, 118]. One method

is to run several populations in parallel with the goal that they explore different regions in the search space. Island model EAs and parallel EAs fall into this category as well. Usually, there is some exchange between the different populations in the runs, and the main difference between existing algorithms is on how often individuals are exchanged and which individuals are exchanged [32]. A similar concept is implemented in the forking GA [105], where subpopulations are created when needed to explore a new part of the decision space. The sequential niche technique [6] runs several EAs in sequence, and passes information from one run to the next in order to prevent the following runs to find the same local optimum.

Other approaches are based on speciation, an observation from nature which states that first, only individuals from the same species can mate to produce offspring and second, there is a certain amount of geographic separation between individuals from the same species, and only neighboring individuals are eligible for mating. Examples for corresponding algorithmic techniques are assigning individuals to species prior to any selection step and to restrict competition, see [80], or placing mating restrictions on the individuals by assigning a geographic location of each individual, see e.g. [32], or by only allowing individuals within a certain distance of each other to mate [25]. Other methods use fitness sharing, see [38], such that individuals that have a lot of close neighbors have a reduced fitness. Another approach is to use crowding, see [22], where individuals can only be replaced by neighboring individuals.

Finally, Ursem [112] switches between exploration phases and exploitation phases, depending on the current diversity. Shimodaira [96] uses the distance to the best solution as the primary selection criterion.

Many of these algorithms do not optimize diversity explicitly by means of a set measure. Maintaining diversity is used to increase the probability to find the global optimum, or at least different local optima.

2.1.2 · Multi-objective Problems

In multi-objective optimization there is no single best solution, but a set of tradeoff solutions. An optimized set should both be close to the true Pareto-front, and well distributed in objective space. In this chapter, we additionally would like the set to be well distributed, i.e. diverse in decision space.

Interestingly, the idea to integrate diversity into multi-objective optimization has been proposed as early as 1994 in the first NSGA paper [100]. NSGA uses fitness sharing on the decision vectors in combination with non-dominated sorting, i.e. it groups solutions that are mutually nondominating into dominance classes, and considers diversity only when a selection between non-dominated solutions has to be made. After that, most algorithms concentrated on properties of the objective space only, such as the front shape and the distribution of optimal solutions [3]. In recent years, however, a few studies have picked up on this idea and have proposed alternative approaches. In 2003, GDEA [104] integrated diversity into the search as an additional objective. In 2008, the Omni-Optimizer [28] was developed which extends the original idea of NSGA, but in contrast to NSGA, its diversity measure takes both the decision and objective space diversity into account. It does so by alternating between considering decision space and objective space diversity, depending on which one is larger at the current evaluation. In 2009, two further studies were proposed. [97] extended a CMA-ES niching framework to include diversity by using an unweighted sum of objective space goodness and diversity. The MMEA [119] on the other hand applies clustering in objective space and then builds a statistical model from the solutions in these clusters. This model is then used during variation in order to generate new offspring. Finally in 2010, SPAM [124] was proposed, which offers the possibility to use a sequence of quality and diversity indicators. Also, a proof of convergence to the Pareto-front has been provided.

Most of the time, the exact optimization goal is often far from clear, meaning that the optimal set of solutions is not well-defined, nor is it easily possible to specify the desired tradeoff between quality of the solutions in the objective

space and their diversity. Diversity is a set measure and should be defined accordingly. A single solution is never diverse on its own, it is diverse with respect to other solutions.

As diversity is a set measure, it is a separate goal to the optimization. The other goal – let's call it the objective space measure – is also a set measure that indicates how well the final population approximates the Pareto-optimal front. With two set measures the question arises how these two measures can be combined. NSGA [100], the Omni-Optimizer [28] and SPAM [124] use a ranking of the two, where the objective space measure is always considered first, and only if there are ties using this measure, diversity is taken into consideration. The drawback of this approach is that the diversity plays an inferior role and there is no possibility to change the tradeoff between the two measures. A second approach is considering the diversity as an additional objective, as for example in GDEA [104]. The problem is that the diversity, which is defined on sets, is treated the same way as the original solution-oriented objectives. A second problem is that all tradeoffs between diversity and original objectives are explored concurrently, without any means to adjust the tradeoff. As the number of incomparable solutions increases, this may lead to an ineffective search. Also, a solution might be accepted into the optimized set which has poor objective values, as long as it adds considerably to the diversity. In this chapter, on the other hand, we are assuming that a decision maker is only interested in high quality solutions. Finally, MMEA [119] and the approach of Shir et al. [97] both assume that the decision space is Euclidean, whereas in this chapter, we do not make any assumptions about the decision space.

2.1.3 · Overview of Proposed Methods

This chapter proposes three methods that tackle the disadvantages of previous approaches, as discussed in the last two sections. All three methods share the notion of a tradeoff between diversity and objective values. Usually, a random population is very diverse, and the diversity decreases when the population evolves towards the Pareto-front. In most state-of-the-art approaches, the quality of a solution in terms of its objective values

is more important than its contribution to population diversity, whereas in the methods proposed in this chapter, the tradeoff can be set by the user.

In the single-objective case presented in Section 2.3, the optimization goal is to determine a set of maximally diverse solutions, with the constraint that the solutions must have a certain quality with respect to the given objective function, i.e. the user can specify a certain *barrier* value, which determines whether a solution is acceptable in terms of its objective value or not. The algorithm iteratively switches between objective value and diversity optimization while automatically adapting a constraint on the objective value until it reaches the barrier. To be able to appropriately set that quality constraint, any standard single-objective optimizer can be used prior to the diversity optimization to calculate the best achievable objective value.

In the multi-objective case, we first present a method that resembles the single-objective approach, where the tradeoff between diversity and goodness in objective space can be explicitly set by the user by specifying the minimal proximity of an acceptable solution to the Pareto-front. As the true Pareto-optimal front is unknown for most problems, an approximation of the Pareto-front is evolved together with the diversity-optimized population. In the second method, the diversity measure is integrated into the hypervolume indicator, leading to a new indicator that can be used for selection, with an adjustable tradeoff between diversity and goodness in objective space.

2.2 · Measuring Diversity

Up to now, we have assumed that a diversity measure $\mathrm{D} : 2^{\mathcal{X}} \to \mathbb{R}$ is given. This section states the requirements a suitable diversity measure should fulfill, and gives a review of commonly used measures.

Typically, measures for the diversity of a set are based on the definition of a pairwise distance between any two elements. As shortly introduced in Section 1.1, we assume that we are given a distance measure $d : \mathcal{X}^2 \to \mathbb{R}^{\geq 0}$ on the decision space. In contrast to the objective space $\mathcal{Y} \subseteq \mathbb{R}^m$, depending on the optimization problem and the corresponding representation of solutions,

we are confronted with many different classes of decision spaces, such as vectors, graphs, trees or even programs. In order to be applicable for a large class of optimization domains, we would like to place as few restrictions on the structure of the decision space as possible, i.e. we do not require that $\mathcal{X}$ is a Euclidean space or that the triangle inequality is satisfied. Instead, we just assume $\mathcal{X}$ to be a semimetric space, i.e., $\forall a, b \in \mathcal{X}$: $d(a, b) \geq 0$ (*non-negativity*), $d(a, b) = d(b, a)$ (*symmetry*), $d(a, a) = 0$ (*identity of indiscernibles*). Given such a distance measure, we now would like to define a set diversity measure $\mathrm{D} : 2^{\mathcal{X}} \to \mathbb{R}^{\geq 0}$ which assigns to each subset of the decision space a real value, i.e. its diversity. Note that while we are looking mainly for diversity in decision space, a diversity measure as defined above could also be used to measure diversity in objective space, e.g. by setting the distance between two solutions $a, b \in \mathcal{X}$ to the Euclidean distance between the solutions in objective space, i.e. $d(a, b) = ||f(a) - f(b)||_2$.

There are many possible interpretations and concepts of set diversity, i.e. how a given number of solutions should be distributed in decision space such that they achieve an optimal set diversity. In order to get a first insight, let us consider a simple example. Figure 2.1 shows the optimized[1] distribution of 100 points in a two dimensional Euclidean space $\mathcal{X} = [0, 1]^2$ for two diversity measures, namely the commonly used measure of summing up all pairwise distances, as well as the Solow-Polasky [98] measure which is described in Section 2.2.3. While the Solow-Polasky measure gives a grid-like structure, the sum of pairwise distance measure distributes all 100 solutions into the four corners. As a result, it appears that we need to define a set of formal requirements for a useful diversity measure.

2.2.1 · Requirements

Measuring diversity of sets is much-discussed in biology, more specifically in the field of biodiversity. The vast amount of studies on this subject offers also potential for the application in evolutionary algorithms. Not

[1] A randomized hill climber that tries to optimize the measure by moving the solutions slightly in a random direction was used. The used distance measure was Euclidean distance. The optimizer was run for 1000 seconds for each measure.

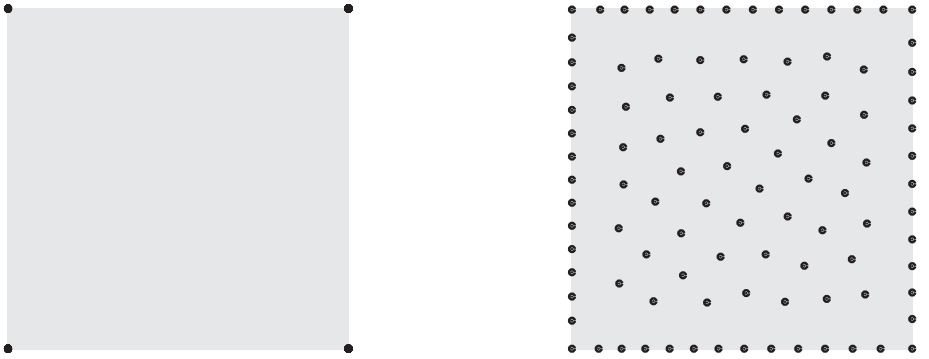


Figure 2.1 Best distributions of 100 points found by the hill-climber, for the sum of pairwise distances diversity measure (left) and the Solow-Polasky measure (right).

only because these algorithms are inspired by biology, but also because the idea of diversity in evolutionary algorithms is similar to the one in biology. However, there are also some substantial differences worth pointing out, in particular the solutions of an EA are arbitrarily duplicable in contrast to individuals in nature.

So what constitutes a diverse set? Just as for the decision maker's preference, no generally agreed-on definition exists neither in biology nor in the field of evolutionary algorithms. In the following, we discuss the most prominent classes of existing biodiversity measures with respect to their applicability to EAs. We thereby in particular consider the following three requirements to a diversity measure D, first proposed by Solow and Polasky [98]:

P1: **Monotonicity in Varieties** The diversity of a set of solutions $\mathcal{A}$ should increase when adding an individual b not yet in $\mathcal{A}$, i.e., if $\mathcal{A} \subset \mathcal{B}$, then $\mathrm{D}(\mathcal{A}) < \mathrm{D}(\mathcal{B})$). This fundamental property assures that increased species richness is reflected by the diversity measure [57].

P2: **Twinning** Diversity should stay constant when adding an individual c already in $\mathcal{A}$, i.e., $\mathrm{D}(\mathcal{A} \cup c) = \mathrm{D}(\mathcal{A})$ iff $c \in \mathcal{A}$. This property is subject to debate in biology, where having more of a rare species mostly is a

desirable thing. In evolutionary algorithms—were solutions are digital blueprints—having the same solution more than once is without benefits and should not increase D.

P3: Monotonicity in Distance The diversity of set $\mathcal{A}$ should not decrease if all pairs of solutions are at least as dissimilar (measured by d) as before. I.e. for a one-to-one mapping of $\mathcal{A}$ onto $\mathcal{B}$ such that $d(a_i, a_j) \leq d(b_i, b_j)$ with at least one strict inequality and $a_i, a_j \in \mathcal{A}$ and $b_i, b_j \in \mathcal{B}$, $\mathrm{D}(\mathcal{A}) \leq \mathrm{D}(\mathcal{B})$ holds. The more dissimilar solutions are, the better. Note that in finite decision spaces, optimally diverse sets of a measure fulfilling this requirement contain solutions on the border of the decision space. This might not be desired in certain situations, for example in a Euclidean space, a diversity measure whose optimal set corresponds to a centroidal Voronoi tessellation might be more desirable.

2.2.2 · Overview of Existing Measures

Based on Relative Abundances One straightforward way of measuring diversity is based on the relative abundance of each solution present in set $\mathcal{A}$. Methods include the indices of Simpson, Shannon, and Berger-Parker [36]. Those measures are easy to calculate, however, they have two major drawbacks: First, the degree of dissimilarity d between individuals has no influence, although it is important for example in real valued problems. Second, the twinning property is not fulfilled.

Based on Taxonomy The second group of diversity measures is based on taxonomy, where the individuals are first arranged in a dendrogram reflecting the taxonomic distinctiveness of solutions. Diversity of the set then corresponds to the overall path length in the taxonomic tree, see e.g. [87]. Most methods that build such trees are based on heuristics which meet neither monotonicity in distance nor monotonicity in varieties. One approach that at least fulfills monotonicity in variety has been proposed by Weitzman [115], but unfortunately building the taxonomic tree has a runtime which is exponential in the number of individuals, and is therefore only feasible for small sets of solutions (containing at most 20 individuals).

Based on Aggregating the Dissimilarities A simple way of aggregating the dissimilarity information give by d into a diversity measure is to sum up the values, $\mathrm{D}(\mathcal{A}) = \sum_{a,b\in\mathcal{A}} d(a,b)$ [57], or similarly, using the average distance, $\mathrm{D}(\mathcal{A}) = \frac{1}{|\mathcal{A}|^2}\sum_{a,b\in\mathcal{A}} d(a,b)$. Shir et al. for instance used this measure in their EA [97]. Other algorithms, such as the Omni-Optimizer, only consider the distance d to the closest neighbors of a solution. However, all these measures do not meet the twinning requirement. Even worse, these measures promote having only two solutions with large distance duplicated multiple times.

Based on Utility of Solutions A completely new approach has been presented by Solow and Polasky [98]. Their measure is based on an utilitarian view on individuals, where the function $u : \mathcal{X} \to \mathbb{R}^{\geq 0}$ defines the utility of any subset of solutions. Every single individual has a predefined utility of 1. Having this individual duplicated does not increase the overall utility. On the other hand, having two completely distinct individuals results in a utility of 2. In between these two extreme cases, the utility needs to increase monotonically. Solow and Polasky suggested to use an exponential increase $u = 2 - e^{(-d(a,b))}$. The major difficulty is to calculate the utility of three or more individuals (similar to the problem of calculating the overlap of more than two solutions provided only the distance (hence overlap) of pairs of solutions). Solow and Polasky have proven that an approximation can be used to get a lower bound on the utility of arbitrary sets. Fortunately, despite using this approximation the three above requirements are fulfilled in most cases.

Coverage Measures Diversity can be intuitively understood as measuring the coverage of a set of points in Euclidean space. The more regularly the points are distributed, the better the space is covered. Each chosen point covers a certain area around it, and the size of the union of the covered space is the diversity measure, see e.g. [109]. The notion of coverage is a special case of the notion of utility as explained above. While this measure satisfies the three requirements, it also assumes that the points can be embedded in Euclidean space, which is generally not possible for a set of points with arbitrary pairwise distances [9].

Diversity in Evolutionary Algorithms In the evolutionary algorithm literature, a lot of thought has been given to decision space diversity. Most of it is in the context of diversity preservation, especially in single-objective optimization, in order to prevent premature convergence. The approaches use different notions of diversity. Some make use of nominal spaces, i.e. discrete spaces where the different values are either equal or different, but no measure for telling how different exists. These approaches then use the Hamming distance to their crossover mate [105] or to the best solution found so far [96] to assess the diversity contribution to the population. Squillero and Tonda [99] use an entropy measure to assess the diversity of the whole set. Others assume an Euclidean space and use the average distance to the centroid of the population as a diversity measure [111, 112]. Others again use the distance to the nearest neighbors [28] or the number of neighbors within a certain distance [100] to measure the diversity of a point. Zhou et al. [119] assume that the Pareto-optimal solutions are known and compute the coverage of these solutions. Li et al [66] do a hierarchical clustering of the solutions until the cluster centroids are sufficiently far apart. The number of clusters then is the diversity of the set. The Omni-Optimizer [28] assumes real-valued decision variables and uses the crowding distance, i.e. the distance to the two nearest neighbors in each dimension. Finally, there are some problem-specific diversity measures [28, 90], when for example it is known that the Pareto-optimal solutions are divided into clusters, the number of found clusters can be used as a diversity measure.

Most of these measures either require a specific decision space, like a nominally scaled or an Euclidean one, or they do not define a measure on sets, but only calculate the relative diversity of each solution with respect to the remaining solutions, or they make assumptions on the Pareto-front or the problem landscape.

Comparison of Measures Table 2.1 summarizes the different diversity measures in context of the three requirements P1, P2 and P3. As can be seen, only the coverage measure and the measure by Solow-Polasky satisfy all three requirements, but not for sets with arbitrary pairwise distances. The coverage measure assumes that the points are given in Euclidean space.

class	method	P1 (Varieties)	P2 (Twinning)	P3 (Distance)
relative abundance	Simpson [36]	no	no	yes
	Shannon [36]	no	no	yes
	Berger-Parker [36]	no	no	yes
taxonomy	clustering [87]	no	yes	no
	Weitzman [115]	yes	yes	no
functions of distance	sum [57]	yes	no	yes
	crowding distance [28]	no	no	yes
coverage	union of point coverages* [111]	yes	yes	yes
utilitarian	Solow-Polasky [98]	yes*	yes*	yes*

* Making assumptions about the decision space that exceed those presented in Section 2.2.

Table 2.1 Comparison of different diversity metrics with respect to the three properties stated in Sec. 2.2.1, monotonicity in varieties (P1), twinning (P2), and monotonicity in distance (P3).

Moreover, calculating it is #P-hard [11], which makes it unusable for larger decision spaces. The measure of Solow and Polasky is the main measure of diversity used throughout this thesis. It works in most cases, but not in all, a fact which is investigated in the next section.

2.2.3 · The Measure of Solow and Polasky

The Solow-Polasky measure $\mathrm{D}(\mathcal{P})$ of a population $\mathcal{P} \subseteq \mathcal{X}$ is determined as follows: Suppose $\mathcal{P}$ contains the n solutions $p_1, ..., p_n$ where $|\mathcal{P}| = n$. Furthermore, $d(p_i, p_j)$ denotes the distance between solutions p_i and p_j. Then we can define the (n, n)-matrix $M = (m_{ij})$ with elements

$$m_{ij} = \exp(-\theta \cdot d(p_i, p_j)) \quad \text{for all } 1 \leq i, j \leq n$$

Then, the Solow-Polasky measure can be given as

$$\mathrm{D}(\mathcal{P}) = eM^{-1}e^T$$

where $e = (1, 1, \ldots, 1)$ and e^T denotes its transpose. In other words, $\mathrm{D}(\mathcal{P})$ is the sum of all matrix elements of M^{-1}.

The Solow-Polasky measure yields real values in the interval $[1, |\mathcal{P}|]$, which can be interpreted as the number of different species found in the population, where individuals that lie close to each other belong to the same species. The parameter θ normalizes the relationship between distance d and the number of species. As the selection of a distance d is problem domain specific, the value of θ has to be appropriately set. Following our experimental evaluations, the choice of θ is not critical as long as the matrix elements of M are in a reasonable interval, i.e. $10^{-5} \leq m_{ij} \ll 1$, $\forall i, j$, $i \neq j$.

Fulfilling the Requirements

In the paper where the Solow-Polasky measure is proposed, the authors give basic proofs that their measure fulfills the three requirements twinning, monotonicity in varieties and monotonicity in distance. However, these proofs make certain assumptions, therefore the Solow-Polasky measure does not satisfy the requirements for solution sets with arbitrary pairwise distances.

In the proofs for both twinning and monotonicity in varieties, they use following formula, taken from [35]:

$$\sup_c \frac{(c^T \mathbb{1}^n c)}{c^T M c} = e M^{-1} e^T$$

where c is any vector of length n and $\mathbb{1}^n$ is a matrix of size $n \times n$ only containing ones. However, according to [35], this formula can only be used if M is positive definite. In the case of the Solow-Polasky measure, the transformation $\exp(-\theta \cdot d(a, b))$ is positive definite, assuring that M is positive semidefinite, but not necessarily positive definite.

Also, Solow and Polasky state that *"[the Solow-Polasky measure] is not monotone in distance. [...] This seemingly paradoxical result arises [...] if M becomes singular in a certain way"*. They also state that in the case of 3 solutions, the fulfillment of the triangle inequality ensures that the measure is monotone in distance. To the best of our knowledge, this result has not been extended to more than 3 points, so in general, the Solow-Polasky

measure is not known to be monotone in distance for an arbitrary number of points.

Despite the fact that the Solow-Polasky measure does not fulfill the requirements in all the cases, it works remarkably well in most cases, especially if the points are given in Euclidean space. It also leads to the nicest set distributions when compared to the other approaches mentioned above. Therefore, this measure will be used throughout the thesis. Nevertheless, care has to be taken when running algorithms using this measure, because if the pairwise distance matrix M becomes singular, the global optimum will (falsely) be located at this point, which leads to non-diverse sets that have a high diversity value. One example of such a singular matrix is shown in Appendix C.

Proposed Diversity-based Selection

During environmental selection, we would like to select a subset of size μ from the given $\mu + \lambda$ parents and offspring. Because the diversity is used as a selection criterion, a way to find the best subset of a population has to be found. The problem of selecting the best subset $\mathcal{P}'$ of size n from $\mathcal{P}$ can be formulated as follows:

$$\underset{\mathcal{P}' \in \mathcal{P}, |\mathcal{P}'|=n}{\operatorname{argmax}} \mathrm{D}(\mathcal{P}')$$

As testing all possible subsets is infeasible due to combinatorial explosion, we suggest to use the usual greedy strategy which removes one solution after another from the population $\mathcal{P}$ until only n solutions remain. In each step, the solution that contributes least to the diversity is discarded. Here, the contribution of a solution $p \in \mathcal{P}$ to the diversity of the set $\mathcal{P}$ is defined as $\mathrm{D}(\mathcal{P}) - \mathrm{D}(\mathcal{P} \backslash \{p\})$, i.e. the difference between the diversity of the whole set and the diversity of the set without the solution p.

The computational complexity of the calculation of a subset optimized for the Solow-Polasky measure is now determined by the fact that we have to remove n solutions and for each of them, we have to test between $n+1$ and $2n$

candidates. Each candidate evaluation for p necessitates the computation of $\mathrm{D}(\mathcal{P}\backslash\{p\})$ whose complexity is dominated by the matrix inverse calculation of the Solow-Polasky measure, which is $\mathcal{O}(n^3)$.

As a result, the computational complexity of the selection problem is reduced to $\mathcal{O}(n^5)$ in comparison to an exponential complexity, while giving up on the optimality of the obtained subset. Unfortunately, the computational complexity still is unacceptable for practical purposes, i.e. large population sizes. The next subsection describes an improved algorithm that reduces the complexity to $\mathcal{O}(n^3)$.

Fast Diversity-based Selection Algorithm

As described above, the complexity of $\mathcal{O}(n^5)$ to determine an optimized subset with maximal diversity is still a serious performance bottleneck. In the following we therefore suggest a novel way to (a) calculate the contributions of solutions to the Solow-Polasky measure and (b) to update the measure after removing a solution that only requires one matrix inversion in the whole selection process, therefore reducing its complexity to $\mathcal{O}(n^3)$.

First, we provide some definitions and known relations from linear algebra that will be used. Assume that we have a symmetric matrix M and its inverse M^{-1} which are partitioned in the following form:

$$M = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix}, \quad M^{-1} = \begin{pmatrix} \bar{A} & \bar{b} \\ \bar{b}^T & \bar{c} \end{pmatrix}$$

where c and $\bar{c}$ are single elements, b and $\bar{b}$ are column vectors and b^T and $\bar{b}^T$ denote their transpose. We also make use of the notion $\Sigma(M) = \sum_{i,j} m_{i,j}$ which is the sum of all elements of the matrix M. Finally, we use the well known result for the block matrix inverse of M:

$$A^{-1} = \bar{A} - \frac{1}{\bar{c}} \cdot \bar{b} \cdot \bar{b}^T$$

We now want to calculate the contribution of a single solution to the Solow-Polasky measure. Remember that the Solow-Polasky measure is the element-wise sum of the inverse M^{-1} of the transformed pairwise distance

matrix M of all solutions, i.e. $\mathrm{D}(\mathcal{P}) = \Sigma(M^{-1})$. Note that M can be described in the partitioned form as M is symmetric due to the symmetry of the distance measure, i.e. $d(p_i, p_j) = d(p_j, p_i)$ for all $p_i, p_j \in \mathcal{X}$.

If a solution is discarded from $\mathcal{P}$, its corresponding row and column are deleted from the distance matrix M. Assume without loss of generality that the solution we want to discard corresponds to the last row and column of M, i.e. we want to delete the last row and the last column from M and determine the impact on the Solow-Polasky measure. This difference in the measure can now be calculated as follows:

$$\begin{aligned}\Sigma(M^{-1}) - \Sigma(A^{-1}) &= [\Sigma(\bar{A}) + 2\Sigma(\bar{b}) + \bar{c}] \\ &\quad - [\Sigma(\bar{A}) - \frac{1}{\bar{c}}(\Sigma(\bar{b}))^2] \\ &= \frac{1}{\bar{c}}[2\bar{c}\Sigma(\bar{b}) + (\bar{c})^2 + (\Sigma(\bar{b}))^2] \\ &= \frac{1}{\bar{c}}(\Sigma(\bar{b}) + \bar{c})^2\end{aligned}$$

The term $\frac{1}{\bar{c}}(\Sigma(\bar{b}) + \bar{c})^2$ can be interpreted as the normalized squared sum of the last column's elements of M^{-1}. By comparing all of these terms we can determine the solution which leads to the least difference in the diversity measure by $\mathcal{O}(n^2)$ operations.

Afterwards, we have to delete from M the solution with the smallest contribution and set the new distance matrix M' to the corresponding submatrix. If we again suppose without loss of generality that the solution with the smallest loss in diversity was associated to the last column, we have $M' = A$. In order to repeat this process for further solutions we would have to determine the inverse $M'^{-1} = A^{-1}$ which would need $\mathcal{O}(n^3)$ computations in a naive implementation. But using the above results on block matrix inverses, we can reduce this computation to $\mathcal{O}(n^2)$ computations. As a result, the removal of one element needs $\mathcal{O}(n^2)$ computations which leads to the desired $\mathcal{O}(n^3)$ complexity for the whole subset computation the selection problem.

2.3 · Maximizing Population Diversity in Single-objective Optimization

This section presents the Diversity-optimizing Single-objective Evolutionary Algorithm (NOAH). NOAH switches between optimizing for fitness, and optimizing for diversity. Quality of fitness is ensured by a constraint on the fitness value, which is tightened during the run. Finally, NOAH returns a population which is diversity optimized, but contains only solutions whose fitness is better than a predefined barrier value.

2.3.1 · Problem Setting

The problem definition tackled in this section can be interpreted as a special kind of multi-objective optimization, denoted as *mixed multi-objective problem*, where the first goal is to generate solutions that optimize some objective function, and the second goal is to have a final set of solutions which is as diverse as possible with respect to some diversity measure. In contrast to typical multi-objective problems, where a vector of objective functions is associated to each individual solution, *mixed multi-objective problems* have a different structure: One objective can be described by a function that maps individual solutions to objective values whereas the other objective is defined by a set indicator that maps sets of solutions to objective values.

We use the general setup presented in Section 1.1. More precisely, we are considering the minimization of a single objective function $f : \mathcal{X} \to \mathbb{R}$. Here, $\mathcal{X}$ denotes the feasible set of solutions in the decision space, i.e. the set of alternatives of the decision problem. A single alternative $x \in \mathcal{X}$ is denoted as a solution x. The image of $\mathcal{X}$ under f is denoted as the feasible set in the objective space $\mathcal{Y} = f(\mathcal{X}) = \{y \in \mathbb{R} \mid \exists x \in \mathcal{X} : y = f(x)\}$. Therefore, the objective value of a single solution x is $f(x)$.

There are no assumptions about the structure of the decision space, except that a symmetric distance measure $d : \mathcal{X}^2 \to \mathbb{R}$ between two solutions is required. Based on the distance measure we define a diversity measure $\mathrm{D} : 2^{\mathcal{X}} \to \mathbb{R}$. It is defined on the powerset of the decision space, i.e. all possible subsets of the decision space, and determines for a subset of the

decision space its corresponding diversity. Finally, a provided barrier value v is used to determine a constraint on the objective values.

The *mixed multi-objective optimization problem* we are trying to solve can therefore be stated as follows:

Problem 2.1 (mixed multi-objective problem): Determine a population $\mathcal{P} \subseteq \mathcal{X}$ with a given size $|\mathcal{P}| = n$ which maximizes the diversity measure D while satisfying the provided barrier v on the objective values:

$$\max_{\mathcal{P} \subseteq \mathcal{X}_v, |\mathcal{P}|=n} \mathrm{D}(\mathcal{P}) \text{ where } \mathcal{X}_v = \{x \in \mathcal{X} \mid f(x) \leq v\}$$

In other words, we are trying to find a population $\mathcal{P}$ that only contains solutions which are better or equal than the barrier v and which maximizes the diversity measure D. Note that this is *not* the same as multi-modal optimization, where multiple local optima are sought without considering their quality, see e.g [91]. Neither are we looking for robust solutions or solutions insensitive to change as for example in dynamic environments [14]. Also, we do not consider diversity as an additional independent objective, as we are not interested in diverse but low-quality solutions. Instead, we want diverse solutions that satisfy a certain quality bound.

Let us now present a simple example. Consider a minimization problem with a one-dimensional real-valued decision space. The objective function is depicted in Figure 2.2. We would like to find a maximally distributed set of solutions below a given barrier value (horizontal line). Figure 2.2 shows the case that the decision maker finds all solutions that have an objective value of 0.45 or lower to be acceptable. One possible set of solutions that satisfy the quality constraint and that are well distributed in decision space is shown as circles in the Figure[2].

[2] These solutions have been generated using NOAH as defined in Algorithm 1 for 450 function evaluations, with the following parameters: $n = 10$, $v = 0.45$, $g = 10$, $r = 5$, $c = 20$ (see Section 2.3.2 for more details), using Euclidean distance as a distance measure and using the Solow-Polasky measure as defined in Section 2.2.3 for calculating the diversity D.

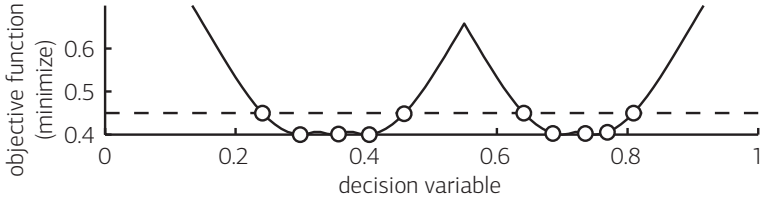


Figure 2.2 Simple objective function with decision space on x-axis and objective space on y-axis. Circles show solutions found by NOAH with barrier $v = 0.45$ (dashed horizontal line).

Note that a similar problem setting has been considered before by Schütze et al. [93]. They developed an algorithm to solve a space mission design, where the trajectory of a satellite going from earth to a comet has to be optimized, see [114]. The objective function in this case is the total variation in velocity throughout the whole trajectory. One crucial design parameter is the date of the satellite launch. Here, it is important to have decision space diversity, in order to present the engineer near-optimal trajectories with different launch dates. Their approach presents a new archiving strategy that has been integrated into a differential evolution algorithm. The algorithm takes two parameters Δ and ε and returns a set $\mathcal{P}^*$, where no point $p \in \mathcal{P}^*$ is more than ε away from the value of the global optimum, and no $p \in \mathcal{P}^*$ has any other solution $s \in \mathcal{P}^*$ in its neighborhood of size Δ. This algorithm has been tailored for problems with real-valued decision spaces, i.e. $\mathcal{X} \subseteq \mathbb{R}^d$, where d is the size of the decision space, although it would probably be possible to extend it to general decision spaces and general distance measures. Also, the size of the final set $\mathcal{P}^*$ strongly depends on the choice of Δ, which is problem-dependent and has to be chosen carefully.

2.3.2 · NOAH Algorithm

In this section, we propose a new algorithm called NOAH to solve the mixed multi-objective problem. Remember from Section 2.3.1 that we assume that there is a certain objective value v, called the barrier, below which all solutions are acceptable. This barrier value can be flexibly chosen. The algorithm we propose in this section then generates a population which only

```
1: function NOAH(n, v, g, r, c)
2:     Initialize population P randomly with n solutions
3:     b = ∞
4:     while (b > v)∧(termination criterion not reached) do
5:         P := OBJOPT(P, g, b)
6:         (P, b) := BOUNDCHANGE(P, r)
7:         P := DIVOPT(P, n, b, c)
8:     return P
```

Algorithm 1 Mixed multi-objective optimization algorithm NOAH. Input parameters: population size n; barrier value v; minimization of objective function is done for g generations; r solutions remain in the population after bound adaptation; the population diversity converged if it did not improve for a total of c generations.

contains solutions that are better or equal than this barrier and that are as diverse as possible, as stated in Problem 2.1. In case the barrier is set to a value lower than any value the algorithm is able to achieve, NOAH performs a conventional single-objective optimization where solutions with a better objective function value are always more desirable than those with a worse value.

NOAH uses two key concepts to solve Problem 2.1: bound adaptation and diversity optimization. Its main structure is shown in Algorithm 1. Each iteration consists of three steps, namely the optimization of the objective function f by means of OBJOPT, the bound adaptation using BOUNDCHANGE and the diversity optimization in DIVOPT. The iteration stops if all solutions p in the population $\mathcal{P}$ have objective value $f(p) \leq v$ or some other termination criterion is satisfied.

The rationale behind NOAH will now be described in some more detail. As mentioned above, in each loop a standard evolutionary algorithm operates for g generations, then the bound is adapted and finally diversity is optimized until it converges. *In other words, objective value and population diversity are jointly optimized by transforming the mixed multi-objective problem into a constrained set diversity optimization problem.* The constraint is the bound b on the objective values which is adaptively reduced until it reaches the provided barrier value v. The diversity optimization

DIVOPT results in a population which is optimized with respect to its diversity $\mathrm{D}(\mathcal{P})$ but respects the constraint imposed by the bound b.

Subalgorithms OBJOPT and BOUNDCHANGE are responsible for optimizing the population with respect to the objective function f. OBJOPT receives a population $\mathcal{P}$ with n elements and objective values $f(p) \leq b$ and uses a standard evolutionary algorithm for g generations to optimize it. Any optimization algorithm can be used as long as the solutions in the resulting population also have objective values $f(p) \leq b$.

In order to balance diversity optimization and objective value optimization, a bound value b is monotonically decreased during the run in BOUNDCHANGE. The new bound value is set in such a way that at least r individuals in the population are still on or below the new bound. These individuals form the new population.

Finally, DIVOPT maximizes the diversity $\mathrm{D}(\mathcal{P})$ under the constraint that the resulting population has again n elements whose objective values are at or below b, i.e. $f(p) \leq b$. The iterative optimization in DIVOPT terminates if the diversity did not improve for a total of c generations. As a result we can state that in each iteration OBJOPT optimizes the population for g generations with respect to the objective function f, then BOUNDCHANGE adaptively adjusts the objective value bound b such that r solutions are on or below the new bound b, and DIVOPT maximizes the diversity while maintaining the bound b. Now, some more details about the different subalgorithms of NOAH are provided.

The objective value optimization OBJOPT uses a simple $(\mu + \lambda)$ evolutionary algorithm with $\mu = \lambda = n$ which respects the bound b, see Algorithm 2. The variation function VARIATEPOP may use any appropriate combination of mutation and crossover operators in order to generate a resulting population with n solutions. Its only difference to a standard variation of a given population is that it returns only solutions that have an objective value not worse than b. One way to achieve this is to call the operators as many times as necessary to generate enough feasible individuals. Selection function SELECTOBJ selects a population of n solutions according to some (possibly standard) selection criterion that ensures selection pressure. Note that any

1: **function** OBJOPT($\mathcal{P}$, g, b)
2: $n := |\mathcal{P}|$
3: **for** g iterations **do**
4: $\mathcal{P}' :=$ VARIATEPOP($\mathcal{P}$, b, n)
5: $\mathcal{P} :=$ SELECTOBJ($\{\mathcal{P} \cup \mathcal{P}'\}$, n)
6: **return** $\mathcal{P}$

Algorithm 2 Objective value optimization OBJOPT. Input parameters: population $\mathcal{P}$; number of generations g; bound b.

1: **function** BOUNDCHANGE($\mathcal{P}$, r)
2: $b :=$ minimal x s.t. $|\{p|p \in \mathcal{P}, f(p) \leq x\}| \geq r$
3: $\mathcal{P}' := \{p \in \mathcal{P}|f(p) \leq b\}$
4: **return** $(\mathcal{P}', b)$

Algorithm 3 Adaptive change of bound BOUNDCHANGE. Input parameters: population $\mathcal{P}$; minimal number of solutions in resulting population r.

other refined strategy can be used for OBJOPT as long as the bound b is respected in the resulting population.

The strategy to adaptively change the bound value b is described in Algorithm 3. In BOUNDCHANGE, the new bound is set to the minimal value such that at least r solutions are still on or below it. The resulting subset of the population contains all elements with objective values equal or below this new bound.

The optimization of diversity DIVOPT is described in Algorithm 4. At first, the already described variation operator VARIATEPOP is called that generates a population $\mathcal{P}'$ by any appropriate combination of mutation and crossover operators. Again, it returns only solutions that have an objective value not worse than b. The number of generated solutions is chosen such that $\{\mathcal{P} \cup \mathcal{P}'\}$ has $2n$ solutions (remember that we have chosen $\mu = \lambda = n$). In the selection phase the solutions are selected according to their diversity contribution using the operator SELECTDIV that we choose to be the fast diversity selection scheme using the Solow Polasky diversity measure as described in Section 2.2.3. This is in contrast to the standard evolutionary algorithm shown

```
1: function divOpt(P, n, b, c)
2:     i := 0
3:     while i < c do
4:         P' := variatePop(P, b, 2n − |P|)
5:         P'' := selectDiv({P ∪ P'}, n)
6:         if D(P'') > D(P) then
7:             P := P''
8:         else
9:             i := i + 1
10:    return P
```

Algorithm 4 Diversity optimization divOpt. Input parameters: population $\mathcal{P}$; population size n; bound value b; the total number of generations the diversity did not change for convergence c.

in Algorithm 2, where solutions are selected according to their objective values. Moreover, the diversity optimization is run until there have been c generations in total without an increase in diversity. Note that as soon as the adaptive bound b has reached the user-specified barrier value v, diversity is optimized one more time until it converges and NOAH is stopped.

2.3.3 · Results

In this section we compare NOAH to several other standard evolutionary algorithms with and without diversity preservation mechanisms. The purpose of this experimental evaluation is to see whether the considered set of algorithms is able to reach a given barrier, and if so, what conclusion can be drawn about the diversity of the final populations.

3-Sat and nk-Landscape Problems

For this comparison, we selected two well-known test problems: The nk-Landscapes problem [60] and the 3-Sat problem [71]. In the nk-Landscapes problem, there are n decision variables (in our case, $n = 100$). Each decision variable is influenced by k (in our case $k = 10$) randomly chosen other decision variables. The decision variables are binary, i.e. they can either take the value 0 or 1. Each decision variable together with the influencing

Name	Diversity Preserving	Mating Selection	Environmental Selection
NOAH	yes	Random without replacement	see Section 2.3.2
DetC	yes	Random without replacement	Deterministic crowding [69]
ResT	yes	Random with replacement	Restricted tournament [48]
Diff	yes	Random without replacement	Diffusion model [32]
Clear	yes	Random without replacement	Clearing procedure [80]
Share	yes	Fitness sharing [32, 38]	Pairwise tournament
Tour	no	Random without replacement	Pairwise tournament
Random	no	n/a	n/a

Table 2.2 Compared algorithms.

decision variables codes an index in a randomly generated fitness matrix. The overall fitness then is the sum of the fitness values coded by each decision variable.

The 3-Sat problem is a specific Boolean satisfiability problem. In our case, the Boolean expression that has to be satisfied consists of 200 clauses with 3 elements each. A clause is true if any of its elements is set to one, and the whole expression is true if all clauses are true. As an objective function, we use the number of false clauses, leading to a minimization problem that has an optimal value of 0 (that can only be reached if the expression is satisfiable). Our problem has 50 decision variables, where each clause contains 3 randomly selected decision variables as its elements.

Both optimization problems have binary search spaces. We suggest to use the Hamming distance between decision vectors as a distance measure. For example considering the 3-Sat problem, we want not only to be able to find out whether the expression is satisfiable, but also to find a whole set of assignments that satisfy the Boolean expression. These assignments should be as diverse as possible in terms of differing decision variables.

As a variation operator, we first apply a two-point crossover with probability 0.5. Then, each solution undergoes a one-point bitflip mutation, i.e. one of its (binary) decision variables is selected at random and set to its inverse value (1 instead of 0 and vice versa).

Compared Algorithms All algorithms that we compare are listed in Table 2.2. Mating selection denotes the step where the parents that will be recombined and mutated are selected. During environmental selection the individuals which are to survive (from the pool of parents and offspring) are chosen. NOAH optimizes according to Algorithm 1, with parameters $n = 20, g = 20, r = 10, c = 10$, and with pairwise tournament for SELECTOBJ in Algorithm 2.

During deterministic crowding (DetC), offspring are generated by recombining and mutating two parents, and then, a pairwise tournament between each offspring and its more similar parent takes place, see [32]. In restricted tournament (ResT), offspring are generated in a standard manner, and then each offspring replaces the most similar parent, if it is better than said parent. In the Diffusion Model Evolutionary Algorithm (Diff), the solutions are located on a grid in a fixed manner, where each solution has 8 neighbors. During variation, each individual is recombined with one of its neighbors. The offspring that is more similar to the neighbor replaces the current individual if it is better. The Clearing Procedure (Clear) generates offspring in a standard way. Then, it performs a pruning on the offspring in order to find the κ best individuals in each niche. Niches are defined by a parameter σ, which in our case is set to 0.2 for all problems. Also, we use $\kappa = 1$, i.e. we use only one representative per niche. This representative then replaces the most similar parent, if it is better than that parent. When using fitness sharing (Share), the fitness of each individual is decreased prior to selection, depending on the closeness and number of neighbors. We set the niche radius σ to 0.2. Random selection with replacement is just a random selection of parents, where each individual can be selected multiple times. In the same selection without replacement, each individual can only be selected once. In pairwise tournament (Tour), pairs of solutions are selected and the better one is kept. Finally, the random algorithm (Random) generates random solutions and keeps the 20 best ones (if more than 20 individuals have the same best value, the most diverse ones are kept).

Experimental Setup We test each problem with different barrier values. We compare the number of runs that achieved at least one solution with the

barrier value, and the diversity of the solutions that reached the barrier value. In order to be able to fairly compare the different algorithms, all objective values below the barrier are set to the barrier, such that there is no selection pressure below the barrier. This way, the algorithms are free to optimize the diversity of the population after the barrier has been reached.

For each problem, the number of objective function evaluations *fEvals* is fixed. Note that NOAH terminates as soon as its bound reaches the barrier value, or when *fEvals* function evaluations have been performed, whichever happens first. For the nk-Landscapes and the 3-Sat problem, *fEvals* was set to $5 \cdot 10^5$ and 10^6, respectively. All algorithms use a population size of 20. For the diversity calculation we use $\theta = 10$.

Each problem/barrier value pair was run on 30 different instances of the 3-Sat and nk-Landscape problems. To test the resulting diversity values for significant differences, a Kruskal-Wallis test as described in [19] has been carried out, using the Conover-Inman procedure, Fisher's least significant difference method performed on ranks and a significance level of 1%.

Results The results are shown in Table 2.3. For the two higher barrier values of 3-Sat, the algorithms (except Random) always reach the barrier value, and the resulting diversity of NOAH is always significantly better than that of the other algorithms, even random search. For the lowest barrier value, NOAH sometimes does not reach the barrier, which can be explained with the fact that it spends a considerable amount of function evaluations on diversity optimizations. But when NOAH does reach the barrier of $v = 2$, the diversity of the solutions reaching the barrier is always significantly higher than the diversity achieved by the remaining algorithms.

For the nk-Landscapes problem it is interesting to note that for the lowest barrier value, most algorithms cannot reach that barrier (except Tour and Share, which reach the barrier once/twice). This is in contrast to NOAH, which reaches the barrier every third time. This indicates that diversity might help identifying the global optimum by covering as many local optima as possible. This can still be seen for the second lowest barrier value, which is always reached by NOAH, whereas it is only reached in about 50% of the

	NOAH		DetC		ResT		Diff	
3-Sat $v = 2$	22	13.4883	30	7.0855^{+}	30	7.3768^{+}	29	6.6820^{+}
3-Sat $v = 5$	30	16.5848	30	8.0943^{+}	30	12.2711^{+}	30	11.7993^{+}
3-Sat $v = 10$	30	17.4981	30	11.8069^{+}	30	14.2957^{+}	30	13.7548^{+}
nk-L. $v = 23$	13	2.0249	0	NaN	0	NaN	0	NaN
nk-L. $v = 25$	30	5.2924	5	1.0000^{+}	8	1.2478^{+}	3	1.0000^{+}
nk-L. $v = 30$	30	17.3183	30	14.2214^{+}	30	13.9020^{+}	30	13.6611^{+}
	Clear		Share		Tour		Random	
3-Sat $v = 2$	29	7.7224^{+}	30	1.3600^{+}	30	1.5759^{+}	0	NaN
3-Sat $v = 5$	30	12.4424^{+}	30	1.7953^{+}	30	2.5646^{+}	7	1.6355^{+}
3-Sat $v = 10$	30	14.3036^{+}	30	1.8493^{+}	30	3.4553^{+}	30	17.2617^{+}
nk-L. $v = 23$	0	NaN	2	1.0000	1	1.0500	0	NaN
nk-L. $v = 25$	6	1.0000^{+}	12	1.0042^{+}	13	1.0077^{+}	0	NaN
nk-L. $v = 30$	30	14.3188^{+}	30	1.1999^{+}	30	1.2114^{+}	0	NaN

Table 2.3 Experiment results of 30 runs. Columns show the different algorithms, rows the different problems (with the corresponding barrier value v). For each problem/barrier value pair and each algorithm there are two values, where the left one is the number of runs that had at least one solution on the barrier, and the right number is the mean diversity of the solutions that reached the barrier. A $+/-$ beside the diversity means that the diversity of NOAH is significantly better/worse than the diversity of that particular algorithm.

cases by the best other algorithms (Tour and Share). For the highest barrier value, all algorithms always reach it (except random search), but NOAH's diversity is always significantly better than the other algorithm's diversity.

Bridge Construction

In order to qualitatively interpret the simultaneous optimization of the objective function and the set diversity, we applied NOAH to a more realistic problem. Here, we would like to see whether truss bridges constructed and optimized by NOAH 'look' more diverse than bridges produced by standard evolutionary algorithms. We are using the bridge optimization problem described in Appendix B. The goal is to find a bridge with minimum weight, as a higher weight implies higher material cost, which in turn implies higher overall cost. The recombination/mutation probabilities were set to 0.5 and 1, respectively.

First, we want to find the best possible bridge. To do so we run all algorithms with a barrier value of $v = -10$, i.e. a value that can never be

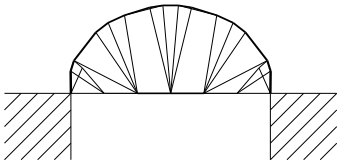


Figure 2.3 Best bridge over all runs with $v = -10$, achieved by NOAH.

achieved as the weight of the bridge cannot be negative. NOAH is run with parameters $n = 20, g = 20, r = 10, c = 10$. The results of this run can be seen in Figures 2.4. When looking at the fitness values, NOAH achieves the best fitness values of the solutions in the final populations, ResT the second best, DetC third, Share and Tour fourth, Diff fifth, Random sixth and Clear worst. Interestingly enough, NOAH finds the bridge with the lowest weight of 538.584kg, as depicted in Figure 2.3. Looking at the achieved diversity, it can be seen that Clear achieves the highest diversity, ResT and Random the second highest, DetC and Diff the second lowest and NOAH, Share and Tour the lowest. It is expected that the better the fitness of the solutions in the final population, the lower the diversity should be. Therefore, there is a tradeoff between fitness and diversity, and no algorithm is best in both of them. However, NOAH is better in both diversity and fitness than Share and Tour, and ResT is better than DetC, Share, Tour, Diff and Random.

Second, we want to see whether the diversity of the solutions can be increased if the constraint on the objective value is relaxed. To do so, we set the barrier to two values, 580kg which is 7.69% heavier than the best bridge shown in Figure 2.3, and 700kg which is 29.97% heavier than the best bridge. As NOAH is expected to reach higher barriers earlier, it can spend the gained time on diversity optimization. Therefore, we used NOAH with parameters $n = 20, g = 20, r = 10, c = 50$ for the barrier of 580kg, and

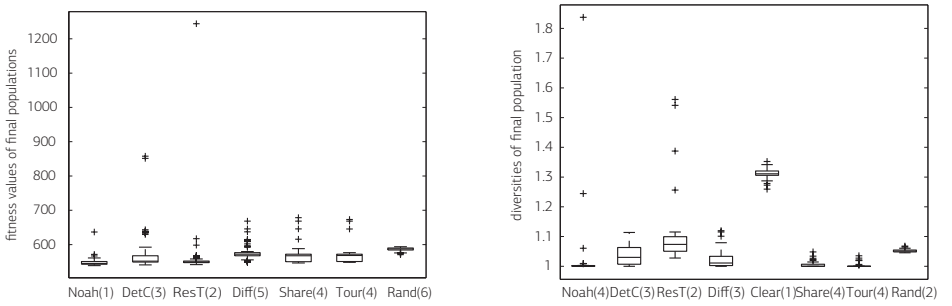


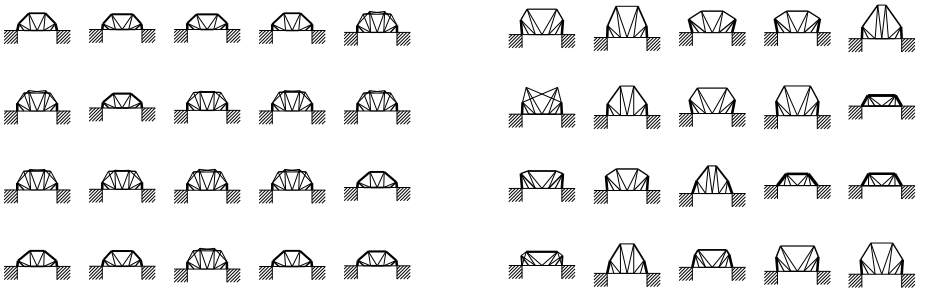
Figure 2.4 Diversity and fitnesses of final populations achieved on the bridge optimization problem with $v = -10$. Clear had an average fitness of 4368 ± 18378, with a worst fitness of $303\,680$ and is therefore not shown in the left plot. Each algorithm name is annotated in brackets with the rank according to a significance test. A lower number means that the algorithm was significantly better than those with higher numbers.

	NOAH		DetC		ResT		Diff	
bridge $v = 580$	26	1.1515	30	1.0682^+	30	1.0866^+	29	1.0388^+
bridge $v = 700$	29	1.6102	30	1.1873^+	30	1.1631^+	30	1.1006^+
	Clear		Share		Tour		Random	
bridge $v = 580$	30	1.0405^+	28	1.0010^+	28	1.0299^+	25	1.0097^+
bridge $v = 700$	30	1.0909^+	30	1.0025^+	30	1.1010^+	30	1.1625^+

Table 2.4 Experiment results of 30 runs. Columns show the different algorithms, rows the different problems (with the corresponding barrier value v). For each problem/barrier value pair and each algorithm there are two values, where the left one is the number of runs that had at least one solution on the barrier, and the right number is the mean diversity of the solutions that reached the barrier. A $+/-$ beside the diversity means that the diversity of NOAH is significantly better/worse than the diversity of that particular algorithm.

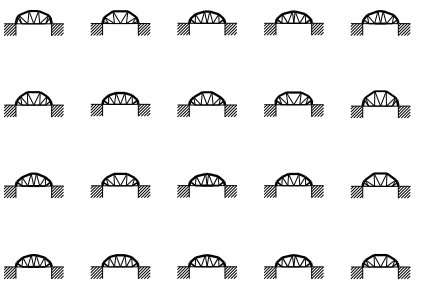
NOAH with parameters $n = 20, g = 20, r = 10, c = 100$ for the barrier of 700kg.

The results can be seen in Table 2.4. NOAH is the only algorithm (besides random search) which does not reach the barrier in all of the runs. However, when it does reach the barrier, it's diversity is always significantly higher than the diversities achieved by the remaining algorithms. Two example populations (with a diversity close to the mean diversity over all NOAH runs



(a) $v = 580$, diversity 1.1563, NOAH

(b) $v = 700$, diversity 1.5959, NOAH



(c) $v = 580$, diversity 1.0849, ResT

(d) $v = 700$, diversity 1.1869, DetC

Figure 2.5 Example populations of the NOAH runs (upper row), as well as of the second best performing algorithms (lower row), for the barriers $v = 580$ (left) and $v = 700$ (right).

for that particular barrier) are shown in Figure 2.5. As can be seen the bridges with the higher barrier value look more dissimilar than the bridges with the lower barrier. Figure 2.5 also shows example populations of the second best performing algorithms, as determined by a statistical test on the achieved diversity values (DetC for $v = 700$ and ResT for $v = 580$). Again, populations were chosen whose diversity is closest to the mean diversity of all 30 runs of that algorithm. As can be seen, the bridges found by NOAH look more dissimilar in shape than the bridges found by the second best algorithm, especially for the higher barrier.

2.3.4 · NOAH Summary

This section proposes a method to generate a set of maximally diverse solutions which are better in terms of objective value than a certain fitness value. All solutions beyond this barrier are supposed to be acceptable to the decision maker.

To this end, we propose an algorithm called NOAH that alternates between optimizing the population for diversity and for objective value, and that uses an adaptive constraint to ensure the quality of the solutions.

NOAH is compared to standard evolutionary algorithms with and without diversity preservation on the nk-Landscapes and the 3-Sat problem. It could be seen that NOAH consistently achieves a significantly better diversity than the other algorithms. On the nk-Landscapes problem it appears that the diversity preservation helps identifying better local optima, as NOAH achieves better fitness values than the other algorithms.

All algorithms are also applied to a truss bridge construction problem. NOAH was able to find significantly more diverse bridges than standard evolutionary algorithms, both if about 7% and about 30% more weight is allowed than the weight of the best bridge found with by the algorithms with an unreachable barrier value.

An important feature of NOAH is its ability to adaptively reduce its current bound value during optimization. In the future, it would be desirable to automatically tune the parameters of NOAH, especially the number of generations for which the optimization of fitness values takes place, as this parameter decides on the tradeoff between diversity and fitness optimization speed.

2.4 · Maximizing Population Diversity in Multi-objective Optimization

This section introduces the Diversity-optimizing Multi-objective Evolutionary Algorithm (DIOP). We present a possibility to improve the diversity of a solution set, while satisfying a user-defined constraint in terms of the

minimum proximity of these solutions to the Pareto-front, i.e. the minimum quality of the solutions regarding their objective values. More specifically, we make the following assumptions about the preferences of a decision maker throughout this section:

1. The decision maker is interested in a set of solutions.
2. Each solution in this set should be close to optimal, i.e., not "far" from the Pareto-front in objective space.
3. The target population should cover large parts of the decision space, i.e. offer decision space diversity.

2.4.1 · Problem Setting

Again, we use the general setup presented in Section 1.1. We additionally introduce the notion of of weak ε-Pareto-dominance defined as $a \preceq_\varepsilon b$ iff $f_i(a) - \varepsilon \leq f_i(b)$, $\forall\ i \in \{1, ..., m\}$. This definition follows the one of weak additive ε-dominance as defined in [123]. In other words, suppose that we improve solution a in every objective by ε. Then $a \preceq_\varepsilon b$ iff the improved solution weakly dominates solution b.

Also, let $\mathcal{X}^* \subseteq \mathcal{X}$ denote the Pareto-optimal set, $\mathcal{X}^* = \{x \mid \nexists a \in \mathcal{X} : a \preceq x \wedge x \not\preceq a\}$, let $\mathcal{T} \subset \mathcal{X}$ denote a target population of solutions, and let $q_{\mathcal{X}^*} : \mathcal{X} \to \mathbb{R}^{\geq 0}$ measure for each solution $x \in \mathcal{X}$ the distance $q_{\mathcal{X}^*}(x)$ to the Pareto-optimal set $\mathcal{X}^*$. Let $\mathrm{D}(\mathcal{T}) : 2^{\mathcal{X}} \to \mathbb{R}^{\geq 0}$ measure the diversity of a set of solutions $\mathcal{T} \subseteq \mathcal{X}$ in the decision space. Given this notation, the three optimization assumptions provided above can be formalized as follows:

Problem 2.2: Formalization of assumptions about the preferences of a decision maker:

1. We are interested in a target population of solutions $\mathcal{T} \subseteq \mathcal{X}$, $|\mathcal{T}| = \mu$, where μ denotes its size.
2. Optimality: $\forall t \in \mathcal{T} : q_{\mathcal{X}^*}(t) \leq \varepsilon$, where ε is a given bound on the optimality of solutions in $\mathcal{T}$.
3. Diversity in decision space: Determine $\mathcal{T}$ such that $\mathrm{D}(\mathcal{T})$ is maximal.

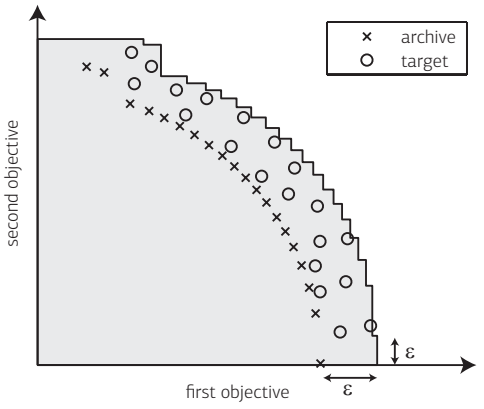


Figure 2.6 Example of a typical archive $\mathcal{A}$ and target $\mathcal{T}$ population for a minimization problem. The feasible region (grey area) is the area containing all solutions that weakly ε-dominate at least one solution from the archive, i.e. $\{(x_1, x_2) \in \mathbb{R}^2 | \exists a \in \mathcal{A} : x_i \leq \varepsilon + f(a_i), \forall i \in \{1, 2\}\}$.

As a consequence, we are dealing with a *constrained optimization problem on sets of solutions*. Given this setting, different problems arise:

- A way to measure the distance $q_{\mathcal{X}^*}(x)$ of a solution x to the Pareto-front $\mathcal{X}^*$ has to be specified.
- Determining $q_{\mathcal{X}^*}(\mathcal{T})$ requires knowledge of the Pareto-optimal set $\mathcal{X}^*$, which in general is not known.
- The constraint on the objective space quality of the solutions might be difficult to fulfill, especially if the specified distance to the front is small. A set fulfilling the constraints can be generated with a standard optimization run. However, it is not clear how a set fulfilling the constraint can be generated while still maintaining the maximally possible decision space diversity. Diversity once lost might be difficult to reintroduce.

2.4.2 · DIOP Algorithm

DIOP provides one way to tackle above problems. As the *Pareto-optimal set $\mathcal{X}^*$ in general is unknown*, we propose using a helper set, called the *archive* $\mathcal{A}$, that approximates $\mathcal{X}^*$. We therefore have two concurrent MOEAs, one

which optimizes the target population according to diversity under the quality constraint, which depends on the archive, and one which optimizes the archive population according to objective values, see Figure 2.6 for an example. This offers the advantage that the quality constraint (decision maker preference 2) continuously tightens as the archive population improves. In order to benefit from one another, the two sets can exchange solutions, therefore improving the diversity in the archive and producing more solutions that satisfy the quality constraint in the target. This is useful as experiments have indicated that considering diverse solutions might speed up the search for some problems [111].

Having an approximation $\mathcal{A}$ of the Pareto-optimal set $\mathcal{X}^*$, a *distance metric* $q_{\mathcal{A}}$ has to be defined. We propose to use $\preceq_\varepsilon$ to define the distance as the smallest ε to reach ε-dominance of any solution in $\mathcal{A}$, i.e.,

$$q_{\mathcal{A}}(x) := \min\{\varepsilon \mid \exists a \in \mathcal{A} : x \preceq_\varepsilon a\}$$

As the decision maker is only interested in solutions not exceeding a predefined distance ε to the Pareto-front, the diversity measure of an arbitrary target population $\mathcal{T}$ is only calculated for those solutions $\mathcal{T}^\varepsilon \subseteq \mathcal{T}$ not exceeding the distance ε from the front approximation $\mathcal{A}$, $\mathcal{T}^\varepsilon = \{t \in \mathcal{T} \mid q_{\mathcal{A}}(t) \leq \varepsilon\}$. All solutions satisfying this condition for a given archive are shown as the grey area in Figure 2.6. The goal of the target population therefore is to optimize the constrained diversity measure

$$\mathrm{D}^{\mathcal{A}}(\mathcal{T}, \varepsilon) = \mathrm{D}(t : t \in \mathcal{T} \wedge q_{\mathcal{A}}(t) \leq \varepsilon)$$

Note that this is an adapted version of the DIOP algorithm proposed in [110]. In the original paper, we used a weighted sum of a diversity measure in objective space and a diversity measure in decision space. The problem with this approach is that a weighted sum only makes sense if the values to be weighted are normalized. However, as the achievable diversity values in objective space may be quite different from the achievable diversity values

1: **function** DIOP(ε, μ^a, μ^t)
2: $\mathcal{A} = \{x_1, ..., x_{\mu^a}\}$, $x_i \in \mathcal{X}$ (*randomly initialize archive)*
3: $\mathcal{T} = \{x_1, ..., x_{\mu^t}\}$, $x_i \in \mathcal{X}$ (*randomly initialize target)*
4: **while** stopping criterion not met **do**
5: $\mathcal{A}' = \text{VARIATE}(\mathcal{A}, \mu^a)$ (*generate archive offspring)*
6: $\mathcal{A}'' = \text{ARCHIVESELECT}(\mathcal{A} \cup \mathcal{A}' \cup \mathcal{T}, \mu^a)$ (*select μ^a new individuals)*
7: (*Only use new archive if its hypervolume value is better or equal)*
8: **if** $I_H(\mathcal{A}'') \geq I_H(\mathcal{A})$ **then**
9: $\mathcal{A} = \mathcal{A}''$
10: $\mathcal{T}' = \text{VARIATE}(\mathcal{T}, \mu^t)$ (*generate target offspring)*
11: $\mathcal{T}'' = \text{TARGETSELECT}(\mathcal{A}, \mathcal{T} \cup \mathcal{T}' \cup \mathcal{A}, \mu^t, \varepsilon)$ (*select μ^t new individuals)*
12: $\mathcal{T} = \mathcal{T}''$
13: Return $\mathcal{T}$

Algorithm 5 DIOP algorithm. Takes a parameter ε, an archive size μ^a, a target size μ^t. Returns the optimized target set.

in decision space, a normalization is not straightforward. In the original paper, this was done by trial and error, which is time consuming. In this thesis a DIOP version that only optimizes the diversity in decision space is proposed for simplicity. Note that the original version with weighted sums can be reintroduced easily if necessary.

To sum up, DIOP simultaneously evolves two populations, namely the archive $\mathcal{A}$ that approximates $\mathcal{X}^*$ according to the hypervolume indicator, and the target population $\mathcal{T}$ that maximizes diversity under a quality constraint. Each population produces its own offspring. During environmental selection, both populations can select from their offspring, the current archive $\mathcal{A}$ and the current target $\mathcal{T}$. The pseudocode of the proposed algorithm is shown in Algorithm 5.

The function $\mathcal{A}'' = \text{ARCHIVESELECT}(\mathcal{A}, \mu^a)$, selects μ^a solutions $\mathcal{A}''$ from a set $\mathcal{A}$. The selection goal is to maximize the hypervolume, i.e. $I_H(\mathcal{A}'')$. The function $\mathcal{P}' = \text{VARIATE}(\mathcal{P}, \mu)$ generates μ offspring $\mathcal{P}'$ from a given set $\mathcal{P}$. The method $\mathcal{T}' = \text{TARGETSELECT}(\mathcal{A}, \mathcal{T}, \mu^t, \varepsilon)$ selects μ^t solutions $\mathcal{T}'$ from set $\mathcal{T}$. The goal is to maximize $\text{D}^{\mathcal{A}}(\mathcal{T}', \varepsilon)$.

2.4.3 · Results

In this section, two main questions are investigated: first, how does the main parameter of DIOP, i.e. ε, influence the obtained target population in terms of the hypervolume and the achieved diversity? Second, we compare DIOP to the Standard Multi-objective Evolutionary Algorithm (sMOEA) (as described in appendix A) and to the Omni-Optimizer (OMNI) [28] on the nine test problems of the Walking Fish Group (WFG) testsuite [55] to assess its performance.

Experimental Setup

The WFG testproblems have a real-valued decision space $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d$. Therefore, we use a standard variation scheme for real-valued decision vectors. The method VARIATE$(\mathcal{P}, n)$ selects $n/2$ random pairs of solutions without replacement from $\mathcal{P}$ to generate the offspring population. These pairs are then recombined using the SBX crossover operator [24] with $\eta_c = 15$, where each pair is recombined with probability one. During a recombination, each decision variable is recombined separately with probability one. With probability 0.5, the recombined values of this decision variable are exchanged between the offspring. After recombination, each individual is mutated with probability one. To mutate an individual, each decision variable is mutated with probability $1/d$ using polynomial mutation [28] with $\eta_m = 20$. To perform the archive selection ARCHIVESELECT$(\mathcal{A} \cup \mathcal{A}' \cup \mathcal{T}, \mu^a)$, the same greedy hypervolume selection strategy as for the sMOEA is used as described in Algorithm 21. The Solow-Polasky measure is used to measure the decision space diversity $\mathrm{D}(\mathcal{P})$. To perform the target selection TARGETSELECT$(\mathcal{A}, \mathcal{T} \cup \mathcal{T}' \cup \mathcal{A}, \mu^t)$, the fast diversity selection scheme described in Section 2.2.3 is used, where only the feasible solutions $\mathcal{T}^f = \{t \in \mathcal{T} \cup \mathcal{T}' \cup \mathcal{A} | q_{\mathcal{A}}(t) \leq \varepsilon\}$ are considered for selection. Note that $\forall a \in \mathcal{A}$, $q_{\mathcal{A}}(a) = 0$ holds, and therefore, the feasible set is always large enough, i.e. $|\mathcal{T}^f| \geq \mu^t$, as long as $|\mathcal{A}| \geq \mu^t$. Here, we set $\mu^t = \mu^a$.

All differences in results mentioned for the remainder of the chapter are statistically significant according to the Kruskal-Wallis test with post-hoc Conover-Inman procedure [19] and with a significance level of 1%.

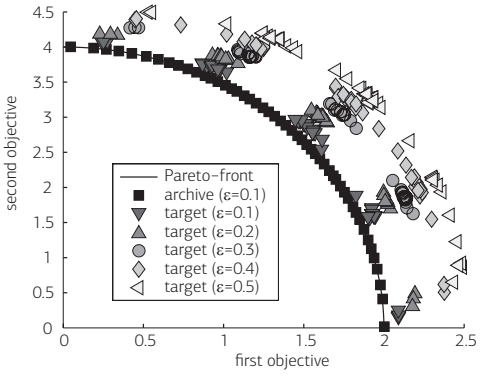


Figure 2.7 One run of DIOP for different values of ε on the WFG7 testproblem with 4 position and 20 distance related parameters.

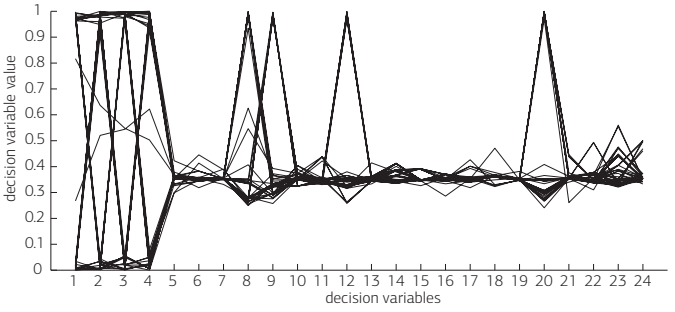


Figure 2.8 Parallel coordinates plot of the $\varepsilon = 0.1$ target population of Figure 2.7.

Influence of ε

To assess the influence of the parameter ε, DIOP is run on WFG7 with 2 objectives and 4 position and 20 distance dependent variables, i.e. a total of 24 decision variables. WFG7 was chosen as it is one of the easier WFG problems, being separable, unimodal, and having a concave Pareto-optimal front. Note though that DIOP can also be run on real-world problems with more complex decision spaces that are not metric. We chose the archive and target size to be 50 and run the algorithm for 50 000 function evaluations. For the Solow-Polasky measure, $\theta = 10$ was chosen. The parameter ε takes the values $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each setting of ε, 30 runs were done.

The results of one run are shown in Figures 2.7 and 2.8. A few things can be noticed. First, the diversity optimized solutions tend to stick to the border of the feasible region. This is due to the setup of the WFG-testsuite, where the decision variables are distributed into position and distance related parameters. For a solution to be Pareto-optimal, the distance related parameters must be equal to 0.35, whereas the position related parameters determine where on the front the solution lies. The further away from the front the solutions are, the more diverse the values of the distance related parameters become.

Second, there are five clusters of solutions in each population. The reason again is the distribution of variables into position- and distance-related variables. Pareto-optimal solutions have their 20 distance related parameters set to 0.35. Because DIOP allows to tradeoff optimality in objective space with diversity in decision space, the target population has these variables set as far from 0.35 as possible, while still satisfying the constraint in objective space. The 4 position related parameters, on the other hand, decide on the position of the solution on the front, and therefore can be set arbitrarily. For the highest diversity, they are either set to 0 or 1, resulting in $2^4 = 16$ distinct clusters in decision space. Using WFG7, these 16 clusters in decision space translate to 5 clusters in objective space, because in WFG7, the position on the front is defined by one parameter, which is calculated as the average of all 4 position dependent variables. As these variables are either set to 0 or 1, there are 5 different values for the average, i.e. $\{0, 0.25, 0.5, 0.75, 1\}$, which directly leads to the five clusters. This is an example where diversity in decision space does not automatically lead to diversity in objective space.

Figures 2.9 and 2.10 show boxplots of the achieved diversity and hypervolume over all 30 runs. As can be seen, the diversity and hypervolume of the archive populations are in the same range[3], whereas for the targets, the diversity increases with increasing ε, whereas the hypervolume decreases. This is as expected, as the further away the solutions are allowed to be, the higher the achievable diversity and the lower the achieved hypervolume.

[3] There are statistically significant differences, where the archive with $\varepsilon = 0.4$ has a better diversity than the archive with $\varepsilon = 0.1$, and the archives with $\varepsilon = 0.1$ and $\varepsilon = 0.5$ have a better hypervolume than the other archives.

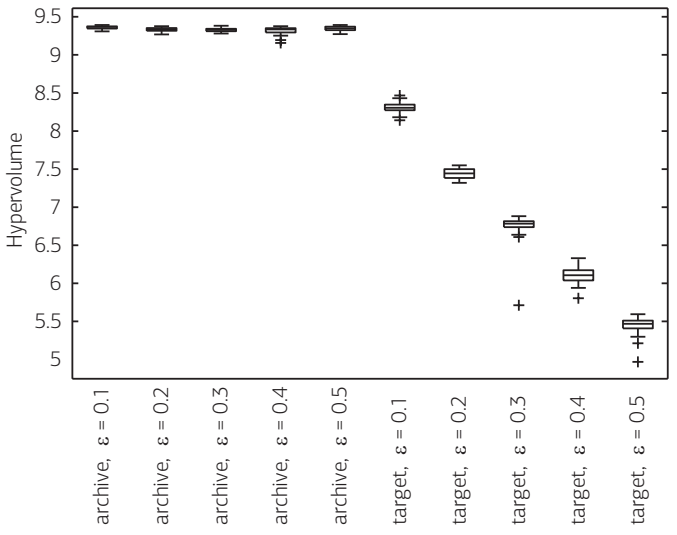


Figure 2.9 Hypervolume values for different settings of ε, for the archive and target populations of DIOP.

Therefore, through ε, the user can set the desired tradeoff between goodness in objective space and diversity. If there are solutions that are slightly suboptimal, but very different from the Pareto-optimal solutions, those can be found using DIOP. At the same time, if only Pareto-optimal solutions are desired, ε can be set to 0.

Comparison to sMOEA and to OMNI

In this section, we compare DIOP to the sMOEA as described in Section A and to OMNI [28]. While the sMOEA has as the sole goal the optimization of the hypervolume, the Omni-Optimizer uses the nondominated sorting procedure known from the Nondominated Sorting Genetic Algorithm (NSGA-II) [29], and in addition optimizes both the crowding distances in decision and in objective space. Note that while the sMOEA and DIOP can handle arbitrary decision spaces, the Omni-Optimizer is designed to work with binary or real valued decision spaces only.

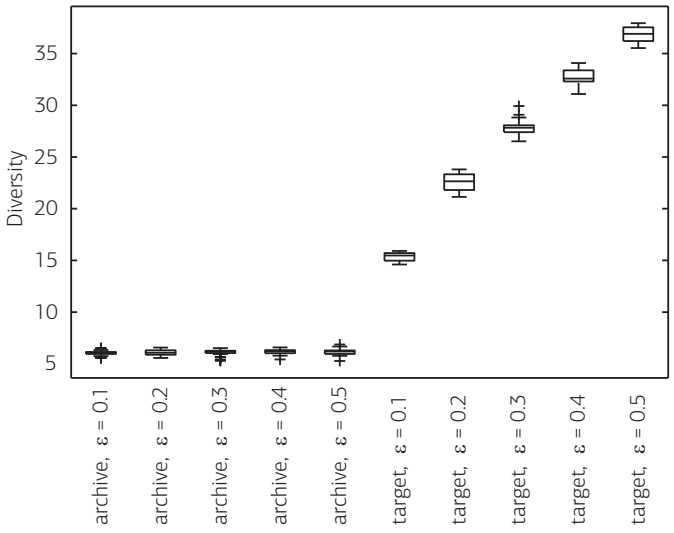


Figure 2.10 Diversity values for different settings of ε, for the archive and target populations of DIOP.

To compare these three algorithms, they were run on the nine testproblems of the WFG testsuite [55] with 3 objectives and 4 position and 20 distance dependent variables, i.e. a total of 24 decision variables. All of these problems have a known Pareto-optimal front. We chose the archive and target size, as well as the population sizes of the sMOEA and the Omni-Optimizer to be 100 and run the algorithms for 50 000 function evaluations. For the Solow-Polasky measure, $\theta = 10$ was chosen. For the hypervolume, a reference point of $r_i = 2 \cdot i + 1.1$ was chosen for the i-th objective. The parameter ε of DIOP was chosen to be 0.1. For each algorithm, 30 runs were done.

The resulting populations of one run for WFG9 are shown in Figure 2.11. As can be seen, the populations of the DIOP archive and the sMOEA look similar and cover the front well. Most solutions of the DIOP target reached the front, but the population is not well distributed in objective space, even though it has the highest diversity in decision space. Finally, the population of the Omni-Optimizer does not reach the front and is not as nicely distributed as the populations of the DIOP archive and the sMOEA. Also, boxplots for

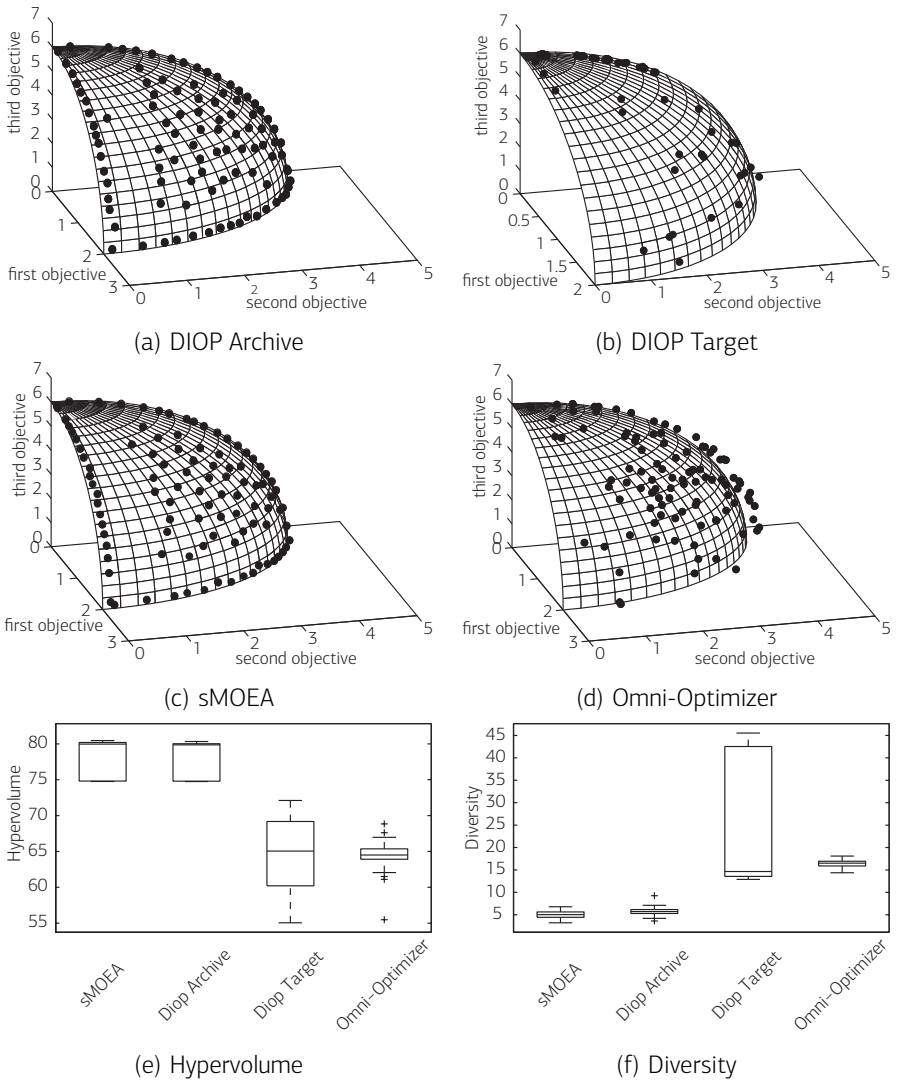


Figure 2.11 Example populations of one single run of the WFG9 problem, Figures (a)-(d). Hypervolume and Diversity of all runs of the WFG9 problem, (e) and (f).

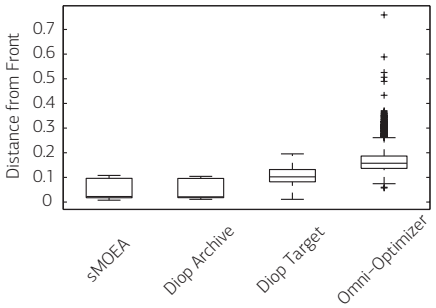


Figure 2.12 Distance to Pareto-front of the WFG9 problem for the different algorithms.

the achieved hypervolume and diversity of all 3 algorithms on WFG9 are shown in the bottom row of Figure 2.11. As can be seen, the hypervolume of both the sMOEA and the DIOP archive is significantly higher than the hypervolume of the DIOP archive and the Omni-Optimizer. There are no statistically significant differences between the sMOEA and the archive, and neither between the target and the Omni-Optimizer. At the same time, the target and Omni-Optimizer diversity is significantly higher than the sMOEA and archive diversity, and the archive diversity is also significantly higher than the sMOEA diversity.

To sum up, the sMOEA and the archive seem to be comparable in terms of hypervolume and diversity. At the same time, the target has no statistically significant differences from the Omni-Optimizer. When looking at the fronts, however, some differences are apparent. The target population is closer to the front, but less nicely distributed in objective space. This can be visualized when plotting the distances of the solutions of all 30 runs to the true Pareto-front (in terms of D as used in the general definition of the objective function of the WFG problems), as shown in Figure 2.12. Here it can be seen that the archive solutions are closest to the front, second comes the sMOEA, third the target and last the Omni-Optimizer.

The achieved hypervolumes/diversities on all testproblems are shown in Tables 2.5 and 2.6, repectively. The significance tests on the hypervolume and diversity values can be found in Tables 2.7 and 2.8. As can be seen, the

	sMOEA	Archive	Target	OMNI
WFG1	50.4 ± 2.6	44.4 ± 0.8	30.2 ± 8.9	24.5 ± 0.8
WFG2	99.2 ± 8.3	105.6 ± 5.7	100.3 ± 5.2	96.6 ± 3.1
WFG3	82.3 ± 0.1	81.9 ± 0.2	31.9 ± 4.4	62.7 ± 1.5
WFG4	83.2 ± 0.2	82.8 ± 0.3	73.6 ± 3.2	49.1 ± 1.6
WFG5	80.0 ± 0.3	79.5 ± 0.2	32.6 ± 12.4	49.3 ± 1.6
WFG6	80.7 ± 0.4	79.7 ± 0.4	61.4 ± 10.1	50.2 ± 2.1
WFG7	84.2 ± 0.0	84.1 ± 0.1	76.3 ± 1.1	52.6 ± 1.7
WFG8	77.7 ± 0.2	76.3 ± 0.2	68.5 ± 3.0	47.1 ± 1.5
WFG9	78.0 ± 2.7	78.6 ± 2.3	64.3 ± 5.1	64.4 ± 2.4

Table 2.5 Mean and standard deviation of hypervolume values of all 3 algorithms on all 9 testproblems.

	sMOEA	Archive	Target	OMNI
WFG1	3.2 ± 0.4	2.9 ± 0.5	29.4 ± 10.3	14.4 ± 1.2
WFG2	5.9 ± 0.3	6.1 ± 0.3	12.4 ± 1.2	7.0 ± 0.4
WFG3	11.2 ± 0.9	14.7 ± 1.6	78.5 ± 0.8	23.5 ± 1.0
WFG4	5.3 ± 0.7	5.3 ± 0.8	12.2 ± 1.1	22.4 ± 1.0
WFG5	5.1 ± 1.1	5.0 ± 0.9	80.7 ± 6.5	44.4 ± 1.2
WFG6	6.3 ± 1.0	7.1 ± 1.5	41.3 ± 9.3	20.3 ± 3.6
WFG7	7.3 ± 0.2	7.3 ± 0.1	14.0 ± 0.8	18.9 ± 0.9
WFG8	7.9 ± 0.2	8.0 ± 0.3	15.8 ± 2.8	18.9 ± 0.8
WFG9	5.0 ± 0.9	5.8 ± 1.0	22.8 ± 13.2	16.4 ± 0.7

Table 2.6 Mean and standard deviation of diversity values of all 3 algorithms on all 9 testproblems.

(diversity-optimizing) target and Omni-optimizer populations are always significantly more diverse than the sMOEA and archive populations. Also, with the exception of WFG2 where there is no significant difference between the sMOEA and the target population, the sMOEA and the archive populations always have a significantly higher hypervolume than the target and the Omni-optimizer populations. This shows the tradeoff between decision space diversity and hypervolume.

Furthermore, it can be seen that the archive has most of the time (with the notable exception of WFG2) a lower hypervolume and a lower diversity than the sMOEA. This can be attributed to the fact that the archive is a $(100 + 200)$-MOEA that was run for 250 generations, whereas the sMOEA is a $(100 + 100)$-MOEA that was run for 500 generations, where the latter seems to produce better results.

Finally for the target and the Omni-optimizer, it can be stated that the target is better than the Omni-Optimizer in the problems WFG2 and WFG6, and never worse in both the hypervolume and the diversity. In WFG1, the target populations have a higher diversity, without having a worse hypervolume. In WFG9, there is no statistically significant difference in either the hypervolume and the diversity. In WFG3 and WFG5, the target population has a higher diversity but a lower hypervolume than the Omni-Optimizer, whereas for WFG4, WFG7 and WFG8, the opposite holds.

To interpret these results, the relation between hypervolume and diversity has to be kept in mind. There are two conceptually different cases in which a lower hypervolume leads to a higher diversity. The first one happens when diverse solutions in decision space are not diverse in objective space, in which case structurally diverse solutions all are on the front, but not in a nicely distributed way. This is what happens to the target population in all WFG problems where the archive solutions are on the front, and therefore, the target solutions have to be within a distance of $\varepsilon = 0.1$ to the front as well. In these cases, it can be seen that structurally diverse solutions do not automatically lead to nicely distributed solutions in objective space.

In the second case, the solutions do not reach the front, and as especially in the WFG testsuite where the Pareto-optimal solutions have the same value in 20 out of the 24 decision variables, solutions further away from the front automatically have a higher diversity. This happens to the Omni-Optimizer most of the time, where in all problems except WFG3 the distance of all solutions produced by the Omni-Optimizer are significantly larger than the distances of the solutions of the sMOEA, the archive and the target (in terms of D as used in the general definition of the objective function of the WFG

	sMOEA	Archive	Target	OMNI
sMOEA	000000000	+-+++++++0	+0+++++++	+++++++++
Archive	-+++++++0	000000000	+++++++++	+++++++++
Target	-0-------	---------	000000000	0+-+-+++0
OMNI	---------	---------	0-+-+---0	000000000

Table 2.7 Hypervolume significance tests. Each entry has nine symbols, corresponding to the nine WFG testproblems. A +/−/0 as the k-th symbol in the i-th row and j-th column means that the j-th algorithm was significantly better / significantly worse / not significantly different from the i-th algorithm in WFGk.

	sMOEA	Archive	Target	OMNI
sMOEA	000000000	+--00--0-	---------	---------
Archive	-++00++0+	000000000	---------	---------
Target	+++++++++	+++++++++	000000000	+++-++--0
OMNI	+++++++++	+++++++++	---+--++0	000000000

Table 2.8 Diversity significance tests. Notation as in Table 2.7.

problems). WFG3 is the exception, because it is a degenerate problem, meaning that the Pareto-front is a line. Here, the archive has solutions that are non-dominated but do not lie on this line, and therefore, the target takes advantage of that and has many solutions in the vicinity of these non-dominated solutions.

2.4.4 · DIOP Summary

This section presented DIOP, a multi-objective evolutionary algorithm that evolves two populations simultaneously, one being optimized according to the hypervolume indicator, and the other being optimized according to diversity, while being constrained by a maximally allowed distance to the hypervolume-optimized population. This constraint can be set arbitrarily, denoting the quality the user is willing to loose in order to gain a higher diversity. Comparing DIOP to the Omni-Optimizer showed that DIOP is able to produce populations with a high diversity, which still lie close to the

Pareto-optimal front, whereas the Omni-Optimizer does not converge as well towards the front.

2.5 · Integrating Diversity into the Hypervolume Indicator

In the last section, DIOP was proposed which allows the user to specify the tradeoff between diversity and objective values through a constraint on the objective values. This approach suffers from the standard problems of setting constraints. Setting a constraint might give a wrong view on what is achievable. Consider the case that the user sets a constraint on the goodness in objective space. The corresponding diversity of the final population is low. If the constraint had been relaxed slightly, the diversity could have increased dramatically, but unless the user tries different settings for the constraint, which is time-intensive, the user will never known that.

Furthermore, there might be some problems where solutions that are diverse in decision space might lie close in objective space, as e.g. several WFG problems as shown in the last section. In such a case, it might be useful to have a method that automatically finds a good tradeoff between diversity in decision and in objective space. For these reasons, this section proposes the Diversity-integrating Multi-objective Evolutionary Algorithm (DIVA), an algorithm that automatically combines the two goals, objective functions and structural diversity in a flexible manner. To do so, a modified hypervolume indicator that includes the diversity contribution of each solution is proposed, and this indicator is then used during environmental selection.

2.5.1 · Problem Setting

Again, we use the general setup presented in Section 1.1. More precisely, we assume that two objective functions $f_i : \mathcal{X} \to \mathbb{R}$, $i = \{1, 2\}$ are to be minimized. We only consider biobjective problems. While the definition of the modified hypervolume indicator is valid for any number of objective functions, its high computational complexity makes it only suitable for problems with two objectives.

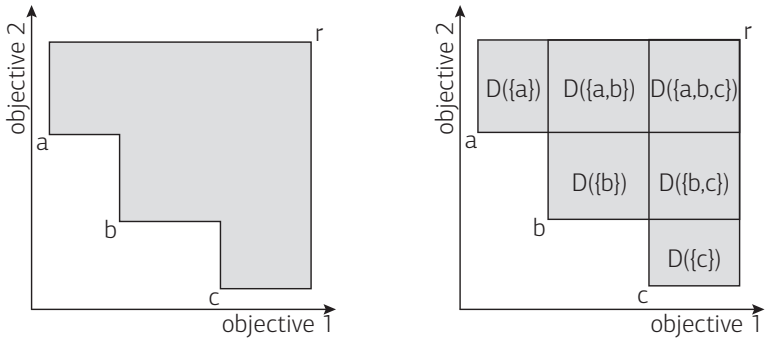


Figure 2.13 Original (left) and modified (right) hypervolume for a population of three solutions $\mathcal{A} = \{a, b, c\}$ with reference set $\mathcal{R} = \{r\}$. $\mathrm{D}(\{a, b\})$ for example is the diversity value of the subset $\mathcal{B} = \{a, b\}$.

We would now like to motivate the idea which is explained in detail in the next section. To this end we consider the example shown in the left part of Figure 2.13, with three solutions $\mathcal{A} = \{a, b, c\}$ and one reference point $\mathcal{R} = \{r\}$. We assume that the hypervolume indicator is given, plus an additional diversity measure $\mathrm{D} : 2^{\mathcal{X}} \to \mathbb{R}^{\geq 0}$ that returns the diversity of a subset $\mathcal{B} \subseteq \mathcal{A}$ of solutions. We would now like to integrate this diversity measure into the hypervolume indicator. The first idea which comes to mind is using a weighted sum. However, this approach comes with a serious drawback. Because only the non-dominated solutions have a contribution to the hypervolume, the dominated solutions are evaluated based on their contribution to diversity only. As solutions that are very diverse from non-dominating solutions usually also have very dissimilar objective values, this leads to populations where the non-dominated front optimizes the hypervolume and the dominated solutions optimize the diversity and are therefore randomly distributed instead of being close to the non-dominated solutions.

Therefore, our approach focuses on the hypervolume indicator. When looking at the hypervolume of a set of solutions, it can be seen that it is divided into partitions, where each partition is dominated by a specific subset of the whole population. In this study we propose to weight these partitions with the diversity of their dominating points before summing them up (see the

right part of Figure 2.13). Note that in the original hypervolume indicator, the partitions are weighted with one.

This adaptation has several nice properties. First, if a population is given, and the objective values of one solution improve, the modified hypervolume also improves. Second, if the diversity of a subset of the population improves (and the diversities of the remaining subsets remain the same), the modified hypervolume also improves. Third, if the diversity measure is chosen to be monotonically increasing with the number of solutions in the subset, adding a solution to the population cannot worsen the modified hypervolume. Fourth, it is more important that two solutions that are close in objective space are diverse than two solutions which are far apart in objective space. This is due to the fact that there are more partitions that are dominated by two close solutions than two far apart solutions.

2.5.2 · Modified Hypervolume

In this section we provide a formal definition of the modified hypervolume indicator. First we discuss diversity measures and the properties they should have, then we show how such set measures in general can be integrated into the hypervolume indicator.

Diversity measures have been discussed in Section 2.2, where it has been found that a diversity measure should fulfill the three requirements twinning, monotonicity in varieties and monotonicity in distance. In this section, a diversity function has to fulfill certain requirements such that the modified hypervolume indicator remains compliant with the underlying preference relation. First, the diversity of a set of solution must not decrease if a new solution is added to the set. Second, the diversity of a non-empty set of solution must be greater than zero, and the diversity of the empty set has to be zero. These properties are formally defined as follows:

P4: Monotonicity If $\mathcal{A}, \mathcal{B} \in \mathcal{X}$ are two sets of solutions for which $\mathcal{A} \subseteq \mathcal{B}$ holds, then $\mathrm{D}(\mathcal{A}) \leq \mathrm{D}(\mathcal{B})$.

P5: Positivity and null empty set For all $\mathcal{A} \in \mathcal{X} \backslash \emptyset$ it holds $\mathrm{D}(\mathcal{A}) > 0$, while $\mathrm{D}(\emptyset) = 0$

Note that while the monotonicity property P4 is satisfied by any measure fulfilling the monotonicity in varieties property P1 as defined in Section 2.2.1, the positivity and null empty set property P5 is fulfilled by most common diversity measures, including the measure by Solow and Polasky which is used in this thesis.

We now explain how any set-based function that fulfills the above properties can be integrated into the hypervolume indicator. As motivated before, we look at the hypervolume as a set of partitions that are dominated by a subset of the population. We call the solutions in $\mathcal{A}$ that dominate a certain point z the dominating points of z:

Definition 2.3 (dominating points): *Given a point $z \in \mathbb{R}^d$, and $\mathcal{A} \subseteq \mathcal{X}$ a set of solutions. We call the set of solutions* $\mathrm{dom}_A(z) := \{x \,|\, x \in \mathcal{A} \wedge f(x) \leqslant z\}$ *the subset of $\mathcal{A}$ dominating the objective vector z.*

We can say that if one set $\mathcal{A} \subseteq \mathcal{X}$ has a better or equal diversity in all hypervolume partitions than another set $\mathcal{B} \subseteq \mathcal{X}$, the set $\mathcal{A}$ is weakly preferred to set $\mathcal{B}$:

Definition 2.4 (diversity preference relation): *Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{X}$ be two sets of solutions and* D *a diversity function. $\mathcal{A}$ is weakly diversity preferred to $\mathcal{B}$, denoted $\mathcal{A} \preccurlyeq_D \mathcal{B}$ iff* $\forall z \in \mathbb{R}^d : \mathrm{D}(\mathrm{dom}_A(z)) \geq \mathrm{D}(\mathrm{dom}_B(z))$

This preference relation has the property that if $\mathcal{A}$ is weakly diversity preferred to $\mathcal{B}$, it is also weakly preferred to $\mathcal{B}$ according to Pareto dominance:

Theorem 2.5: *Given two sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{X}$ and a diversity measure* D*, then $\mathcal{A} \preccurlyeq_D \mathcal{B} \Rightarrow \mathcal{A} \preccurlyeq_{par} \mathcal{B}$ holds, where $\preccurlyeq_{par}$ is the extension of Pareto dominance to sets, i.e. $\mathcal{A} \preccurlyeq_{par} \mathcal{B}$ holds iff $\forall y \in \mathcal{B} : \exists x \in \mathcal{A} : f(x) \leqslant f(y)$.*

Proof. Remember that $\mathcal{A} \preccurlyeq_D \mathcal{B}$ means that for all points z in objective space, the diversity of the solutions in $\mathcal{A}$ that dominate the point, $D(\mathrm{dom}_{\mathcal{A}}(z))$ has to be larger than the diversity of the solutions in $\mathcal{B}$ that dominate the same point, $D(\mathrm{dom}_{\mathcal{B}}(z))$. Now it is obvious that if there is a solution in $\mathcal{B}$ dominating that point, then $D(\mathrm{dom}_{\mathcal{B}}(z)) > 0$ holds. If there would be no solution in $\mathcal{A}$ dominating that point z, $D(\mathrm{dom}_{\mathcal{A}}(z)) = 0$ would hold, which is a contradiction to $\mathcal{A} \preccurlyeq_D \mathcal{B}$. Therefore, each point in objective

space which is dominated by a solution in $\mathcal{B}$ is also dominated by at least one solution in $\mathcal{A}$, which means that $\mathcal{A} \preccurlyeq_{par} \mathcal{B}$.

More formally the proof can be given as follows: Given $\mathcal{A} \preccurlyeq_D \mathcal{B}$, by definition it holds that $\forall b \in \mathcal{B} : \mathrm{D}(\mathrm{dom}_{\mathcal{A}}(f(b))) \geq \mathrm{D}(\mathrm{dom}_{\mathcal{B}}(f(b)))$. We want to proof that this means that also $\mathcal{A} \preccurlyeq_{par} \mathcal{B}$ as defined in this theorem holds. Assume the contrary, i.e. $\exists b \in \mathcal{B} : \nexists a \in \mathcal{A} : f(a) \leqslant f(b)$. In this case, $\mathrm{dom}_{\mathcal{A}}(f(b)) = \emptyset \Rightarrow \mathrm{D}(\mathrm{dom}_{\mathcal{A}}(f(b))) = 0$. At the same time $\mathrm{D}(b) > 0$ and with monotonicity $\mathrm{D}(\mathrm{dom}_{\mathcal{B}}(f(b))) > 0$. Therefore $\mathrm{D}(\mathrm{dom}_{\mathcal{B}}(f(b))) > \mathrm{D}(\mathrm{dom}_{\mathcal{A}}(f(b)))$ which is a contradiction. □

Now we are able to formally define the diversity integrating hypervolume. The objective space is divided into hypervolume partitions. Each partition is dominated by a specific subset of the population. The partitions weight is equal to the diversity of that subset of solutions. To calculate the diversity integrating hypervolume, the partitions size multiplied with its weight is summed up.

Definition 2.7 (diversity integrating hypervolume): *Let $\mathcal{A} \subseteq \mathcal{X}$ denote a set of solutions. Furthermore, let $\mathrm{D} : 2^{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ be a diversity measure fulfilling the properties P4 and P5, and let $\mathrm{dom}_{\mathcal{A}}(z)$ according to Def. 2.3 give the subset of $\mathcal{A}$ dominating the objective vector z. Then the diversity integrating hypervolume indicator $I_H^{\mathrm{D}}(\mathcal{A})$ corresponds to a weighted Lebesgue measure of the set of objective vectors weakly dominated by the solutions in $\mathcal{A}$ but not by a so-called reference set $\mathcal{R} \in \mathcal{Y}$:*

$$I_H^{\mathrm{D}}(\mathcal{A}) = \int_{y \in \mathbb{R}^m} A_{\mathcal{A}}(y)\mathrm{D}(\mathrm{dom}_{\mathcal{A}}(y))dy$$

where $A_{\mathcal{A}}(y)$ is called the attainment function of set $\mathcal{A}$ with respect to a given reference set $\mathcal{R}$, and it holds that $A_{\mathcal{A}}(y) = 1$ iff $\exists a \in \mathcal{A}, r \in \mathcal{R} : f(a) \leqslant y \leqslant r$, else $A_{\mathcal{A}}(y) = 0$.

This indicator is a weak refinement of the diversity preference relation defined in Definition 2.4:

Theorem 2.8: *If a set $\mathcal{A} \subseteq \mathcal{X}$ is weakly diversity preferred to another set $\mathcal{B} \subseteq \mathcal{X}$, the modified hypervolume of set $\mathcal{A}$ is larger or equal the one of $\mathcal{B}$, i.e. $\mathcal{A} \preccurlyeq_{\mathrm{D}} \mathcal{B} \Rightarrow I_H^{\mathrm{D}}(\mathcal{A}, \mathcal{R}) \geq I_H^{\mathrm{D}}(\mathcal{B}, \mathcal{R})$.*

Proof. We know that $\mathcal{A} \preccurlyeq_D \mathcal{B} \Rightarrow \mathcal{A} \preccurlyeq_{par} \mathcal{B}$ (Theorem 2.5), therefore, $\{z \in \mathbb{R}^d : \alpha_{\mathcal{B}}(z) \neq 0\} \subseteq \{z \in \mathbb{R}^d : \alpha_{\mathcal{A}}(z) \neq 0\}$. Also, we know that $\mathrm{D}(\mathrm{dom}_{\mathcal{A}}(z)) \geq \mathrm{D}(\mathrm{dom}_{\mathcal{B}}(z))$. Therefore, $I_H^{\mathrm{D}}(\mathcal{A}, \mathcal{R}) \geq I_H^{\mathrm{D}}(\mathcal{B}, \mathcal{R})$. □

The optimization goal therefore is to find a population of a fixed size that maximizes the modified hypervolume indicator.

2.5.3 · DIVA Algorithm

First, we need to decide how the modified hypervolume indicator can be calculated. As we optimize a problem with two objective functions, we propose to use the hypervolume by slicing objectives algorithm [116] to calculate the hypervolume partitions, and then calculate the size and diversity of each partition. We therefore need a notion of the size of a partition dominated by a set $\mathcal{B}$: Out of a set $\mathcal{A}$, the size of the objective space solely dominated by solutions in $\mathcal{B} \subseteq \mathcal{A}$ is $s(\mathcal{A}, \mathcal{B}, \mathcal{R})$, where $\mathcal{R}$ is the set or reference points:

$$s(\mathcal{A}, \mathcal{B}, \mathcal{R}) = \int_{y \in \mathbb{R}^m} \alpha_{\mathcal{A}}^{\mathcal{B}}(y) dy$$

where

$$\alpha_{\mathcal{A}}^{\mathcal{B}}(y) = \begin{cases} 1 \text{ if } \exists r \in \mathcal{R} : y \leqslant r \wedge \mathrm{dom}_{\mathcal{A}}(y) = \mathcal{B} \\ 0 \text{ else} \end{cases}$$

The modified hypervolume can therefore be rewritten as:

$$I_H^{\mathrm{D}}(\mathcal{A}, \mathcal{R}) = \sum_{\mathcal{B} \subseteq \mathcal{A} \setminus \emptyset,\, \exists z \in \mathbb{R}^2 : \mathrm{dom}_{\mathcal{A}}(z) = \mathcal{B}} s(\mathcal{A}, \mathcal{B}, \mathcal{R}) \cdot \mathrm{D}(\mathcal{B})$$

The calculation of the modified hypervolume is shown in Algorithm 6. Using the measure of Solow and Polasky to quantify diversity, the calculation

1: **function** MODHYP($\mathcal{P}$)
2: $h = 0$ *(the indicator value)*
3: *(For all non empty hypervolume partitions)*
4: **for** all $\mathcal{B} \subseteq \mathcal{P} \backslash \emptyset$, $\exists z \in \mathbb{R}^2 : \mathrm{dom}_{\mathcal{P}}(z) = \mathcal{B}$ **do**
5: $h \leftarrow h + s(\mathcal{P}, \mathcal{B}, \mathcal{R}) \cdot \mathrm{D}(\mathcal{B})$ *(Increment indicator)*
6: Return h

Algorithm 6 Calculation of the modified hypervolume indicator I_H^{D}. Takes a population $\mathcal{P} \subseteq \mathcal{X}$ and returns the indicator value.

1: **function** NAIVEENVSEL($\mathcal{P}$, μ)
2: $\{\mathcal{P}', \mathcal{S}', \mu'\} =$ DUPLICATESELECTION($\mathcal{P}, \mu$)
3: **while** $|\mathcal{S}'| > \mu'$ **do**
4: *(Simulate removing each solution from the population. Remove the solution that induces the smallest indicator loss.)*
5: $\mathcal{S}' \leftarrow \mathcal{S}' \backslash \mathrm{argmax}_{x \in \mathcal{S}'} I_H^{\mathrm{D}}(\mathcal{S}' \backslash x)$
6: **return** $\mathcal{P}' \cup \mathcal{S}'$

Algorithm 7 Environmental Selection. Takes a Population $\mathcal{P}$, $|\mathcal{P}| \geq \mu$, and the number μ of selected individuals. $\mathcal{R}$ is the reference set. n is the population size. The whole environmental selection is of complexity $\mathcal{O}(n^7)$.

is of complexity $\mathcal{O}(n^3 + n^2 \cdot n^2) = \mathcal{O}(n^4)$, where the n^3 comes from the calculation of the matrix inverse of all pairwise distances as required by the Solow-Polasky diversity measure, and the $n^2 \cdot n^2$ is the calculation of the matrix inverses for the remaining partitions, using the block matrix inverse mentioned in Section 2.2.3. Note that without using the block matrix inverse, the calculation would be of complexity $\mathcal{O}(n^2 n^3) = \mathcal{O}(n^5)$.

Next, we need an environmental selection strategy. We propose to use Algorithm 7 where first any duplicates are thrown away using Algorithm 18, and then the standard greedy environmental selection scheme is used. In this greedy strategy, the solution with lowest fitness is removed until the population is of size μ'. The fitness of a solution is equal to the loss in the modified hypervolume if that solution is removed from the population. As soon as one solution is removed, the fitnesses of the remaining solutions are reevaluated. This greedy procedure is similar to the greedy strategy used

in the usual hypervolume selection scheme presented in Algorithm 20, with the only difference that I_H^D is used to calculate fitness instead of the normal hypervolume indicator I_H. The environmental selection furthermore differs from the standard hypervolume-based environmental selection scheme presented in Algorithm 21, because in DIVA, no non-dominated sorting is used. The reason is that dominated solutions usually have a higher diversity contribution than non-dominated solutions, and therefore can have a large impact on the modified hypervolume and should not be discarded prematurely.

The whole environmental selection algorithm including the indicator calculation is of complexity $\mathcal{O}(n \cdot n^2 \cdot \mathcal{O}(\mathrm{D}) \cdot n)$. The first n is the number of greedy steps in the environmental selection. The n^2 is due to the number of hypervolume partitions, the $\mathcal{O}(\mathrm{D})$ is the complexity of the diversity calculation, and the last n is the fact that the effect of removing a solution has to be calculated for each solution in each step. If using the measure of Solow-Polasky which is of complexity $\mathcal{O}(n^3)$, the complexity of the environmental selection equals $\mathcal{O}(n^7)$. Due to its high combinatorial complexity, this algorithm can only be applied to small population sizes, e.g. $|\mathcal{P}| \leq 10$.

However, using the fast diversity calculation strategy for the Solow-Polasky measure and storing relevant data (such as the distance matrix inverses of the dominating solutions of each partition), this complexity can be reduced to $\mathcal{O}(n^5)$, see Algorithm 8. Starting with the partition which is dominated by all solutions, the relevant matrix inverses of subsequent partitions can be calculated in quadratic time using the block matrix inverse. Calculating all matrix inverses therefore is of complexity $\mathcal{O}(n^3 + n^2 \cdot n^2) = \mathcal{O}(n^4)$, where the first $\mathcal{O}(n^3)$ comes from the matrix inverse calculation of the pairwise distances of all solutions, and the $\mathcal{O}(n^2 \cdot n^2)$ comes from the recalculation of all inverses for the remaining partitions. The diversity contribution of one solution in one particular partition can be calculated in linear time, see Section 2.2.3. Therefore, calculating all relevant diversity contributions is of complexity $\mathcal{O}(n^2 nn) = \mathcal{O}(n^4)$, where the n^2 is the number of partitions, the first n is the complexity of the contribution calculation (according to the fast diversity selection scheme), and the second n is because the contribution

```
1: function SPEDUPENVSEL(P, μ)
2:     {P', S', μ'} = DUPLICATESELECTION(P, μ)
3:                                  (Calculate matrix inverses and diversity contributions)
4:     Calculate matrix inverse of S'
5:     for all B ⊂ S'\∅, ∃z ∈ ℝ² : dom_S'(z) = B do
6:         Calculate matrix inverse of B, using the inverse of S'
7:         Calculate D(B) − D(B\s), ∀s ∈ B
8:                                                  (Iteratively throw away solutions)
9:     while |S'| > μ' do
10:                                        (Throw away solution with lowest contribution)
11:        s = argmax_{x∈S'} I_H^D(S'\x, R)
12:        S' ← S'\s
13:                                                  (Update partition information)
14:        for all B ⊆ S': s ∈ B ∧ ∃z ∈ ℝ² : dom_S'(z) = B do
15:            Update matrix inverse of B
16:            Calculate D(B) − D(B\s), ∀s ∈ B
17:    return P' ∪ S'
```

Algorithm 8 Environmental Selection. Takes a Population $\mathcal{P}$, $|\mathcal{P}| \geq \mu$, and the number μ of selected individuals. $\mathcal{R}$ is the reference set. n is the population size. Produces the same output as NAIVEENVSEL, but with a lower complexity of $\mathcal{O}(n^5)$.

of all solutions have to be calculated. Using these precalculated diversity contributions, the contribution of a solution to the overall modified hypervolume can be calculated with complexity $\mathcal{O}(n^2 n) = \mathcal{O}(n^3)$, where the first n^2 is the number of partitions, and the second n is due to the fact that the contribution of each solution has to be calculated. Finally, after selecting one solution for removal, all matrix inverses as well as the contributions have to be updated, which again is of complexity $\mathcal{O}(n^2(n^2 + n^2)) = \mathcal{O}(n^4)$, i.e. $\mathcal{O}(n^2)$ for the number of partitions and one $\mathcal{O}(n^2)$ to recalculate the matrix inverses and the other to update the diversity contributions. This whole loop of removing a solution and updating the matrix inverses and the diversity contributions has to be executed n times, therefore the overall complexity is $\mathcal{O}(n^5)$, leading to the same results as the naive implementation described above.

```
1: function DIVA(μ, λ, R, g)
2:     Initialize population P^1 randomly with μ solutions
3:     i = 1
4:     for g generations do
5:         O^i := VARIATE(P^i, λ)
6:         P^{i+1} := SPEDUPENVSEL(P^i ∪ O^i, μ)
7:                                          (Only take new population if not worse)
8:         if I_H^D(P^{i+1}, R) ≤ I_H^D(P^i, R) then
9:             P^{i+1} := P^i
10:        i = i + 1
11:    return P^i
```

Algorithm 9 Complete DIVA algorithm, optimizing the modified hypervolume indicator I_H^{D}. Input parameters: population size μ, offspring size λ, reference set $\mathcal{R}$; minimization of objective functions is done for g generations

Now that we have designed an environmental selection strategy, we can integrate it into DIVA, see Algorithm 9. To do so we adapt our reference SMOEA described in Algorithm 17 by replacing the standard selection scheme SELECT$(\mathcal{P}, \mu)$ with our DIVA selection scheme SPEDUPENVSEL$(\mathcal{P}, \mu)$, and the hypervolume indicator I_H with the modified hypervolume indicator I_H^{D}.

2.5.4 · Results

In this section, we first quantify the influence of the diversity parameter θ on the achieved hypervolume and diversity. Then we compare DIVA to the Omni-Optimizer, to the Standard Multi-objective Evolutionary Algorithm (SMOEA) as described in Appendix A, and to DIOP on the WFG testsuite. Also, DIVA, DIOP and the SMOEA are applied to the bridge construction problem.

Influence of θ

The parameter θ which is used in the calculation of the Solow-Polasky diversity measure can be used to adjust the tradeoff between diversity and hypervolume. The larger θ is, the sooner two solutions are considered as two different species, leading to a higher diversity. Therefore, with a large

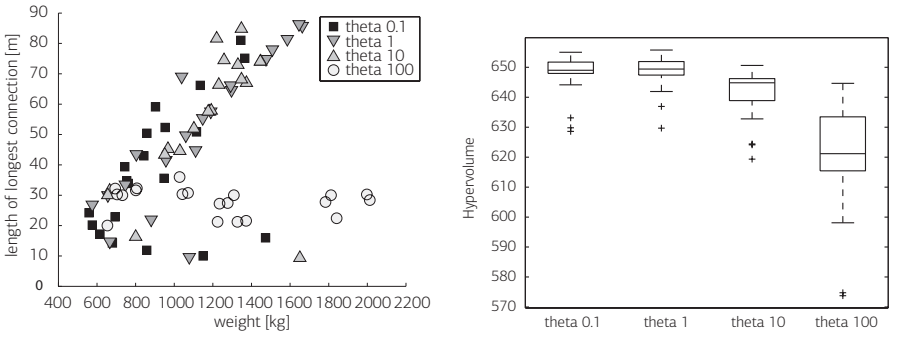
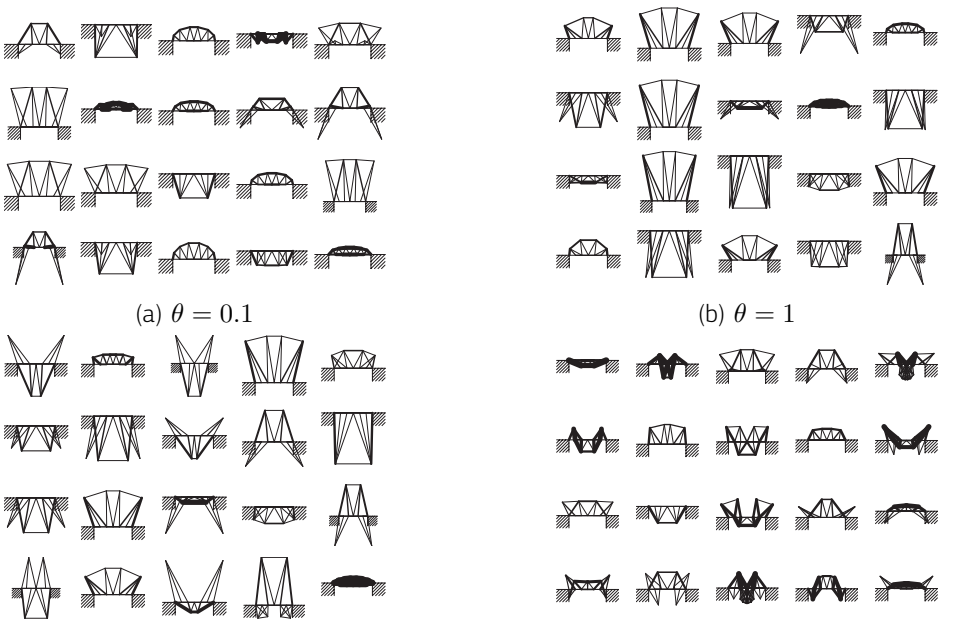


Figure 2.14 Left: Objective space values (not normalized) of the same populations shown in Figure 2.15. Right: Resulting hypervolume of all 30 runs on the bridge problem, for 4 different settings of θ.

θ, most sets of solutions have a diversity which is close to the maximally achievable value (i.e. the size of the set). With a low θ, on the other hand, all solutions are considered to lie close, leading to a diversity close to one for any set of solutions. Therefore, intuitively, working with a low theta should lead to populations similar to those achieved by the greedy hypervolume selection scheme, but without the beneficial effect of nondominated sorting.

To test the influence of θ, DIVA was run on the biobjective version of the bridge problem as described in Appendix B, where the first objective is the weight of the bridge, and the second objective is the length of the longest connection. As the two objectives are not within the same value range, we normalize the first/second objective by dividing it by 200 and 10, respectively.

We ran DIVA 30 times, for 50 000 function evaluations, with a population size of 20. The recombination/mutation probabilities were set to 0.5 and 1, respectively. The nadir point for the normalized objectives was 75 and 10 for the first/second objective. DIVA was run for 4 different settings of θ, namely $\theta \in \{0.1, 1, 10, 100\}$.



(a) $\theta = 0.1$ (b) $\theta = 1$

(c) $\theta = 10$ (d) $\theta = 100$

Figure 2.15 Example populations of one single run of the bridge problem, for different θ values. Corresponding objective values are shown in Figure 2.14.

The results for the remainder of Section 2.5.4 are tested for statistical significance according to the Kruskal-Wallis test with post-hoc Conover-Inman procedure [19] and with a significance level of 1%.

The objective space values of the resulting population of one run are shown in Figure 2.14. As can be seen, the population with $\theta = 0.1$ has more solutions on the front than the population with $\theta = 1$, whereas the populations with $\theta = 10$ and $\theta = 100$ do not lie on the front anymore. The corresponding bridges are shown in Figure 2.15.

The results of all runs are depicted in Figures 2.14 (hypervolume) and 2.16 (diversity). As expected, using a lower θ leads to a higher hypervolume[4],

[4] the difference between the $\theta = 0.1$ and the $\theta = 1$ run is not statistically significant.

as it starts to resemble the standard greedy hypervolume selection scheme. The runs with $\theta = 100$ always have a significantly lower hypervolume and diversity than the rest of the runs (except for the diversity calculated with $\theta = 100$). This can be explained as follows: Using such a high θ, all solutions are viewed as completely dissimilar, no matter what their true distance is (remember, the similarity is calculated as $\exp(-\theta \cdot d)$, where d is the distance between two solutions). Therefore, as stated before, the diversity of a given set is close to the size of the set. DIVA now tries to optimize the modified hypervolume, which in this case means that the hypervolume partition which is dominated by all solutions should be as large as possible.

To sum up, the parameter θ adjusts the number of solutions that lie on the front, which in turn directly relates to the achieved hypervolume. Lower values of θ lead to higher hypervolume values. When looking at the achieved diversity, the results depend on what setting of θ is used to calculate the diversity of the final population. If a lower value of θ is used, optimizations also using a lower value perform better. If a high value is used for diversity calculation, optimizations with a higher θ value perform better. There are limits, tough, as an optimization with $\theta = 100$ neither achieved a good hypervolume nor a good diversity.

Comparison of sMOEA, DIOP, DIVA, and the Omni-Optimizer

In this section, the proposed diversity-integrating optimizers DIOP and DIVA are compared against the sMOEA and against the Omni-Optimizer, an adapted version of NSGA-II which also integrates diversity in decision space. These four algorithms were run on the WFG testsuite (consisting of the testproblems WFG1-WFG9), as well as on the Omni problem (Equation (16) in [28], with $n = 5$ as suggested in the paper) which has been proposed together with the Omni-Optimizer. For the WFG problems, we use 2 objectives and 4 position and 20 distance dependent variables, i.e. a total of 24 decision variables. ε in DIOP was chosen to be 0.2. ε in the Omni-Optimizer was chosen to be 0. The population size was chosen to be 20 (for DIOP, the archive and target size were both 20), θ was chosen to be 1. Each algorithm was run 30 times on each problem, for 50 000 function evaluations on each WFG instance, and for 20 000 function evaluations on the Omni problem.

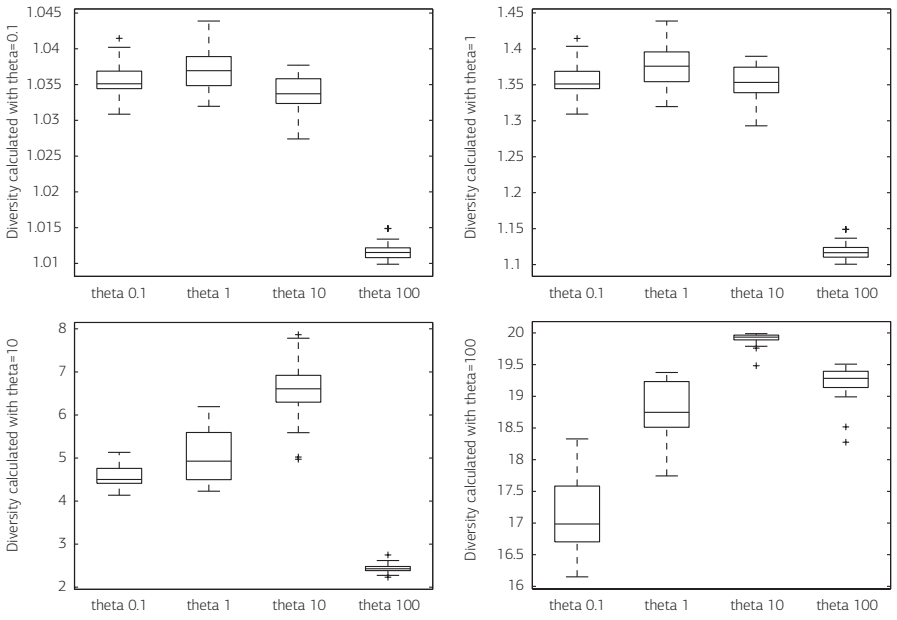


Figure 2.16 Resulting diversity of all 30 runs on the bridge problem. The four figures correspond to the θ setting used to calculate the final diversity. In each figure, four boxplots are shown, one for each of the 4 different settings of θ for which the population was optimized.

Both the WFG testproblems and the Omni problem have real-valued decision spaces. The variation setup is similar to the one in Section 2.4.3, where we used a standard variation scheme for real-valued decision vectors. In this scheme, $m/2$ random pairs of solutions are selected without replacement from $\mathcal{P}$ to generate the offspring population. These pairs are then recombined using the SBX crossover operator [24] with $\eta_c = 15$, where each pair is recombined with probability 0.5. During a recombination, each decision variable is recombined separately with probability one. With probability 0.5, the recombined values of this decision variable are exchanged between the offspring. After recombination, each individual is mutated with probability 1. To mutate an individual, each decision variable is mutated with probability $1/d$, where d is the length of the decision vector, using polynomial mutation [28] with $\eta_m = 20$.

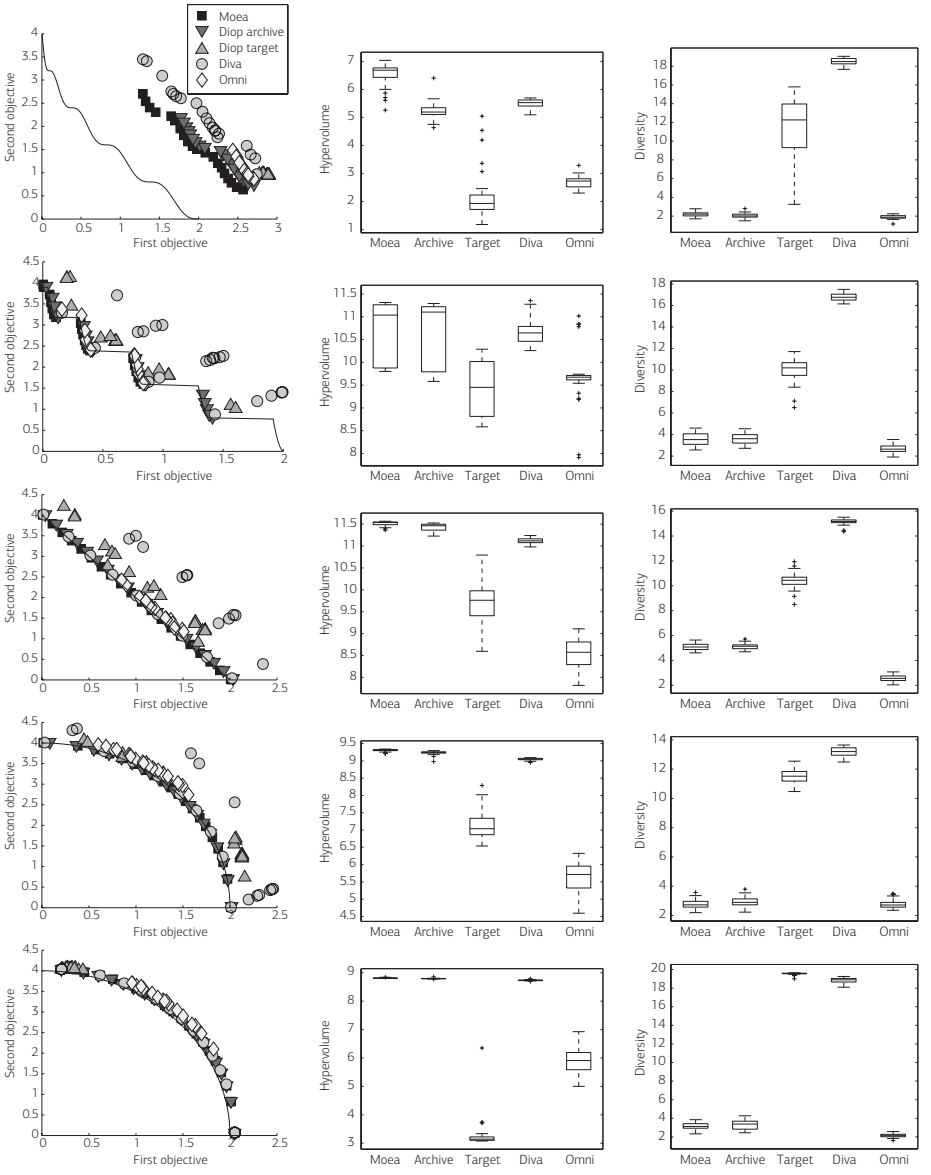


Figure 2.17 Results of the sMOEA, DIOP, DIVA and OMNI on several testproblems. Each row presents the solutions of one of the testproblems WFG1 - WFG5. The first column shows a random population for each algorithm, where the Pareto-front is shown as a black line. The second/third column shows the achieved hypervolume/diversity of each algorithm.

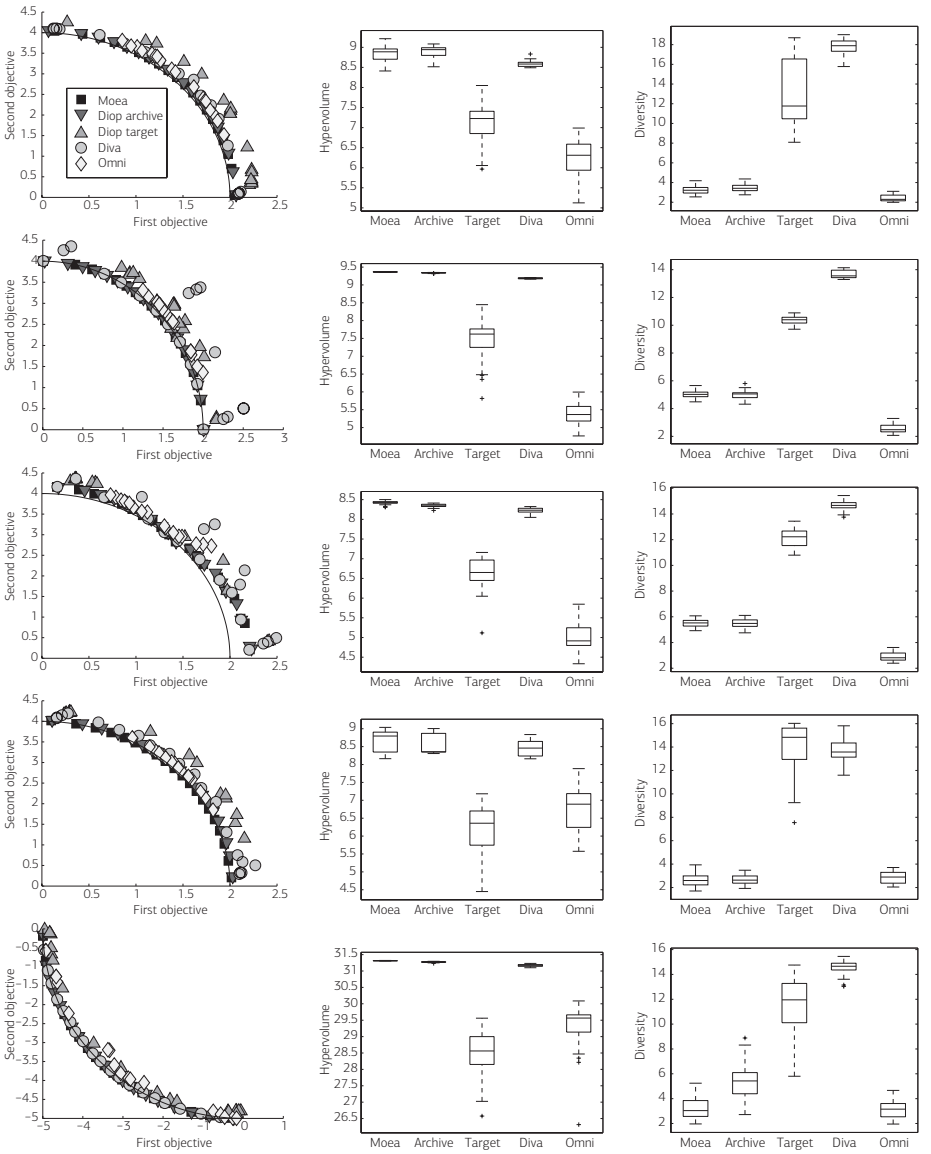


Figure 2.18 Results of the sMOEA, DIOP, DIVA and OMNI on several testproblems. Each row presents the solutions of one of the testproblems WFG6 - WFG9, and the Omni problem (last row). The first column shows a random population for each algorithm, where the Pareto-front is shown as a black line. The second/third column shows the achieved hypervolume/diversity of each algorithm.

The results can be seen in Figures 2.17 and 2.18, where each row corresponds to a test problem. The first column plots a random final population out of the 30 runs, while the second and third column plot the resulting hypervolume and diversity, respectively. The results of the statistical tests are shown in Table 2.9. As can be seen, the sMOEA never has a worse hypervolume than the other algorithms. The second best performing population in terms of hypervolume is the DIOP archive, and the third best is the DIVA population. There is no clear distinction between the DIOP target population and the OMNI population, where the DIOP target population is better in some problems, and worse in others. When looking at the DIOP target and OMNI populations in objective space, it can be seen that in general, the OMNI populations are closer to the front, but at the same time those populations are not well spread on the front. The DIOP target populations are further from the front, and sometimes well spread, sometimes not. Remember that the DIOP target population only optimizes diversity (within a certain distance to the front), therefore if the most diverse solutions can be found in a small region in objective space, the DIOP target population will concentrate in that region.

When looking at diversity, it can be seen that DIVA performs best, the DIOP target second best, the DIOP archive and the sMOEA third and OMNI worst. There are several explanations for the poor performance of the Omni-Optimizer. First, it was designed to be fast even when operating on large populations, while DIOP and DIVA will not work on population sizes of e.g. 1000. Secondly, the Omni-Optimizer does not optimize either the hypervolume or the Solow-Polasky diversity. It is unknown which measures it optimizes, as the used crowding distance only defines the contribution of a single solution to the diversity (both in decision and objective space), not the diversity of the whole set.

The use of the crowding distance as a diversity measure comes with a further problem as soon as the considered space has more than 2 dimensions. The problem is that a point which is far from any other point in terms of Euclidean distance might still get a bad crowding distance (low contribution to

diversity), whereas two points which are very close according to Euclidean distance might get a very good value, as shown in Example 2.10.

Example 2.10: Consider the following 8 points in three-dimensional Euclidean Space:

	d_1	d_2	d_3
p_1	-140	-140	-140
p_2	1	1	1
p_3	2	2	2
p_4	999	999	900
p_5	900	900	999
p_6	1000	1000	1000
p_7	1001	1001	1100
p_8	1100	1100	1001

The minimal distance to the nearest neighbor (MDIST), the contribution to the Solow-Polasky measure with $\theta = 1$ (SP) and the crowding distance (CD) of each point is as follows:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
MDIST	244.22	1.73	1.73	100.01	141.42	100.01	100.01	141.42
SP	0.84	0.01	0.01	0.30	0.45	0.04	0.30	0.45
CD	∞	426	2697	1197	2094	6	∞	∞

The minimal Euclidean distance for each point to any other point is larger than 100, with the only exception of the two points p_2, and p_3, which have an Euclidean distance of 1.73. Intuitively, either p_2 or p_3 should have the lowest contribution to diversity, as one of them is redundant. Using the Solow-Polasky measure with $\theta = 1$, p_2 and p_3 have the lowest contribution. The correlation coefficient between MDIST and SP is 0.9345. Without p_6 it is 0.9965. p_6 is an outlier, because it has a similar minimal nearest neighbor distance as p_4 and p_7, but lies in the middle of p_4, p_5, p_7 and p_8, thereby having a lower contribution to the Solow-Polasky measure.

	sMOEA	DIOP archive	DIOP target	DIVA	OMNI
sMOEA	0000000000 0000000000	+0+++0++0+ +00000000−	++++++++++ −−−−−−−−−−	+0++++++++ −−−−−−−−−−	++++++++++ +++0++++00
DIOP archive	−0−−−0−−0− −00000000+	0000000000 0000000000	++++++++++ −−−−−−−−−−	−0++++++0+ −−−−−−−−−−	++++++++++ +++0++++0+
DIOP target	−−−−−−−−−− ++++++++++	−−−−−−−−−− ++++++++++	0000000000 0000000000	−−−−−−−−−− −−−−+−−−0−	−0++−+++0− ++++++++++
DIVA	−0−−−−−−−− ++++++++++	+0−−−−−−0− ++++++++++	++++++++++ ++++−+++0+	0000000000 0000000000	++++++++++ ++++++++++
OMNI	−−−−−−−−−− −−−0−−−−00	−−−−−−−−−− −−−0−−−−0−	+0−−+−−−0+ −−−−−−−−−−	−−−−−−−−−− −−−−−−−−−−	0000000000 0000000000

Table 2.9 Pairwise significances of the Kruskal-Wallis test. Each entry $e_{i,j}$ of a given pair of algorithms i and j consists of two rows, where the first row gives the hypervolume results, and the second row gives the diversity results. In both rows of an entry $e_{i,j}$, there are 10 elements 0, $+$ or $-$, one for each testproblem, namely WFG1-WFG9 and the Omni-Problem. A $+/-$ means that the algorithm i was significantly better/worse than algorithm j, and a 0 means there was no statistically significant difference between in the two algorithms.

The crowding distance, on the other hand, is lowest for p_6, second lowest for p_2, and highest for p_3. The first solution to be discarded would not be the obvious choice of p_2 or p_3, but p_6. The correlation coefficient between the (finite) elements of mDist and cd is -0.0735, indicating that there is no correlation between the minimal Euclidean distance to the nearest neighbor and the crowding distance of a solution. We therefore discourage to use the crowding distance as a diversity measure for Euclidean spaces with 3 or more dimensions. ○

To sum up, the sMOEA, the DIOP archive and DIVA achieve a good hypervolume, whereas the DIOP target and the Omni-Optimizer achieve a poor hypervolume. At the same time, DIVA and the DIOP target achieve a good diversity, whereas the sMOEA, the DIOP archive and the Omni-Optimizer achieve a poor diversity. Therefore, DIVA is the only algorithm which achieves both a good hypervolume and a good diversity. However, remember that the use of DIVA is limited to biobjective problems and small population sizes. DIOP produces two populations, one which has a high hypervolume but a poor diversity, and the other which has a high diversity, but a poor hypervolume.

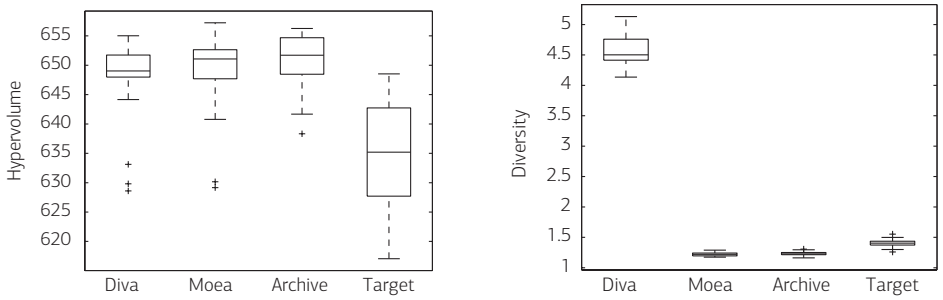


Figure 2.19 Hypervolumes and diversities achieved by DIVA, the sMOEA and DIOP on the biobjective bridge construction problem.

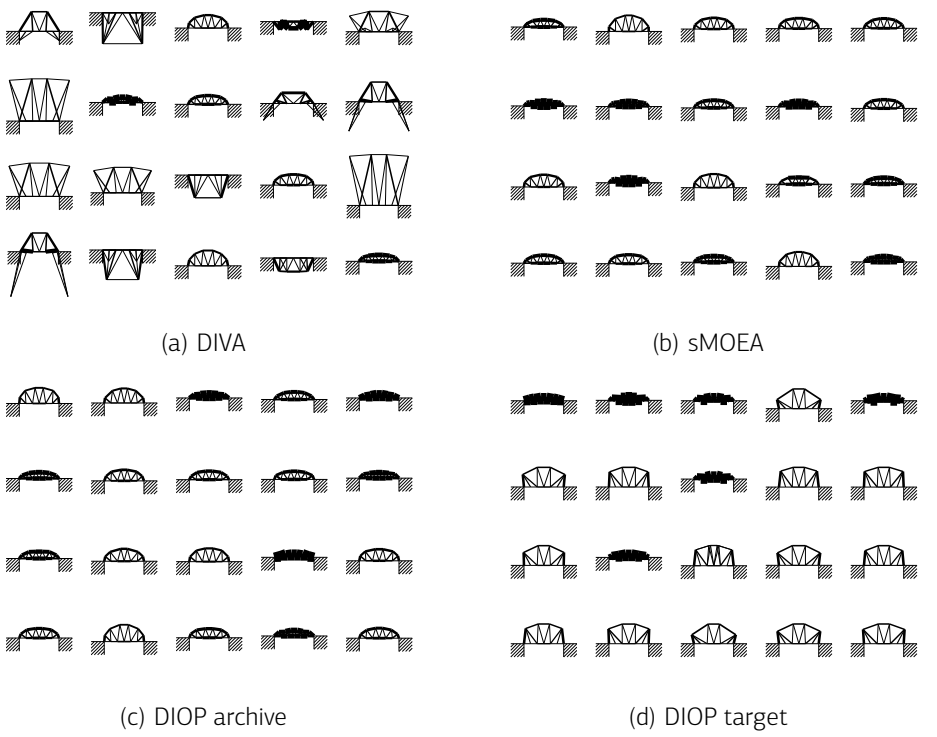
The sMOEA achieves the highest hypervolume. Finally the Omni-Optimizer produces populations with both a poor hypervolume and a poor diversity.

Bridge Optimization Problem

In this section, we compare the sMOEA, DIVA and DIOP on the bridge construction problem as described in Appendix B, using the same parameter setup as in Section 2.5.4, with $\theta = 0.01$. Note that we do not compare to the Omni-Optimizer, because the Omni-Optimizer only works for real valued or binary decision spaces, and it is not clear how to transform the bridge problem into a problem with a real valued or binary decision space.

The resulting hypervolumes and diversities can be seen in Figure 2.19. There is no significant difference between the hypervolumes achieved by DIVA, the sMOEA and the DIOP archive, whereas the hypervolumes achieved by the DIOP target are significantly lower. When looking at the diversity, a statistical test shows that DIVA has the highest diversity, the DIOP target the second highest, and the sMOEA and the DIOP archive the lowest.

The bridges of randomly selected populations can be seen in Figure 2.20. The corresponding points in objective space are shown in Figure 2.21. As can be seen, the bridges of the sMOEA and the DIOP archive look similar. The bridges of the DIOP target seem be more dissimilar, although all bridges fall into one of two categories: Flat bridges with large connection diameters, and



(a) DIVA (b) sMOEA

(c) DIOP archive (d) DIOP target

Figure 2.20 One random population per algorithm, for the bridge optimization problem.

tall bridges. The DIVA bridges look much more dissimilar than the sMOEA or the DIOP bridges. The reason for this can be seen in the objective space, as shown in Figure 2.21. The bridges of the sMOEA and the DIOP archive are all non-dominated with respect to the other solutions in their population. The DIOP target population contains dominated solutions, but all solutions are somewhat close to the front (as parametrized by ε). Finally, some of the DIVA bridges are quite far from the front, thereby allowing to increase the diversity of the population. At the same time there are enough bridges on the non-dominated front to yield an acceptable hypervolume.

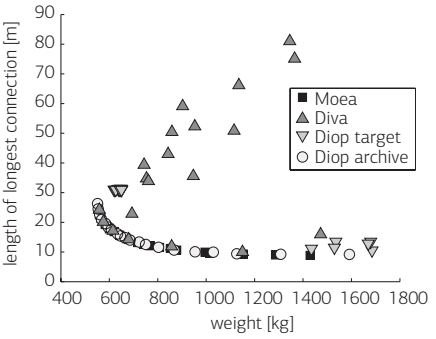


Figure 2.21 Objective space values of one random population per algorithm, for the bridge optimization problem.

2.5.5 · DIVA Summary

This section proposes DIVA, which works like a standard hypervolume-based evolutionary algorithm, but uses a modified version of the hypervolume that incorporates diversity. As a diversity measure, the measure proposed by Solow and Polasky is used. The tradeoff between objective function values and diversity can be adjusted by the parameter θ of the diversity measure. Tests on the influence of θ showed that a low θ leads to a population with a higher hypervolume, but a lower diversity than populations optimized with a higher θ.

DIVA was compared against the sMOEA, DIOP and the Omni-Optimizer, on the WFG testsuite and the Omni problem which has been proposed together with the Omni-Optimizer. It was found that DIVA leads to populations with both a good hypervolume and a good diversity, the sMOEA produces the best hypervolume, and DIOP produces an archive population with a good hypervolume, and a target population with a good diversity. The Omni-Optimizer performs worst both with respect to hypervolume and to diversity, which may be explained by the fact that it (a) does not explicitly optimize either the hypervolume or the diversity measure by Solow and Polasky, and (b) the used crowding distance is not suited for spaces with more than 2 dimensions.

Tests on a bridge optimization problem with DIVA, DIOP and the sMOEA showed that DIVA, the sMOEA and the DIOP archive produce the highest hypervolume without significant differences, whereas the best diversity is produced by DIVA, the second best by the DIOP target and the worst by the DIOP archive and the sMOEA. Therefore, DIVA performs best on the bridge construction problem in terms of achieved hypervolume and diversity.

In the future, it might be desirable to address the major shortcoming of DIVA, which is its computational complexity. Due to its complexity, the usage of DIVA is restricted to biobjective problems with small population sizes. It would be useful to find a way to approximate the modified hypervolume indicator, and/or the calculation of a solution's contribution to this indicator.

2.6 · Comparison of Approaches

This chapter aimed at optimizing diversity together with the usual optimization of objective function(s). We first discussed what a suitable diversity measure should look like, and found that the measure by Solow and Polasky fits our needs best, and therefore was used throughout the rest of the chapter. We proposed three diversity optimizing algorithms, one for single-objective problems, and two for multi-objective problems. NOAH, the algorithm for single-objective problems, tackles the problem of finding a set of solutions that is optimal in terms of diversity, while fulfilling a constraint on the objective function. DIOP, an algorithm for multi-objective problems, builds on this idea. It also finds a set of solutions with a maximal diversity, where the solutions fulfill a constraint in objective space in terms of their distance to a known approximation of the true Pareto-front. Finally, DIVA takes a different approach, where the diversity information is integrated into the hypervolume indicator, and this modified hypervolume indicator is then used for selection.

When comparing the two multi-objective algorithms DIOP and DIVA, the following considerations have to be taken into account: While the tradeoff

between closeness to the Pareto-front and diversity could be explicitly set by the user in DIOP, DIVA sets the tradeoff implicitly through the parameter θ, depending on the problem at hand. Different settings for θ will have to be tested in order to find the desired tradeoff. Also, DIVA has a high computational complexity. Furthermore, the solutions in the DIOP target population fulfill a quality constraint in terms of its objective values, and whether the solutions are distributed well in objective space depends on the problem. The solutions of DIVA, on the other hand, are distributed well over the objective space, and depending on the chosen θ value, the solutions can have poor objective values, as long as they contribute enough to diversity. To sum up, it is recommended to use DIVA for biobjective problems where the population size is relatively small, e.g. $\mu = \lambda = 20$, and if there is no quality constraint in objective space and/or if the optimization focus is more on diversity than on good objective space values. For larger population sizes, and if the decision maker has a clear idea of what objective values are acceptable, the use of DIOP is recommended.

3

Pareto-Set Analysis Through Clustering

In a multi-objective setting, the result of an optimization run is a set of compromise solutions. Users therefore have alternative solutions at hand that they can directly compare. However, the number of solutions can be large and the solutions can be time consuming to interpret, as there may be no intuitive visualization of all aspects of a solution, or there may be many decision variables. Therefore, comparing solutions may be time consuming and corresponding tools are desirable to support a decision maker in separating relevant from irrelevant information. Therefore, this chapter presents two methods to extract structural information from Pareto-set approximations that offer the possibility to present and visualize the trade-off surface in a compressed form.

The chapter is organized as follows: First, Section 3.1 gives an introduction of the problem and motivates why and when Pareto-set analysis can be useful. Section 3.2 discusses related work, whereas Sections 3.3 and 3.4

introduce two methods to analyze Pareto-sets. Finally, Section 3.5 compares the two methods and their results.

3.1 · Motivation and Background

When solving optimization problems, there are two scenarios about how the results can be used. In the first scenario, the end goal is to pick one preferred solution which will be used as-is. In the second scenario, the goal is to get new ideas about how a problem could be solved, and also justify the use of existing solutions by showing that alternative solutions do not dominate the existing solutions. In a real-world optimization problem such as the E/E-Architecture problem described in the introduction, the second scenario applies. The optimization serves as an inspiration of how an E/E-Architecture could be designed differently. It is therefore useful to approximate the set of Pareto-optimal solutions in order to learn about the underlying problem and to gain information that provides a better basis for decision making. By being presented with such a set of compromise solutions, the so-called Pareto-set approximation, the decision maker can not only study the relationships among the objectives, but also gain insights about the inherent structure of the problem.

In real-world problems, engineers who optimize such problems are not only interested in the objective values of the found solutions, but also in their structure, i.e. how the designs look like. For example considering the E/E-architecture problem, the engineers are not only interested in the cost and the complexity of an optimized architecture, but also in the bus structures of the optimized solutions, or whether these solutions were centralized, with a few master electronic control units (ECUs) or distributed with many small ECUs with comparable functionality. Therefore, a method to analyze the Pareto-front approximation should also include information about the decision space, and not only the objective values of the found solutions.

Including the decision space into the decision making process introduces a new level of difficulty. The interpretation of the achieved solutions might

be time-consuming, because the representation of each solution is complex. For example in the E/E-architecture problem, there were approximately 150 components that were partitioned to an average of 80 assembly units, and approximately 270 signals that were transmitted over 3 busses on average. In addition, the assembly units need to be placed in the car, microcontrollers and gateways have to be selected, and the physical wires have to be routed. There is no easy way to plot all aspects of an E/E-architecture. Comparing two architectures is even more onerous. The more detailed the model to be optimized becomes, the more time it takes to inspect the solutions returned by the optimization algorithm. Here, it is useful to have an automated method to structure the solutions, such that the engineer only has to look at a few of them, and knows that the remaining solutions are of a similar structure with similar objective values. If possible, the method should also highlight the specific similarities and differences between solutions.

To sum up, when developing methods that help with decision making, three problems have to be tackled. First, the Pareto-optimal solutions are difficult to interpret in objective space if there are more than two or three objectives. Second, they are also difficult to interpret if the decision space is complex and if there are many decision variables. Third, the set is also difficult to interpret because it contains a large number of solutions, and the decision maker might not have time to look at all of them. Nowadays, increased computing resources allow to cope with problems that have more and more decision variables and objectives, see e.g. [56, 92]. Therefore, in practice all three entities can become large and tools are needed that help the decision maker in analyzing the trade-off surface.

In this thesis, we focus on the problems caused by the fact that there are complex decision spaces and many solutions. The issue of dealing with many objective functions has been recently addressed in a few studies, see [13] for an overview. Different methods have been proposed to reduce the number of objective functions by omitting certain criteria such that the resulting error is minimized; this can be helpful both for assisting in decision making and for speeding up the search.

The problem of many decision variables has been mainly studied in the context of search, see e.g. [50, 113]; only few contributions exist in the context of Pareto-set analysis, see e.g. [76]. However, Deb and Srinivasan [27] have shown that important structural information in the decision space may be contained within a non-dominated set. This thesis proposes two methods, the Module-Annotating Hierarchical Clustering Algorithm (MANA) that aims at biobjective problems with binary decision spaces, and the Pareto-Front Analyzer (PAN) that aims at general optimization problems with an arbitrary number of objectives.

In MANA, the main idea is to identify modules of decision variables that are strongly related to each other. Thereby, the set of decision variables can be reduced to a smaller number of significant *modules*. Furthermore, the solutions are grouped in a hierarchical manner according to their module similarity. Overall, the output is a dendrogram where the leaves are the solutions and the nodes are annotated with modules that are contained in the solutions below that node.

The second method, PAN, helps the decision maker by clustering a given set of tradeoff solutions. The found clusters are compact and well separated both in decision and in objective space. A good clustering of the tradeoff solutions both in decision space and in objective space elicits information from the front about what design types lead to what regions in objective space. The novelty of PAN over existing work is its general nature, as it does not require the identification of distinct design variables or feature vectors. Instead, PAN only requires that a distance measure between a given pair of solutions can be calculated both in decision and in objective space. This clustering problem is formulated as a biobjective optimization problem, and a multi-objective evolutionary algorithm is used in PAN to generate promising partitionings.

3.2 · Related Work

The first approach presented in this thesis, which identifies modules of similar decision variable settings and clusters the solutions according to these modules, is strongly related to the concepts of building blocks as well as to biclustering. Building blocks [51] have already been used explicitly *during* both single-objective search, e.g., in the messy GA [39], and multi-objective search [113]. In the messy GA, promising building blocks are generated prior to the search. Here, we argue that an automated identification of those building blocks in a given Pareto-set approximation makes also sense *after* the search to assist in decision making. Unlike in the messy GA, we would like to generate building blocks based on problem specific information that is provided in the decision space. The consideration of the decision space is in fact crucial in the case of multi-objective optimization, as was indicated recently by Preuss et al. [83].

Identifying sets of decision variables that exhibit homogeneous behavior over a large number of solutions also corresponds to the concept of biclustering. Biclustering is a recent extension of standard clustering that aims at finding large homogeneous submatrices in a matrix, the so-called biclusters. Biclustering has become popular especially in computational biology, see Madeira and Oliveira [68] for a survey. Biclustering methods mainly differ in the definition of homogeneity, the distribution of the biclusters found and the strategies that are used to find the biclusters. One of the first biclustering algorithms presented in the literature was the one of Hartigan [49] which is, due to its simplicity, also used in the first method of this chapter. However, Hartigan's algorithm is not able to find biclusters that overlap which is its main drawback. An algorithm that not only allows the biclusters to overlap but also finds the exhaustive set of all biclusters[1] is Bimax [82]. Such an exhaustive search, however, is only applicable for small and/or sparse matrices.

In the second approach presented in this thesis, a clustering that is acceptable both in decision and in objective space is sought, for general de-

[1] Except for biclusters that are entirely contained in larger ones.

cision and objective spaces. This problem is closely related to traditional clustering, which aims at finding groups of points in such a way that the points within a cluster are as similar as possible, whereas points belonging to different clusters should be well distinguishable. Clustering is an unsupervised process that groups solutions based on how near they are to each other. This differs from classification, which uses supervised learning to derive rules to assign solutions to groups by using training data, i.e. given assignments that are known to be correct. Clustering problems have been known for a long time, see e.g. Xu and Wunsch [117] for a good overview of the field and an introduction into standard clustering techniques, including *partitional clustering* which is used in this work. Other techniques that are not considered in this chapter because they either place some assumptions on the solution space or do not produce crisp clusters are hierarchical, neural network-based, kernel-based, sequential or fuzzy clustering techniques.

The clustering problem tackled in this chapterhk2007 differs from traditional clustering as the considered points are characterized by two aspects, namely the decision space representation of solutions as well as their objective space values. We would like to group solutions such that the clusters are close in objective space, but at the same time exhibit strong similarities in decision space. Note that this is not the same as multi-objective clustering, as it is e.g. described by Handl and Knowles [46]. Multi-objective clustering aims at solving common problems in standard clustering, such as setting the tradeoff between cluster compactness, cluster separation and cluster number. It does so by transforming the clustering problem into a biobjective problem, a process which is also known as multiobjectivization, where the first goal is to optimize the cluster compactness and the second goal is to optimize cluster separation.

Clustering of data which is characterized by more than one aspect has recently gained attention in the field of bioinformatics, where for instance genes need to be grouped according to their mRNA expression profiles and their protein interaction partners. A commonly used approach combines this data into one matrix and then applies conventional clustering techniques [33]. In our case, the cluster measures are different for the objective

and the decision space, so merging the two spaces is not an option. Other approaches consider both datasets separately, but are designed to find only a single best cluster [17, 64]. In this study, however, we would like to find multiple groups of solutions. Bushel et al. [15] use a common distance measure, i.e. the sum of Euclidean distances in both spaces, and then apply the k-means clustering algorithm to derive the groups. In our problem however we are considering data sets where the best partitioning in decision space might be different from the best partitioning in objective space, and we would like to generate the tradeoff solutions in between. Pollard and van der Laan [81] apply iterative clustering, which means that the data is first clustered in one space, and the resulting clusters are then clustered again in the other space. This process can be repeated, or the order of the spaces can be reversed. This approach is similar to the approach proposed by Aittokoski et al. [1], that applies a modified k-means algorithm to cluster the solutions in objective space. For a refinement, the same algorithm can be applied to group the solutions of individual clusters in decision space. Finally, Narayanan et al. [74] propose a measure to quantify the goodness of clusters in different spaces. They assume that each space can be transformed into a graph, where the nodes are the genes and the edges are the relations between genes. Here, different relations can be modeled in different graphs. The measure then calculates for each cluster a score on each graph and the worst score over all graphs is selected as the representative score for that cluster. The partitioning goodness measure then is defined as the sum of these representative scores of all clusters.

Some recent efforts have been undertaken in order to infer relationships between decision and objective space, which helps to extract design principles that can be useful to the decision maker. One such method is called 'innovization' (innovation through optimization), see Deb and Srinivasan [26]. To be able to apply innovization it is assumed that the decision space is built from real and/or discrete decision variables which can take certain values. In earlier innovization approaches [26], solutions were examined manually on a specific problem to derive interesting facts about variables such as common variable settings, variable importance, and relations between variable settings and objective values. A more recent approach [4],

automates this process by first using clustering in objective space and then fitting some basis functions to model the data in the cluster.

Other approaches aim at visualizing the Pareto-front and/or the Pareto-optimal solutions in decision space, and inferring design principles from this visualization. One such approach is using self-organizing maps (SOMs) [77], where high-dimensional decision and objective spaces are mapped to two-dimensional maps. Another approach is using heatmaps [84], where real-valued variable and/or objective vectors of a set of solutions are plotted as colored heatmaps. Both approaches assume that the decision space is a real-valued space.

Some work has also been done in order to do feature extraction. Sheng et al. [95] assume that each solution can be described as a set of features, which can, but do not have to be equal to the decision variables. They then optimize a partitioning using an evolutionary algorithm, where they also evolve a subset of features which is to be taken into account when calculating the partitioning goodness. Sugimura et al. [101] also assume that there are design variables, and mine for design rules that specify which variable settings lead to which fitness levels. Note that all of the previously mentioned approaches make some assumptions about the decision space, i.e. that there is a given set of continuous or discrete design variables, such that each solution can be represented as a vector of real or discrete values. It is also assumed that solutions with similar vectors have similar designs. Considering the E/E-architecture problem described in the introduction, it might be difficult to define the space of all possible architectures using design variables. In fact, we decided to represent an E/E-architecture as a hierarchical partitioning with labeled nodes. Our approach aims at such problems with complex decision spaces, as the only requirement of our approach is that it is possible to measure the distance or dissimilarity between any two solutions.

There has been a multitude of approaches to do clustering using evolutionary algorithms, see e.g. Hruschka et al. [54] for a comprehensive overview of current approaches. These approaches mainly differ in the used representations, variation operators, fitness functions (i.e. the used cluster validity

index) and whether the number of clusters is variable or is assumed to be fixed.

Also, clustering has been used to prune a given set of tradeoff solutions e.g. produced by a multi-objective optimizer in order to help the decision maker. Typically, this clustering is done solely in objective space. Taboada and Coit [102] apply the k-means algorithm for all possible number of clusters. Morse [73] uses both partitional and hierarchical clustering. Rosenman and Gero [88] tackle the problem of differently scaled objectives.

Finally, there has been some work that aims at maintaining diversity in decision space during optimization, see for example [90], or Chapter 2 of this thesis. If there are so-called preimages (i.e. distinct regions) in the decision space that map to the whole Pareto-optimal front, a decision maker might be interested in finding all of those preimages. In such cases, clustering the solutions not only in objective space, but also in decision space is advantageous.

3.3 · Binary Decision Spaces with Two Objectives

This section presents the Module-Annotating Hierarchical Clustering Algorithm (MANA), a method that helps interpreting Pareto-sets or approximations thereof, which have many solutions and many decision variables. The main idea is to identify sets of decision variables, called modules, that are strongly related to each other. For binary decision spaces, a solution is said to contain a module if all decision variables belonging to the module are set to 1 in that particular solution. We are looking for large modules that are contained in as many solutions as possible. Using these modules, the solutions can be clustered hierarchically. This hierarchical clustering can be visualized both in decision and objective space, yielding information about the relationship between decision space and objective space.

3.3.1 · Problem Setting

We here follow the notation introduced in Section 1.1: Suppose we have a multi-objective minimization problem $f : \mathcal{X} \to \mathcal{Y}$, $f = \{f_1, ..., f_m\}$. Here, we only consider biobjective problems, i.e. $\mathcal{Y} \subseteq \mathbb{R}^2$, and only binary decision spaces with d decision variables, i.e. $\mathcal{X} \subseteq \{0,1\}^d$. Furthermore, assume we are given a Pareto-set approximation, i.e. a set of non-dominated solutions, which can, but does not have to be generated by a multi-objective optimizer. Such a Pareto-set approximation can be considered as a set of *decision vectors* $\{x^1, \ldots, x^n\} \subseteq \mathcal{X}$ that are mutually non-dominated. In this thesis, we represent a Pareto-set approximation as a *decision matrix* $\Xi \in \mathcal{M}_{n,d}(\{0,1\})$ where $\mathcal{M}_{n,d}(\{0,1\})$ is the set of binary matrices with n rows and d columns.

Definition 3.1 (decision matrix): *A* decision matrix $\Xi = (\xi_{i,j})_{n \times d}$ *is a matrix with d columns and n rows that is composed of the decision vectors $x^r = (\xi_{r,1}, \ldots, \xi_{r,d})$ of n solutions ($1 \leq r \leq n$).*

In practice, two main problems emerge. The first problem is that there are too many decision variables. The methods proposed in this thesis tackle this problem by merging the decision variables into so-called *modules*.

Definition 3.2 (module): *A* module *is a subset $S \subseteq \{1, \ldots, d\}$ of the decision variables.*

These modules are then used to generate a new reduced representation of the decision variables.

The second problem is that there are too many solutions. This problem is tackled by grouping solutions hierarchically. Sec. 3.3.2 introduces a method to generate such a grouping based on modules. Both the problem of finding the best reduced representation and the problem of grouping the solutions are formalized in the following.

Transformation to a New Representation

When identifying modules, the goal is to find a small set of large modules. By representing modules instead of decision variables, a reduced representation of a decision matrix can be achieved. More precisely, given a set of

modules $S = \{S_1, \ldots, S_l\}$, we would like to transform the decision matrix $\Xi \in \mathcal{M}_{n,d}(\{0,1\})$ into a new representation, the *module matrix* Υ wherein the rows correspond to the original solutions in Ξ and the columns correspond to the modules in S. For a certain solution x^r, the ith bit in the new representation y^r is set to 1 if and only if the original representation contains the module S_i, i.e., if and only if all decision variables belonging to S_i are set to 1 in x^r.

Definition 3.3 (module matrix): *Given a decision matrix* $\Xi = (\xi_{i,j})_{n \times d} \in \mathcal{M}_{n,d}(\{0,1\})$ *and a set of modules* $S = \{S_1, \ldots, S_l\}$*, the function* $T_{\Xi \to \Upsilon}(\Xi, S)$ *yields a corresponding module matrix* $\Upsilon = (\upsilon_{i,j})_{n \times l}$*, which is defined as* $\upsilon_{r,c} = 1 \Leftrightarrow \forall i \in S_c : \xi_{r,i} = 1$ *for all* $1 \leq r \leq n$ *and* $1 \leq c \leq d$*. Each row of* Υ *is called a* module vector*.*

Note that we here assume that whenever a module is selected, all contained decision variables are set to 1. In general, one could consider an arbitrary variable assignment representing the module; for reasons of simplicity, we do not consider this further.

Example 3.4: Consider a decision matrix Ξ with five solutions and decision vectors of length 5 as depicted on the left of Figure 3.1. In addition, the module set S consists of three modules $S_1 = \{1,2,3\}$, $S_2 = \{2,3,4\}$, and $S_3 = \{4,5\}$. The above defined transformation $T_{\Xi \to \Upsilon}$ maps the decision matrix Ξ to the new representation $\Upsilon = T_{\Xi \to \Upsilon}(\Xi, S)$ as shown on the right of Figure 3.1. For example, the decision vector x^3 has ones at the positions 1 to 4 and therefore contains both modules $S_1 = \{1,2,3\}$ and $S_2 = \{2,3,4\}$ but not module S_3 since the fifth bit is not set to 1. Therefore, its corresponding module vector y^3 in Υ contains ones at the positions 1 and 2 and a zero at position 3. ◦

Note that in the above example, the module matrix can cover all 1s in the original decision matrix. In general, this is not the case as the following example shows.

Example 3.5: Consider the decision vector x^1 in Figure 3.2 and the same modules as in the Example above. Since x^1 only contains module S_3 but

	1	2	3	4	5
x^1	1	1	1	0	0
x^2	1	1	1	0	0
x^3	1	1	1	1	0
x^4	0	1	1	1	0
x^5	0	0	0	1	1

decision matrix $\Xi = (\xi_{ij})$ with decision vectors x^r

$\xrightarrow{T_{\Xi \to \Upsilon}}$

	S_1	S_2	S_3
y^1	1	0	0
y^2	1	0	0
y^3	1	1	0
y^4	0	1	0
y^5	0	0	1

module matrix $\Upsilon = (\upsilon_{ij})$ with module vectors y^r

Figure 3.1 Illustration of the decision matrix/module matrix concept for the given modules $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 3, 4\}$, and $S_3 = \{4, 5\}$.

not S_1 and S_2, the bit $\xi_{1,2}$ cannot be reconstructed with the module representation Υ. Therefore with the transformation $T_{\Xi \to \Upsilon}$ information is lost. ∘

To measure the information loss described in the previous example, we interpret a module matrix again as a decision matrix by retransforming it with the following function.

Definition 3.6 (module vector): *Given a set $S = \{S_1, \ldots, S_l\}$ of modules, a module vector $y^r = (\upsilon_{r,1}, \ldots, \upsilon_{r,l})$ can be interpreted as the decision vector $T_{\Upsilon \to \Xi}(y^r, S) = (\xi_{r,1}, \ldots, \xi_{r,d})$ where a bit $\xi_{r,c}$ is set to 1 if at least one entry $\upsilon_{r,i}$ in y^r is set to 1 for which the module S_i contains the column c, i.e., $\xi_{r,c} = 1 \Leftrightarrow \exists S_i \in S : \upsilon_{r,i} = 1 \wedge c \in S_i$ for all $1 \leq r \leq n$ and $1 \leq c \leq d$.*

When reducing the decision matrix to the module matrix, we want to achieve the smallest representation while most of the information has to be preserved. More formally, we assess a certain transformation by computing an error function $e(\Xi, T_{\Upsilon \to \Xi}(T_{\Xi \to \Upsilon}(\Xi, S), S))$ between the original decision matrix Ξ and the corresponding retransformed module matrix $T_{\Upsilon \to \Xi}(T_{\Xi \to \Upsilon}(\Xi, S), S)$. This error can be defined with respect to both decision space and objective space. Here, we use the following two error functions:

Definition 3.7 (error function): *Let* $\Xi = (\xi_{i,j})_{n\times d} \in \mathcal{M}_{n,d}(\{0,1\})$ *and* $\Xi^T = (\xi^T_{i,j})_{n\times d} \in \mathcal{M}_{n,d}(\{0,1\})$ *be two decision matrices. Then, one possible error function with respect to decision space is the Hamming distance between the matrices:*

$$e_{\text{dec}}(\Xi, \Xi^T) := \sum_{1\leq i\leq n} \sum_{1\leq j\leq d} |\xi_{i,j} - \xi^T_{i,j}|$$

An error function with respect to to objective space can be defined as

$$e_{\text{obj}}(\Xi, \Xi^T) := \sum_{1\leq i\leq n} I_\varepsilon \left(f((\xi^T_{i,1}, \ldots, \xi^T_{i,d})), f((\xi_{i,1}, \ldots, \xi_{i,d})) \right)$$

where I_ε *is the binary additive epsilon indicator of [123]. Note that other quality indicators like the hypervolume indicator in [123] can be used as well. The second error function gives an idea of the change in objective vector values if the new module representation is used.*

Now, we can state the problem of finding a best set of modules according to a given error function:

Problem 3.8 (bi-objective module selection): Let $\Xi \in \mathcal{M}_{n,d}(\{0,1\})$ be a decision matrix and $e : \mathcal{M}_{n,d}(\{0,1\}) \times \mathcal{M}_{n,d}(\{0,1\}) \to \mathbb{R}$ an error function that computes an error between two arbitrary decision matrices. Then, the bi-objective problem of *simultaneously selecting a module set and minimizing the number of modules* can be stated as finding a set $S = \{S_1, \ldots, S_l\}$ such that both the number of modules l and the error $e(\Xi, T_{\Upsilon\to\Xi}\left(T_{\Xi\to\Upsilon}\left(\Xi, S\right), S\right))$ are minimized.

Theorem 3.9: *Problem 3.8 is* $\mathcal{NP}$*-hard with respect to the error function* $e_{\text{dec}}(\Xi, \Xi^T) := \sum_{1\leq i\leq m} \sum_{1\leq j\leq n} |\xi_{i,j} - \xi^T_{i,j}|$.

The proof is given in [109]. Methods to tackle this module selection problem are presented in Sec. 3.3.2.

Grouping Solutions by Using Structure Information

Given a set of modules, we would like to reduce the number of solutions by merging them into hierarchical groups. The goal is to generate groups whose solutions are as similar as possible. This in general corresponds to

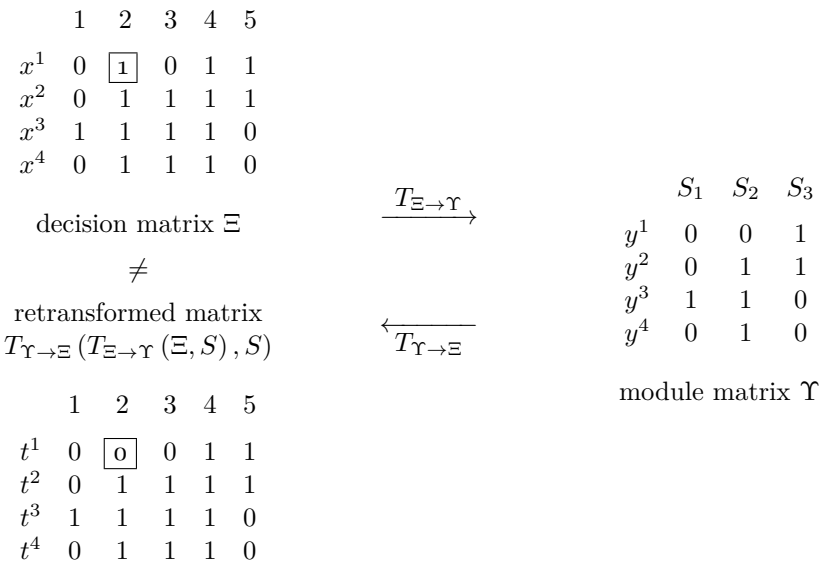


Figure 3.2 An example where the retransformation does not yield the original decision matrix. The modules are defined as $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 3, 4\}$, and $S_3 = \{4, 5\}$.

the task of clustering. Instead of setting the number of groups a priori, we would like to be able to traverse the group hierarchy from the largest group, which contains all solutions, to the smallest groups where each group consists of only one solution. To achieve this, we propose to use dendrograms to represent the grouping structure. The resulting groups should strongly depend on the modules found, such that each group can be uniquely defined by a sequence of modules that are selected in this group. To this end, module-annotated dendrograms are introduced.

In general, a dendrogram is a binary tree that can be used to represent a hierarchically organized grouping structure. An example is given in Figure 3.3. The nodes are distributed on so-called levels, i.e., each node has a fixed distance from the root. In a module-annotated dendrogram, each level has exactly one node, reflecting the order in which modules are selected for the grouping. Each node is associated with one module, where

solutions containing the module all belong to the left branch of the node, and solutions that do not contain the module belong to the right branch. The leaves represent the rows of the decision matrix, i.e., the solutions in a Pareto-set approximation. The branches represent groups that contain all solutions (leaves) below that branch. In general, solutions and groups of solutions that lie close to each other have many modules in common and therefore have a high similarity.

We consider the goal of identifying the dendrogram that minimizes the distances of the solutions within the groups. As a distance measure of a group $G \subseteq \{1, \ldots, n\}$ of solutions, we use the average pairwise Hamming distance $s(G) := 1/\binom{|G|}{2} \sum_{r,s \in G} d_{\mathrm{H}}(x^r, x^s)$ where the Hamming distance between two points $x^r = (x^r_1, \ldots, x^r_d)$ and $x^s = (x^s_1, \ldots, x^s_d)$ is defined as $d_{\mathrm{H}}(x^r, x^s) = \sum_{1 \leq j \leq d} |x^r_j - x^s_j|$. For evaluating an entire dendrogram, we use the intra-group distance measure as defined above averaged over all groups in a level cut and averaged over all these cuts. A level cut divides the dendrogram horizontally, such that with each level cut a set of groups is associated. For example, the level cut between S_2 and S_3 in Figure 3.3 contains three groups: The one where all solutions contain S_1 and S_2 (left subtree), one where all solutions contain S_1 but not S_2 (middle) and the third where all solutions neither contain S_1 nor S_2 (right subtree).

Definition 3.10 (distance measure of dendrograms): *As* distance measure *s of a dendrogram D with the level cuts $C_1, \ldots, C_m \subseteq 2^{\{1,\ldots,n\}}$, where each level cut C_i is a set of groups $C_i = \{G_{i,1}, \ldots, G_{i,|C_i|}\}$ ($G_{i,j} \in \{1, \ldots, n\}$) we propose the average pairwise intra-group Hamming distance, averaged over all groups and all cuts. The number of groups associated with a cut is equal to the number of intersections between the cut and the dendrogram branches.*

$$s(D) := \frac{1}{n} \sum_{1 \leq i \leq n} \frac{1}{|C_i|} \sum_{1 \leq j \leq |C_i|} \frac{1}{\binom{|G_j|}{2}} \sum_{r,s \in G_{i,j}} d_{\mathrm{H}}(x^r, x^s).$$

Overall, this leads to the following problem which has been shown to be $\mathcal{NP}$-hard [65].

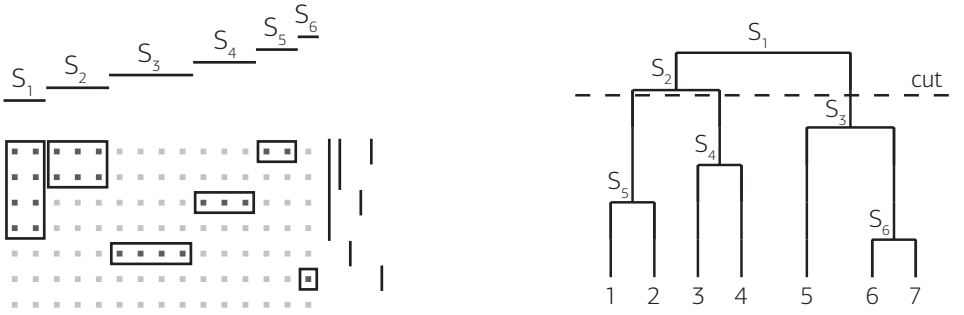


Figure 3.3 Example of a dendrogram with additional module annotations (right) for a given decision matrix (left). The solutions are denoted by the numbers from 1 to 7 and the modules by S_1 to S_6. The vertical lines on the right of the decision matrix indicate the corresponding groups.

Problem 3.11 (finding the optimal dendrogram): Given a decision matrix Ξ, the problem of *finding the optimal module-annotated dendrogram* corresponds to finding the dendrogram D with the lowest distance measure $s(D)$ as defined in Definition 3.10.

3.3.2 · MANA Algorithm

We propose MANA to solve the two problems presented in the previous section. Since the two problems are $\mathcal{NP}$-hard, we propose corresponding heuristics in the following. More precisely, we propose (i) two approaches based on biclustering for approximating the module selection problem and on that basis (ii) a method to construct a module-annotated dendrogram. To apply MANA, one of the proposed biclustering algorithms has to be selected in order to calculate an approximation of the optimal set of modules. After the modules have been generated, MANA uses these modules to produce an approximation of the optimal dendrogram.

Module finding

As described in Sec. 3.3.1 we would like to find modules that exhibit homogeneous behavior over many solutions. This problem corresponds to the task

of biclustering. In the following, a bicluster is defined as a submatrix of Ξ that only contains ones. Each of these biclusters forms a module consisting of the bicluster's columns. Here, we are using two exemplary biclustering algorithms: Hartigan's algorithm [49] and Bimax [82].

Both algorithms have their advantages and drawbacks but due to their complementary behavior, we selected them as representative examples of biclustering algorithms. The Hartigan algorithm is the first proposed biclustering algorithm, and many other algorithms are based on its principles, cf. [68]; it is simple and fast. In contrast to the Bimax algorithm, it limits the number of possible biclusters substantially as it does not find overlapping biclusters. The Bimax algorithm, however, finds all possible inclusion maximal biclusters, i.e., all biclusters that are not contained in larger ones. As the number of all biclusters is in general exponential in the matrix size this algorithm is impractical for larger matrices.

Hartigan's Algorithm: Hartigan's algorithm is based on a simple divide-and-conquer strategy; it iteratively divides the decision matrix into smaller submatrices. Due to this strategy, the order of the rows and columns of the decision matrix is fixed as soon as the splitting starts. The matrix therefore has to be sorted prior to algorithm execution. To be able to identify large biclusters, an appropriate sorting measure is essential.

In this thesis, we use two criteria for the initial sorting: one sorts according to the Hamming distances in decision space and the other sorts according to the objective space values. The first criterion places the two solutions with the highest Hamming distance as first and last row, making them the upper and lower border solution. It then iteratively selects the solution with the smallest Hamming distance to either border solution, places it next to this border solution and makes it the new respective border solution. The second criterion is restricted to two-objective problems; it sorts the solutions in the decision matrix according to their values of the first objective.

After sorting, the iterative splitting of the matrix takes place. In each step, the theoretical best split for each existing submatrix is calculated and the best overall split is performed by splitting one of the existing submatrices

into two new submatrices. The algorithm stops as soon as each submatrix contains only ones or only zeros. As a splitting measure for Hartigan's algorithm, we take the following *percentage split measure*, defined as

$$Q(M_1, M_2) = \left| \frac{\#\text{ ones in } M_1}{|M_1|} - \frac{\#\text{ ones in } M_2}{|M_2|} \right|$$

where M_1 and M_2 are the two submatrices resulting from the split. This split measure has to be maximized in order to find the best split.

Bimax: The recursive Bimax algorithm performs an exhaustive search for the set of all biclusters using a branch-and-bound strategy. Even for reasonably sized matrices, the number of biclusters found can become high. Therefore, we use a heuristic method to prune the set of biclusters found. The pruning method iteratively selects the bicluster which covers most of the remaining 1s. The remaining 1s are defined as the 1s not yet covered by any selected bicluster. This iteration stops if either a predefined number of selected biclusters is reached or all 1s of the matrix are covered by the selected biclusters.

Grouping Solutions Within Dendrogram

To create a module-annotated dendrogram, hierarchical clustering could be used on the reduced representation, in which case each module would contribute equally to the grouping. Here, however, we would like a group to be defined by the sequence of modules that are selected in all solutions of the group. We propose the following simple approach. The grouping starts according to the largest bicluster. This bicluster divides the solutions into two groups, namely those solutions which contain the module given by the bicluster and those that do not. This is the root of the dendrogram. Then, the next largest bicluster has to be selected where the size of a bicluster is defined as the number of ones covered by this bicluster that are not covered by any previously chosen bicluster. The generation of the dendrogram stops if all groups contain only one solution.

3.3.3 · Experimental Validation

In this section, we address two questions: (i) are the algorithms successful in finding meaningful groups, and (ii) are there interesting structures present in Pareto-optimal sets and approximations thereof. These aspects are studied on the basis of the bi-objective 0-1-knapsack problem [121].

Proof-of-principle Results on Well-structured Matrices

To show that both methods presented in the last section can find known structures in a given decision matrix Ξ, we implant random biclusters, each defining a particular module, into a matrix[2] and analyze the capability of the two biclustering algorithms to find the corresponding modules. In detail, biclusters that contain the same solutions are merged to constitute one bicluster beforehand. Each of these enlarged implanted biclusters corresponds to a module that contains all columns the bicluster contains. To check whether both Hartigan's algorithm (with sorting according to Hamming distance) and Bimax find these modules, we use the following measure. For each implanted module, we compute the module found by the biclustering algorithms that matches the implanted module best, i.e., that has the highest ratio of shared columns to the union of both column sets. The average of these best ratios over all implanted modules indicates the percentage of implanted modules that are covered by the automatically identified modules.

The results for different matrix sizes and different densities are shown in Table 3.1. Two major observations can be made. First, Bimax finds more of the implanted structure than Hartigan due to its exhaustive search for biclusters. The covering is not 100% for Bimax because it only finds inclusion maximal biclusters, that can be larger than the implanted biclusters. Second, the results for the sparse matrices are in all cases better than for the dense matrices. This can be explained by the high number of implanted

[2] To this end, random biclusters are generated until the desired number of ones is reached. The biclusters are then placed in the matrix randomly in order of their sizes—starting with the largest—with the restriction that biclusters cannot overlap.

matrix size	percentage of ones in matrices	percentage of covering	
		Hartigan	Bimax
50x50	20	69.72	99.11
50x50	50	61.11	82.74
100x100	20	75.22	93.73
100x100	50	49.97	71.72
300x300	20	51.65	75.92
300x300	50	35.56	n/a

Table 3.1 Percentage of modules found in structured random matrices that are covered by implanted modules. Note that the number of biclusters found by Bimax on the dense 300×300 matrix was already too large, i.e., its running time longer than one day.

biclusters in the dense matrices and the issue that even Bimax does not find all of these biclusters since the pruning heuristic of Sec. 3.3.2 was used.

Pareto-Optimal Sets Contain Structure

We would now like to show that Pareto-optimal sets actually contain structure. As a test case, the knapsack problem is chosen, as its Pareto-optimal set can be calculated exactly using an integer linear programming solver. If the hypothesis that a Pareto-optimal set actually contains structure holds, the corresponding decision matrix should contain larger and fewer modules than a random matrix; this, in turn, should be reflected in a smaller error as defined in Definition 3.7.

Here, we compare the Pareto-optimal sets of 11 different bi-objective knapsack instances including 100 items with 11 randomly generated matrices of similar size[3] with respect to the structure that is found by the two proposed methods based on Hartigan's algorithm (with sorting according to Hamming distance) and Bimax. The random matrices are generated by

[3] The size is chosen by calculating the average length and width of the knapsack Pareto-optimal sets. In this case, there are on average 150 solutions and 55 items that are neither contained in all nor in none of the solutions. Note that we are not interested in decision variables that are contained in all or no solution. Therefore, such columns are deleted prior to module finding.

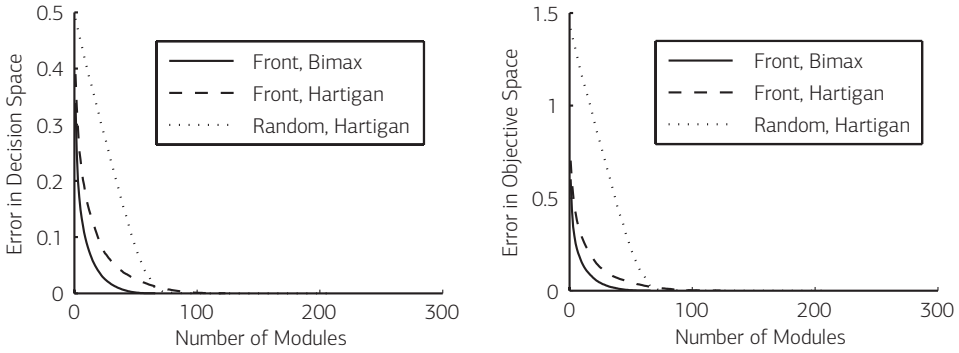


Figure 3.4 Comparison between Pareto-optimal sets and random matrices with respect to error function e_{dec} (left) and e_{obj} (right) averaged over 11 instances. The error is plotted against the number of modules taken into account if the modules are chosen as in the dendrogram, i.e., according to their size—starting with the largest.

setting every entry to 1 independently with probability 0.5; the solution's objective vectors are also randomly chosen by assigning randomly generated profits and weights to the 0-1-knapsack problem.

The results as depicted in Figure 3.4 indicate that the Pareto-optimal fronts contain more structure than the random matrices. In detail, both Bimax and Hartigan find modules that yield smaller errors for the Pareto-optimal fronts than for the random matrices if the same number of modules is taken into account. Note that although the objective space values are not taken into account by either method, the error in objective space is significantly smaller for Pareto-optimal fronts than for random matrices. Furthermore, we have to note that Bimax was not applicable on the random matrices since the number of biclusters found is too high. However, Bimax finds better modules in the structured Pareto-optimal sets yielding a lower error than those found by Hartigan's algorithm. An error of zero is already reached with about 50 modules which results in a reduction of the decision variables of about 50% in the corresponding module matrix.

Progress of Structure During Search

To study the change of the structure of the Pareto-optimal set approxima-

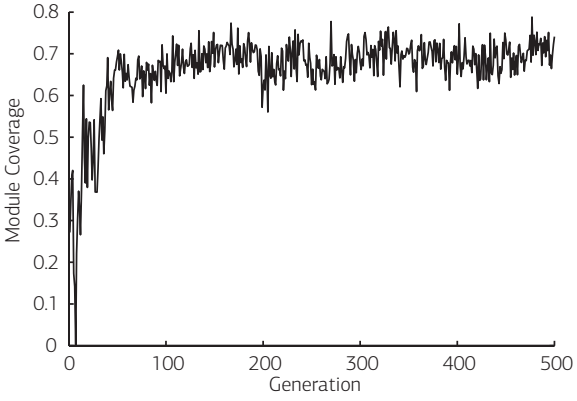


Figure 3.5 Average coverage of the modules found by Hartigan's algorithm on SPEA2's current non-dominated individuals plotted over time.

tion *during* the run of a multi-objective evolutionary algorithm, we apply the Modified Strength Pareto Evolutionary Algorithm (SPEA2) [122] to one of the 0-1-knapsack instances of the previous section[4]. This is the first step towards an automated detection of problem structure to speed up the search. However, it remains future work to study an online search space reduction in depth. Figure 3.5 shows the progress of similarity between the population's modules and the modules contained in the Pareto-optimal set over time. In this case, Hartigan's algorithm with sorting according to Hamming distance was applied both to the sets of non-dominated solutions in each generation and on the Pareto-optimal set itself to find the contained modules. The similarity of modules is defined as the deviation between the two column sets as described in Sec. 3.3.3.

Figure 3.5 shows the trend of the module coverage over time. As expected, the modules found in the population become more similar to the ones contained in the Pareto-optimal set as the population converges to the Pareto-optimal set, although the fluctuations of the similarity are large.

[4]For SPEA2, the implementation from the PISA toolbox with standard parameter values is used [8]. The population size is set to 300 and the knapsack instance has 100 items.

Running Times

The Bimax algorithm has a worst-case running time which is exponential in the matrix size. This is mainly due to the number of biclusters found, which limits the usage of Bimax. For example, the decision matrix of size 701×123 containing the solutions of a Pareto-optimal set from a 250 items knapsack instance produce more than 2 GB of data. One way to reduce this huge amount of data is to restrict the minimum bicluster size. However, this cannot solve the problem completely. Although structured matrices of size 300×300 can be processed, Bimax needs more than one day on an AMD 64bit Linux machine with 4 cores and 2.6GHz to process random matrices of the same size. The usage of Bimax is therefore limited to small instances. However, it served as a reference method that yields—due to its exhaustive search for biclusters—better results than Hartigan's algorithm.

For Hartigan's algorithm, a similar restriction on the minimum bicluster size can be used which makes the algorithm applicable to matrices of reasonable size, see Figure 3.6. For example, the computation of the modules within a Pareto-optimal set of a 250 item knapsack instance with 344 solutions takes about one minute on the AMD Linux machine mentioned above if the minimum bicluster size is set to 10% of the matrix dimensions.

3.3.4 · Results

In this section, we apply MANA to analyze two populations of a knapsack and a network processor design problem to show what can be gained from an analysis of the structure in Pareto-optimal set approximations.

Knapsack Problem

For the biobjective 0-1-knapsack problem, we focus on the grouping according to both similarity in decision space and objective space. This is not directly provided by the proposed approaches but can be gained indirectly by sorting the decision matrix within the Hartigan framework according to objective space and doing the grouping according to decision space. For sets of non-dominated solutions of a two-objective problem, the sorting of the decision matrix according to objective space values can be achieved without

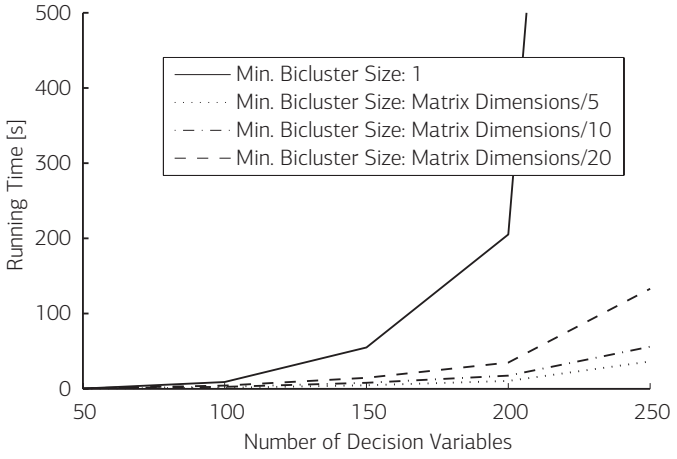


Figure 3.6 Running time of Hartigan's algorithm on Pareto-optimal sets of the 0-1-knapsack problem for different input sizes and different minimal bicluster sizes. Note that the number of solutions in the Pareto-optimal sets is 142 on average for the 100 item instances and 344 for the 250 item instances.

loss of generality by sorting according to the values of the first objective. Figure 3.7 shows a grouping example for a Pareto-set approximation of a 0-1-knapsack problem instance with 100 items, generated by SPEA2 as in the previous section, using the settings of [121].

For illustrating the similarities within the groups, we can additionally plot the profit-to-weight ratios of the items of the knapsack instance and indicate for each group which items are included in all solutions of the considered group (black), and included in no solution of the group (white). Figure 3.7 shows the profit-to-weight ratio plots for three exemplary groups of the investigated knapsack instance. For clarity, items that are contained in all or in no solutions of the entire Pareto-optimal set are not plotted. Interestingly, the analyzed Pareto-optimal set contains structure within both decision and objective space: solutions that are neighbored in objective space also show similarities in their decision vectors. Solutions located on the same extreme of the Pareto-optimal frontier have similar items selected whereas solutions on opposite extremes have complementary decision vectors; for solutions

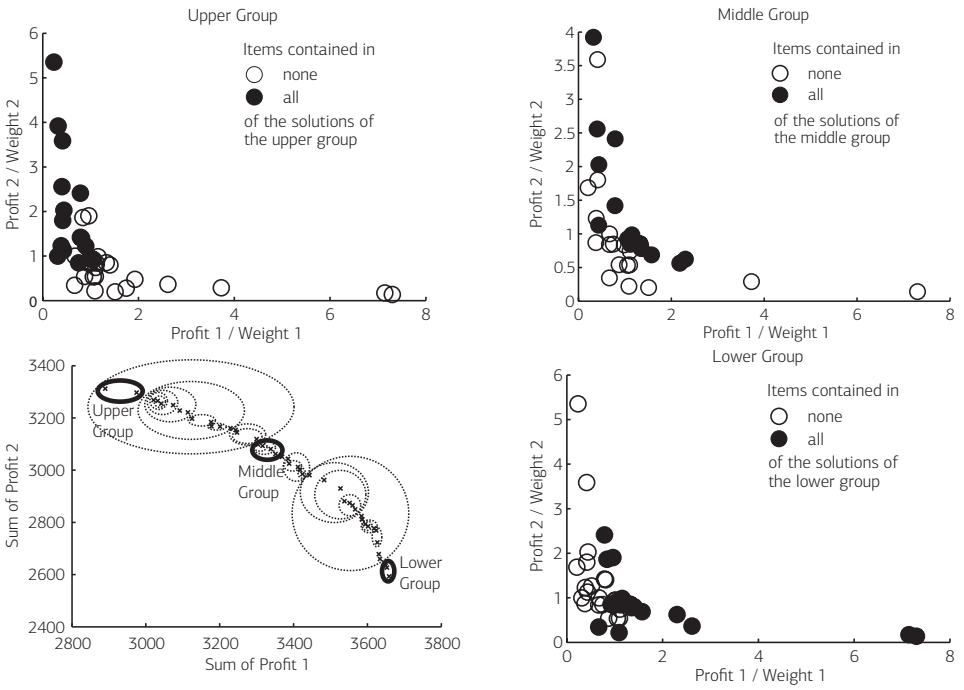


Figure 3.7 Grouping of a Pareto-optimal set approximation for the knapsack problem with 300 solutions and 11 groups (lower left) and item representation of three exemplary groups: the group with highest f_1 values is shown in the lower right figure, the group with the highest f_2 values is shown in the upper left figure, and the upper right figure shows a group with intermediate objective values. The grouping is done with modules of Hartigan's algorithm.

that have a high f_1 value, items with high f_1 profit are selected, whereas solutions with high f_2 values contain more items with high f_2 profit.

Network Processor Design

As a second application, we choose the problem of a network processor design as described in [103] and as provided in the PISA framework [8]. The problem is to optimize the architecture of packet processing devices with respect to the two objectives performance and cost. In more detail, components of the processor have to be chosen and computing tasks have to be assigned to these components afterwards. To investigate the underlying

structure of this problem, we use the multi-objective optimizer IBEA [120] to generate a Pareto-optimal set approximation. To this end, the algorithm is run with a population size of 150 for 300 generations. Only the 33 non-dominated solutions found are used in the analysis based on Hartigan's algorithm.

Figure 3.8 illustrates the original decision matrix ordered by objective space similarity together with the largest found biclusters and shows the resulting dendrogram. The modules found and the dendrogram help to gain a basic understanding of the problem, even when the decision maker cannot be sure about whether the known solutions are Pareto-optimal or not.

For our example instance, 143 out of all 233 decision variables are set to zero for all 33 solutions, which means that certain tasks are never mapped to certain components. Four of the remaining 90 decision variables are set to 1 in all 33 solutions. In this case, it says that in all 33 different processor designs, one particular component, namely a digital signal processor (DSP), is chosen and three of the 25 tasks are allocated to this component. This can assist in decision making in a way that these parts do not have to be taken into account by the decision maker because all known solutions have the same sub-structure. From the dendrogram, we can also extract some information about the problem. For example, in the case of three groups (horizontal cut between S_2 and S_3), one group contains module S_1 (left branch of the dendrogram) and the second one only module S_2 (middle branch). In the third group, indicated by the rightmost branch in the dendrogram of Figure 3.8, all solutions contain neither the module S_1 nor the module S_2. S_1 maps all remaining tasks to the DSP. S_2, on the other hand selects a cipher and assigns it two other tasks. Interestingly and similar to the observation for the knapsack problem, all solutions that contain a certain module, here S_1, occur on an extreme of the Pareto-optimal front: the solutions are cheap but slow.

3.3.5 · MANA Summary

When solving multi-objective optimization problems, three problems occur during decision making: (i) the solutions are represented by too many de-

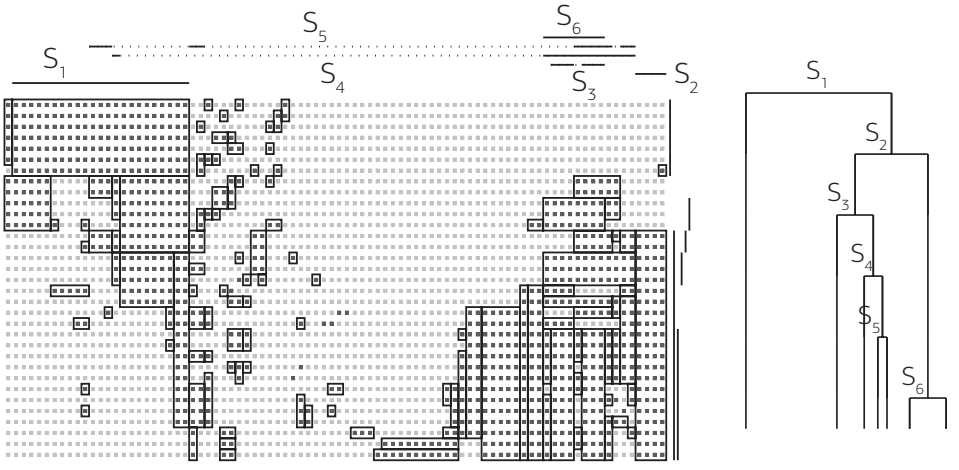


Figure 3.8 Visualization of structure in a Pareto-set approximation for the network processor design problem: (left) decision space values of the 33 non-dominated solutions found; (right) dendrogram.

cision variables, (ii) too many non-dominated solutions exist, and (iii) too many objectives are involved in the evaluation. The approach presented in this section tackled the first and second problem simultaneously. The first problem is solved by using two biclustering algorithms to automatically reduce the number of decision variables, by finding so-called modules of the decision space, i.e. subsets of decision variables that are as large as possible and are set to the same value in as many solutions as possible. The information encoded in a large number of decision variables can therefore be reduced to a smaller number of modules. The second problem is solved by using these modules to group similar solutions in a dendrogram, where the solutions are the leaves. Each node in the dendrogram is annotated with a module, and the solutions to the left of the node contain this module, whereas the solutions to the right do not contain the module. Each level of the dendrogram corresponds to a partitioning of the solutions into clusters, which groups solutions containing similar modules.

The proposed methods have been extensively tested. When running the biclustering algorithms on artificial binary matrices with implanted biclus-

ters, it was found that the exhaustive biclustering algorithm Bimax finds more of the implanted clusters than the other tested biclustering algorithm by Hartigan. The Hartigan algorithm, on the other hand, can be applied to larger matrices than Bimax. When comparing the biclusters found in the Pareto-set of a knapsack problem with random matrices, we found that the Pareto-set contains fewer but larger modules, indicating that Pareto-sets do contain structure. We also found on a knapsack problem that the modules of the population of an evolutionary algorithm become more similar to the modules contained in the Pareto-set during the search. Finally, when applying MANA to a knapsack problem and a network processor design problem, it was found that there is a relation between modules contained in solutions and the solution's objective values, and that these relations could be visualized using MANA, as solutions containing the same modules map to the same objective space region.

In the future, it may be promising to extend the proposed approach to non-binary decision spaces. Here, advanced biclustering techniques could be useful [68]. For a more general approach, modules with arbitrary decision variable values could be considered. Furthermore, one may think of using the reduction techniques online, i.e., *during* the search. The idea would be to reduce the decision space whenever significant modules have been found. Thereby, the search may be better focused towards promising regions.

3.4 · General Decision and Objective Spaces

This section presents the Pareto-Front Analyzer (PAN), another method that helps to analyze Pareto-sets or approximations thereof. It tackles the case that there are many solutions and that the solutions in general are difficult to interpret or visualize. It formulates the problem as a biobjective clustering problems where a partitioning is sought which yields compact and well separated clusters both in decision and in objective space. A clustering which is good both in decision space and in objective space elicits information from the front about what design types lead to what regions in

objective space. This section therefore proposes PAN, an evolutionary algorithm, which yields a set of tradeoff solutions to this biobjective partitioning problem. The novelty of the presented approach over existing work is its general nature, as it does not require the identification of distinct design variables or feature vectors. Instead, the proposed method only requires that a distance measure between a given pair of solutions can be calculated both in decision and in objective space.

3.4.1 · Problem Setting

We here follow the notation introduced in Section 1.1. Consider a multi-objective optimization problem with a decision space $\mathcal{X}$ and an objective space $\mathcal{Y} \subseteq \mathbb{R}^m = \{f(x) \,|\, x \in \mathcal{X}\}$, where $f : \mathcal{X} \to \mathcal{Y}$ denotes a mapping from the decision space to the objective space with m objective functions $f = \{f_1, ..., f_m\}$. An element $x \in \mathcal{X}$ of the decision space is also named a solution. While the objective space is a real-valued space, we make no assumptions about the structure of the decision space. In particular, we do not require the decision space to be an Euclidean space, and we also do not require the decision space to be spanned by a predefined set of decision variables that can take a certain set of values. Instead, we only assume that we are given a distance measure on solution pairs, i.e. $d_D : \mathcal{X}^2 \to \mathbb{R}$, where $d_D(x_1, x_2) \in \mathbb{R}$ denotes the structural distance between the two solutions x_1 and x_2.

We also need a distance measure in objective space. As the objective space is a real-valued metric space, we choose Euclidean distance, i.e. $d_O : \mathcal{X}^2 \to \mathbb{R}$, with $d_O(x_1, x_2) = \sqrt{\sum_{i=1}^{m}(f_i(x_1) - f_i(x_2))^2}$ denoting the Euclidean distance between x_1 and x_2 in objective space. Note that we here assume that the objective space is of a reasonable dimensionality. For high-dimensional real spaces, the Euclidean distance is not a good distance measure anymore. See [52] for more information and a rank-based solution to this problem.

Consider now that we are given a set of such solutions $\mathcal{X}^* \subset \mathcal{X}$. We do not make any assumptions about this set itself or about how this set has been generated, for example it can be the output of a multi-objective optimizer, and it can contain both dominated and non-dominated solutions. This set

may be time consuming to interpret, especially if there are many solutions, many objectives and if the solutions have a complex decision space representation. We therefore would like to generate a partitioning of this set, i.e. we would like to group the solutions into clusters to ease the interpretation of $\mathcal{X}^*$.

Definition 3.12 (cluster): *A* cluster $c \subseteq \mathcal{X}^*$ *is a subset of all solutions in the given set* $\mathcal{X}^*$.

Definition 3.13 (partitioning): *A* partitioning $C = \{c_1, \cdots, c_k\}$ *is a set of* k *clusters such that each solution is included in exactly one cluster, i.e.* $\forall x \in \mathcal{X}^* : (\exists c_i \in C : x \in c_i)$ *and no solutions is included in more than one cluster, i.e.* $x_i \in c_j \wedge x_i \in c_l \Rightarrow j = l$.

But what is a good partitioning? Usually, a good partitioning is one where the clusters are compact and well separated. This means that solutions within a cluster should be close to each other (i.e. the cluster has a small intra-cluster distance), and solutions of different clusters should be far from each other (i.e. the clusters have a large inter-cluster distance). In the literature [117], these two measures are usually combined into one goodness measure, the so-called validity index. We therefore assume that we are given such a validity index $V : (D, d) \to \mathbb{R}$. Here, D is an arbitrary space and d is a distance measure defined on D, i.e. $d : D^2 \to \mathbb{R}$. Note that many validity indices cannot cope with partitionings that contain only one cluster (which in turn contains all solutions). Also, they sometimes have problems with clusters that only contain one solution. We therefore assume that a feasible partitioning must contain at least 2 clusters, and each cluster must contain at least two solutions. This reduces the possible number of clusters k to the interval $k \in [2, \lfloor \frac{|\mathcal{X}^*|}{2} \rfloor]$.

In this section, we would like to find a partitioning which is good both in decision and in objective space. We therefore have two objectives, the first is the validity index in objective space and the second is the validity index in decision space. Whether these two indices are conflicting or not depends on the given solution set $\mathcal{X}^*$. There may well be solution sets where a good partitioning in objective space does not result in a good decision space

partitioning and vice versa, e.g. for optimization problems where radically different designs can lead to similar objective space values. Depending on the chosen distance measure, it can also happen that two solutions with a low distance to each other have quite dissimilar objective values, in particular if the distance measure does not capture all differences between the solutions.

As two conflicting objectives generally lead to a tradeoff front, we here suggest to optimize the two validity indices as a biobjective problem in order to find that front. This has the advantage that the two indices do not have to be combined into one goodness measure a priori. Furthermore, the tradeoff between a good partitioning in decision space and in objective space can be visualized, and the user can then choose one of the partitionings depending on which space is more important to him. The optimization problem can therefore be stated as follows:

Problem 3.14 (biobjective clustering): Find a partitioning C^* such that $V(f(C^*), d_O)$ and $V(C^*, d_D)$ are optimal. Here, $V(f(C^*), d_O)$ is the validity index calculated on the objective space values of the solutions in C^*, and $V(C^*, d_D)$ is the validity index calculated on the decision space values. d_O and d_D are the distance measures in objective and decision space, as defined in the first two paragraphs of this section.

3.4.2 · PAN Algorithm

Clustering problems in general are hard to solve. A simultaneous clustering in two spaces is even more challenging, and it is not clear how an algorithm should be designed to achieve good clusters, especially if several cluster validity indices are considered. We therefore propose PAN, an evolutionary algorithm, to optimize the biobjective problem defined in Section 3.4.1. The general framework of our evolutionary algorithm is shown in Algorithm 10. This is a standard form of an evolutionary algorithm, where variation and selection is iteratively applied for a fixed number of iterations. Note that in PAN, each solution corresponds to a partitioning. The population

```
1: function EA(n, g)
2:     Initialize population P randomly with n partitionings
3:     for g generations do
4:         P' = VARIATE(P,n)                          (generate n offspring)
5:         P = SELECT(P ∪ P', n)                      (select n partitionings)
6:     return P
```

Algorithm 10 General framework of an evolutionary algorithm. Input parameters: population size n; minimization is done for g generations.

```
1: function SELECT(P, n)
2:     while |P| > n do
3:                                   (remove partitioning with smallest contribution)
4:         P = P\{argmin_{p_i ∈ P}(I_H(P) − I_H(P\p_i))}
5:     return P
```

Algorithm 11 Selection Procedure. Input parameters: population $\mathcal{P}$, number of partitionings to select n. $I_H(\mathcal{P})$ is the hypervolume of population $\mathcal{P}$.

$\mathcal{P}$ therefore is a set of partitionings and the objective functions are the partitioning goodness measures in decision and in objective space.

For the selection procedure, we opted to go for the standard greedy hypervolume-based selection which is shown in Algorithm 11, where $I_H(\mathcal{P})$ is the hypervolume of population $\mathcal{P}$. The hypervolume in turn is calculated on the objective values of the solutions that are defined by the selected cluster validity index.

The variation procedure is shown in Algorithm 12. We assume that the number of offspring to generate is equal to the population size. Also, we are using random sampling without replacement as a mating selection scheme. Note that there are some constraints on the partitionings, namely that each partitioning must at least contain two clusters and that each cluster must at least contain two solutions. The functions ISVALID and ISINVALID check whether a given partitioning is valid or invalid. We deal with these constraints by using a repeat strategy, i.e. for each parent pair selected during mating selection, we keep generating offspring until two feasible offspring

```
1: function VARIATE(P, n, p_R)
2:     for 1 to n/2 do
3:         set o_1 and o_2 to an invalid partitioning
4:         while ISINVALID(o1) or ISINVALID(o2) do
5:                                          (randomly select two parents from P)
6:             {p_1, p_2} =MATINGSELECTION(P)
7:             o'_1 = p_1, o'_2 = p_2                    (set offspring to parents)
8:             With probability p_R: {o'_1, o'_2} =RECOMBINE(p_1, p_2)
9:             o'_1 =MUTATE(o'_1)
10:            o'_2 =MUTATE(o'_2)
11:            if ISINVALID(o_1) and ISVALID(o'_1) then
12:                o_1 = o'_1
13:            if ISINVALID(o_2) and ISVALID(o'_2) then
14:                o_2 = o'_2
15:        P = P ∪ {o_1, o_2}
16:    return P'
```

Algorithm 12 Variation procedure. Input parameters: population $\mathcal{P}$, (even) number of offspring n; recombination probability p_R.

have been found. The recombination and mutation depends on the selected representation and is described in more detail in Section 3.4.2.

Speed Up by Local Heuristic

Preliminary tests (see also Section 3.4.3) showed that without any speedup, PAN with an arbitrary representation and validity index takes a long time to reach satisfying partitionings. In order to speed up the search we therefore propose to integrate a local heuristic into the search. One of the most common clustering algorithms is the k-means algorithm [67]. The k-means algorithm is known to converge quickly towards the nearest local optimum, which makes it well suitable as a local heuristic during optimization.

To integrate the local heuristic, we propose to locally optimize the offspring partitionings both in decision and in objective space, and then select the future parents from the set containing both original offspring, offspring locally optimized for partitioning goodness in objective space, and offspring locally optimized for partitioning goodness in decision space. The adapted

1: **function** PAN(n, g)
2: Initialize population $\mathcal{P}$ randomly with n partitionings
3: **for** g generations **do**
4: $\mathcal{P}' = \text{VARIATE}(\mathcal{P},n)$ (*generate n offspring)*
5: (*apply local optimization in both spaces)*
6: $\mathcal{P}'_o = \text{LOCALOPT}(\mathcal{P}', \text{obj})$
7: $\mathcal{P}'_d = \text{LOCALOPT}(\mathcal{P}', \text{dec})$
8: $\mathcal{P} = \text{SELECT}(\mathcal{P} \cup \mathcal{P}' \cup \mathcal{P}'_o \cup \mathcal{P}'_d, n)$ (*select n solutions)*
9: **return** $\mathcal{P}$

Algorithm 13 PAN algorithm with local search. Input parameters: population size n; minimization is done for g generations.

general framework of PAN that incorporates the local search is shown in Algorithm 13.

Note that the original k-means algorithm makes use of the cluster centroids, which assumes that the solutions are given in Euclidean space. As we only require pairwise distances in decision space, we therefore use the k-medoids [62] algorithm instead, which is an adapted version of k-means that works with cluster medoids instead of centroids, see also Section 3.4.2 for more details about cluster medoids. The adapted k-means algorithm is shown in Algorithm 14.

Representation

When designing an evolutionary algorithm, a suitable representation has to be chosen for the problem at hand in order to code the different solutions, in this case partitionings. In the literature, several representations are used for clustering problems, namely the centroid representation, the graph representation, the integer representation and the direct representation, see e.g. [54].

Centroid representation The centroid representation is used by several authors [20, 63] and codes only the cluster centroids. Each solution is assigned to the nearest centroid. This is similar to the cluster allocation of the well-known k-means clustering algorithm [67]. Two versions of this representa-

```
1: function LOCALOPT(P, d)
2:     for p ∈ P do
3:         ∀c_i ∈ p : m_old(c_i) = ∅                                  (initialize medoids)
4:         while forever do
5:                                                                   (calculate cluster medoids)
6:             ∀c_i ∈ p : m(c_i) = argmin_{x_j ∈ c_i} Σ_{x_k ∈ c_i} d(x_j, x_k)
7:                                                                   (reassign partitionings to nearest medoid)
8:             ∀c_i ∈ p : c_i = {x ∈ X* | ∄c_j, c_j ≠ c_i s.t. d(x, m(c_j)) < d(x, m(c_i))}
9:             if ∀c_i ∈ p : m(c_i) == m_old(c_i) then
10:                break
11:            ∀c_i ∈ p : m_old(c_i) = m(c_i)
12:    return P
```

Algorithm 14 Local Heuristic: adapted k-means. Input parameters: population of partitionings $\mathcal{P}$; pairwise distance measure d in the space where the partitionings have to be optimized. $\mathcal{X}^*$ contains the solutions to be partitioned.

tion exist, the first one has a fixed (maximum) number of cluster centroids, and the representation also contains a bitstring that says for each cluster centroid whether it is activated or not. The number of activated centroids then is the number of actually considered clusters. The second version has a variable-length representation, where centroids can be added and removed from a list of centroids. The centroid representation has the advantage that it considerably reduces the search space, as only a small number of centroids has to be chosen. The disadvantage is that it is especially useful for spherical clusters, but can lead to wrong partitionings on more general cluster shapes. Also, the centroid calculation assumes that the solutions are given in Euclidean space. While this problem could be solved by using a different definition of centroids, we still have the problem that the centroids are defined in the same space where the solutions are defined. In our case, the solutions are defined in two spaces, and it is not at all clear how one centroid can be decoded into two spaces. We therefore need a representation that directly represents the solutions assignment to clusters, without making any assumptions about the used spaces.

Graph Representation Park and Song [79] suggest the graph representation, which is an adjacency list of length n where n is the number of solutions. The i-th value in the list codes one link that says to which other solution the i-th solution is connected to. The connections of the whole adjacency list return a graph, where the clusters are the unconnected subgraphs. Handl and Knowles [45] did extensive tests with this representation, and found that it works satisfactorily. The advantage of this representation is that standard variation operators can be applied. Here, we follow Handl and Knowles and use uniform crossover with switching probability of 0.5 for each element to do recombination and randomly change one element in each mutation. Note that Handl and Knowles proposed to reduce the search space by allowing each solution to be only connected to its L nearest neighbors. Also, Handl and Knowles state that links to further away individuals are less favorable than links to close neighbors and should therefore be mutated with a higher probability [44]. To keep the comparison between different representations fair, we do not make use of these techniques.

When using the graph representation, applying the local heuristic using k-medoids is not straightforward as the locally optimized partitioning p_i^{opt} might look quite different from the original partitioning p_i. If so, it is not clear how to incorporate these changes into the original graph structure, while keeping as many common links as possible. We here use the following approach: First, starting from the original partitioning p_i, all links between solutions that are not in the same cluster in the optimized partitioning p_i^{opt} are removed. Then, for each remaining cluster in p_i an unweighted minimum spanning tree is calculated and all links not present in the minimum spanning tree are removed. Then, all links that have been removed in the previous two steps are reinserted in a random manner, and it is checked whether the new partitioning p_i' corresponds to the optimized one p_i^{opt}. If not, another random assignment is selected. If no assignment is found which produces the optimized partitioning, the locally optimized partitioning is discarded.

Integer Representation Another representation we consider in this section is called the integer representation. It is coded by an integer string $x \in$

$\{1, 2, ..., \lfloor \frac{n}{2} \rfloor\}^n$ of length n, where n is the number of solutions. Solutions with the same integer value are assigned to the same cluster. As a mutation operator, we use single-point mutation, where one randomly chosen position in the string is assigned a randomly chosen new integer value already present in the string, i.e. $v_{new} \in \{x\}$. As a recombination operator we use uniform crossover, where for each position in both parent strings, the two integers are exchanged with probability 0.5.

Direct Representation We also suggest to use a direct representation, inspired by the work of Falkenauer [34]. The direct representation stores a list of clusters or variable length, where each cluster in turn is a list of solutions. We then define three mutation operators for this representation:

- Move Operator: Moves a randomly selected solution to a randomly selected other cluster, with probability p_m
- Merge Operator: Merges two randomly selected clusters, with probability p_u
- Split Operator: Splits a randomly selected cluster into two random parts, with probability p_s

As a recombination operator we suggest to use an operator proposed by Falkenauer [34]. It resembles a two-point crossover in the following way: given two parents p_1 and p_2, from which we want to create two offspring o_1 and o_2. First, we set $o_1 = p_2$ and $o_2 = p_2$. Then, we choose two random cut points in the cluster list of both parents. The clusters between the two cut points of p_1 are added to o_2 in the position after the first cut point in p_2. Now there are several original clusters in o_2 that contain the same solutions as the clusters added from p_1. Therefore, these duplicate solutions are removed from their clusters. The second offspring is generated in the same way, with the roles of the parents reversed. An example is shown in Figure 3.9.

The direct representation has the advantage that the impact of the variation operators on the partitioning is obvious and known in advance. This for example contrasts with the graph representation, where the redirection of one edge can lead to a move, merge or split, depending on the remaining

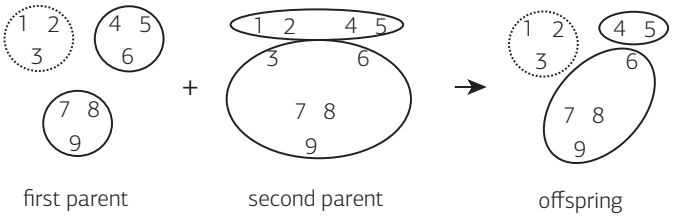


Figure 3.9 Example of how an offspring is generated using recombination in the direct representation. The data set contains 9 points, and the clusters of the parents as well as of the offspring are shown. The dotted cluster of the first parent is the cluster which is implanted into the second parent to generate the offspring.

graph edges. Note that the direct representation together with its operators is the only representation which is context sensitive according to [34], i.e. which variates the solutions taking into account the specific cluster structure at hand.

Validity Indices

As stated in Section 3.4.1, we would like to find partitionings that have a good cluster validity index both in objective and in decision space. In the literature a multitude of different validity indices can be found, see e.g. [5, 42, 54, 117]. They combine the two clustering goals, i.e. cluster compactness and cluster separation, into one goodness measure. Usually, these validity indices are used to find the correct number of clusters to a given clustering problem. To do so, clustering optimizers that take the number of clusters k as a parameter (e.g. the well-known k-means algorithm) are run for different values of k, and the resulting partitioning which achieves the highest cluster validity index is chosen to be the correct one. Optimizing such a validity index therefore leads to a partitioning with the correct number of clusters, see also [5, 42, 70] for overviews over indices that are used to identify the correct number of clusters.

Many of the validity indices found in the literature assume that the points to be clustered are given in Euclidean space. Most of the time, they assume that a cluster centroid can be calculated, where in each dimension the centroids value is the mean value of all solutions in the cluster and in

the respective dimension. Examples for such indices are the Davies-Bouldin index [21], the CS index [18], some variants of the Dunn index [7], the SD index [41], the $I(k)$ index [5] and the adapted silhouette index [53]. As we only assume that we are given pairwise distances, but without any information about the underlying decision variables, the cluster centroids cannot be calculated. To solve that problem, we here propose to use the medoids instead of the centroids. The medoid of a cluster is the solution with the smallest average distance to all other solutions in the cluster, see [61]. Note that while the calculation of the centroid is linear in the number of solutions, the calculation of the medoid is quadratic. See [43] for a sampling approach that faces this issue and speeds up the medoid calculation.

Also, there are some validity indices that do not only use centroids, but use the notion of a direction in the solution space, e.g. the S_Dbw index [40] or the ReD index [59]. Such indices cannot be used for our problem.

In the following, we shortly describe each of the validity indices we selected for our problem. We assume that we are given a set of n points $\mathcal{X}^* = \{x_1, ..., x_n\}$, with a distance measure $d(x_i, x_j) \in \mathbb{R}$. We now want to evaluate a given partitioning $C = \{c_1, ..., c_k\}$, with $c_i \subset \{x_1, ..., x_n\}$, where the definition of a cluster and of a partitioning corresponds to Definition 3.12 and 3.13, respectively. The medoid of a cluster c_i calculated as $m(c_i) = \text{argmin}_{x_j \in c_i} \sum_{x_k \in c_i} d(x_j, x_k)$. For reasons of comparison, we adapt the indices where necessary, such that each index has to be minimized.

Silhouettes Index The silhouettes index [89] is defined as follows:

$$S_{orig}(C) = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i) = \frac{1}{|c_p|-1} \sum_{x_j \in c_p} d(x_i, x_j)$, with c_p such that $x_i \in c_p$ (i.e. c_p is the cluster containing x_i). $a(i)$ therefore denotes the average distance of solution x_i to all other solutions in the same cluster. Also, $b(i)$ denotes the minimum distance of x_i to any other solution which is in a different cluster

than x_i, i.e. $b(i) = \min_{c_l \neq c_p} d(x_i, c_l)$. Here $d(x_i, c_l) = \min_{x_r \in c_l} d(x_i, x_r)$ is the minimum distance of x_i to any solution in c_l.

The silhouettes index can obtain values in the interval $S \in [-1, 1]$, where a value of 1 denotes a good partitioning and -1 denotes a bad partitioning. In order to transform this problem into a minimization problem with an optimal value of 0, we propose to use the following formula:

$$S(C) = -(S_{orig}(C) - 1), \quad , S(C) \in [0, 2]$$

Adapted Silhouettes Index The adapted silhouettes index [53] is defined as follows:

$$AS_{orig}(C) = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i) = d(x_i, m(c_p))$, with c_p such that $x_i \in c_p$ (i.e. c_p is the cluster containing x_i). $a(i)$ therefore denotes the distance of solution x_i to the medoid of the cluster containing x_i. Also, $b(i)$ denotes the minimum distance of x_i to any medoid of a cluster not containing x_i, i.e. $b(i) = \min_{c_l \neq c_p} d(x_i, m(c_l))$.

The same value adaptation as for the standard silhouettes index is used:

$$AS(C) = -(AS_{orig}(C) - 1), \quad AS(C) \in [0, 2]$$

Dunn Index The Dunn index [30] is defined as:

$$D_{orig}(C) = \frac{\min_{c_i, c_j \in C, c_i \neq c_j, x_p \in c_i, x_l \in c_j} [d(x_p, x_l)]}{\max_{c_t \in C, x_p, x_l \in c_t} [d(x_p, x_l)]}$$

i.e. the Dunn index divides the distance of the closest two points between any cluster by the largest spread of any cluster. This is a maximization problem, that lies in an interval of $D(C) \in [0, \infty]$. In order to transform it into a minimization problem, we propose the following adaption:

$$D(C) = -D_{orig}(C), \quad D_{orig}(C) \in [-\infty, 0]$$

Generalized Dunn Index The generalized Dunn index [7] is defined as:

$$GD_{orig}(C) = \frac{\min_{c_i, c_j \in C, c_i \neq c_j} \delta(c_i, c_j)}{\max_{c_i \in C} \Delta(c_i)}$$

As the original Dunn index was found to be strongly influenced by outliers, [7] suggested several new definitions of the inter- and intra-cluster distances. Two well performing definitions that only rely on pairwise distances are the following:

$$\delta(c_i, c_j) = \max\{\max_{x_l \in c_i} \min_{x_p \in c_j} d(x_l, x_p), \max_{x_l \in c_j} \min_{x_p \in c_i} d(x_l, x_p)\}$$

$$\Delta(c_i) = \frac{1}{|c_i| \cdot (|c_i| - 1)} \sum_{x_l, x_p \in c_i, x_l \neq x_p} d(x_l, x_p)$$

Again, this is a maximization problem, such that we need an adapted version:

$$GD(C) = -GD_{orig}(C), \quad GD(C) \in [-\infty, 0]$$

VRC Index The VRC Index [16] is defined as follows (BGSS/WGSS = between/within group sum of squares):

$$VRC_{orig}(C) = \frac{BGSS}{k-1} / \frac{WGSS}{n-k} = \frac{\bar{d^2} + \frac{n-k}{k-1} A_k}{\bar{d^2} - A_k}$$

where $\bar{d^2} = \sum_{i \in [1,n]} \sum_{j \in [1,n], j \neq i} [d(x_i, x_j)]^2 \frac{2}{n \cdot (n-1)}$ is the average squared pairwise distance, $A_k = \frac{1}{n-k} \sum_{c_i \in C} (|c_i| - 1)(\bar{d^2} - \bar{d_i^2})$ is a weighted mean

of differences between the general and the within-group mean squared distances, and $\bar{d}_i^2 = \sum_{x_l, x_p \in c_i, x_l \neq x_p} d(x_l, x_p)^2 \frac{2}{|c_i| \cdot (|c_i|-1)}$ is the average pairwise distance in cluster c_i.

This index has to be maximized, with values in the interval $[-\infty, \infty]$. We therefore suggest an adapted version that has to be minimized:

$$VRC(C) = -VRC_{orig}(C), \quad VRC(C) \in [-\infty, \infty]$$

Davies Bouldin Index The DB Index [21] is defined as follows:

$$DB(C) = \frac{1}{n} \sum_{c_i \in C} \max_{c_j \in C, c_i \neq c_j} \frac{\sigma(c_i) + \sigma(c_j)}{d(m(c_i), m(c_j))}$$

Where the dispersion $\sigma(c_i)$ of a cluster c_i is defined as $\sigma(c_i) = \frac{1}{|c_i|} \sum_{x_j \in c_i} d(x_j, m(c_i))$. This measure has to be minimized by design, with $DB(C) \in [0, \infty]$.

CS Index The CS Index [18] is defined as follows:

$$CS(C) = \frac{\sum_{c_i \in C} \left\{ \frac{1}{|c_i|} \sum_{x_j \in c_i} \max_{x_k \in c_i} d(x_j, x_k) \right\}}{\sum_{c_i \in C} \left\{ \min_{c_j \in C, c_i \neq c_j} d(m(c_i), m(c_j)) \right\}}$$

This measure is minimized by design, with $CS(C) \in [0, \infty]$.

I Index The I Index [5] is defined as follows:

$$I_{orig}(C) = \left(\frac{1}{k} \cdot \frac{E_1}{E_k} \cdot D_k \right)^2$$

where k is the number of clusters, $E_k = \sum_{c_i \in C} \sum_{x_j \in c_i} d(x_j, m(c_i))$ is the sum of the distances of all solutions to their respective cluster medoid, $E_1 = \sum_{x_i \in \mathcal{X}^*} d(x_i, m(X))$ is the sum of the distances of all solutions to the medoid of the whole population $\mathcal{X}^*$, and $D_k = \max_{c_i, c_j \in C} d(m(c_i), m(c_j))$ is

the maximum medoid distance. This measure has to be maximized, so we use an adapted version:

$$I(C) = -I_{orig}(C), \quad I(C) \in [-\infty, 0]$$

SD Index The SD Index [41] is defined as follows:

$$SD(C) = a \cdot Scat(C) + Dis(C)$$

$$Scat(C) = \frac{1}{k} \sum_{c_i \in C} \frac{\sigma(c_i)}{\sigma(\mathcal{X}^*)}$$

$$Dis(C) = \frac{D_{max}}{D_{min}} \sum_{c_i \in C} \left(\sum_{c_j \in C} d(m(c_i), m(c_j)) \right)^{-1}$$

where the variance $\sigma(c_i)$ of a cluster c_i is defined in the same way as for the Davies Bouldin Index, $D_{max} = \max_{c_i, c_j \in C} d(m(c_i), m(c_j))$ and $D_{min} = \min_{c_i, c_j \in C} d(m(c_i), m(c_j))$ are the maximum and minimum distance between any two cluster medoids, respectively, and $a = Dis(C_{max})$ is the dispersion of the partitioning with the maximum number of input clusters, i.e. in C_{max}, each solution is in its own cluster, or to put it differently, each cluster contains exactly one solution.

This measure has to be minimized by definition, with $SD(C) \in [0, \infty]$.

Practical Considerations

The PAN algorithm only makes a few assumptions about the dataset at hand. The first one is that the best partitioning in objective space is different from the best partitioning in decision space. If the two clustering goals are not conflicting, there is no set of tradeoff partitionings, but a single best partitioning. In this case, the final PAN population will contain a partitioning which dominates all others.

The second assumption is that each cluster contains at least two solutions, because some validity indices cannot handle clusters with only one solution.

We therefore suggest to do a data cleaning step where outliers, i.e. solutions that have a large distance to all other solutions, both in objective and in decision space, are identified by hand, and removed prior to clustering.

3.4.3 · Selection of Validity Index and Representation

Clustering problems in general are hard to solve and the search space is huge, even for a reasonable number of points to be clustered. If the optimization should work satisfactorily, the representation and partitioning goodness measure have to be selected carefully. This is due to the fact that some indices might introduce plateaus, or many local optima. In this section we try to find a combination of validity index and representation that performs satisfactorily on several clustering problems.

Usually, to test which validity index / representation combination works best on a standard clustering problem, the different combinations are tested on datasets where the optimal partitionings are known. Then, the combination whose result is closest to the known partitioning is chosen as the best one. In this section, however, the dataset is given in two spaces, namely the objective and the decision space, and a good partitioning should be good in both spaces. Assuming that the best partitioning in one space is not equal to the best partitioning in the second space, we are given the choice between different tradeoff partitionings. Unfortunately it is not clear what qualifies as a good tradeoff partitioning. Moreover, a tradeoff which is good with respect to one validity index can be poor with respect to another validity index. However, we know that all combinations should be able to find those two partitionings that are best in either the first or the second space, because these two partitionings are Pareto-optimal, independently of the chosen validity index (assuming that the validity index is actually best for the known optimal partitioning).

Therefore, we test our combinations by constructing different clustering problems where we know the optimal partitionings in both spaces in advance, and see whether the combinations can find the two extremal partitionings (those best in one of the two spaces) in the same run. We selected three testcases. The first testcase is the simplest where both spaces to

be clustered contain four clearly distinguishable clusters with five solutions each. The second testcase has clusters with different numbers of solutions to test PANs capability to recognize differently sized clusters. Finally, the third testcase has a larger set of solutions to be clustered in order to test PANs capability to achieve good partitionings even for a large number of solutions.

We use the same experimental setup for all three testcases, i.e. we use a population size of 10 for 500 generations, Euclidean distance as a distance measure (where we normalize all pairwise distances to lie in the interval $[0, 1]$). For each setting, we do 30 runs, with recombination probability $p_R = 0.7$ and $p_m = 0.6, p_u = 0.2, p_s = 0.2$ for the mutation operator of the direct representation.

To compare the results, we consider two aspects. First we check whether the partitionings that PAN finds to be optimal either in decision or objective space correspond to the expected optimal partitionings. And second, we measure the minimum number of function evaluations that is needed to find the optimal partitionings in both spaces.

1st Testcase: Proof of Concept and Validation of Local Heuristic

The simplest testcase is shown in Figure 3.10, which shows the optimal partitioning in both spaces. In both spaces, the best partitioning consists of 4 clusters of 5 solutions each. However, these best partitionings do not correspond to each other, as can be seen in the figure, where the upper two plots show the best partitioning in the first space (and the corresponding partitioning in the second space), whereas the lower two plots show the best partitioning in the second space (and the corresponding partitioning in the first space). For reasons of simplicity, we used the same location of points in both spaces. Note that if using the Pareto-optimal set of an optimization problem, the first space might be the objective space and the second space might be the decision space. We applied PAN with and without the local heuristic to get a feeling about the speedup when adding the local heuristic.

In both cases, we found that the SD Index found suboptimal partitionings that have a better validity index value than the known optimal ones. When

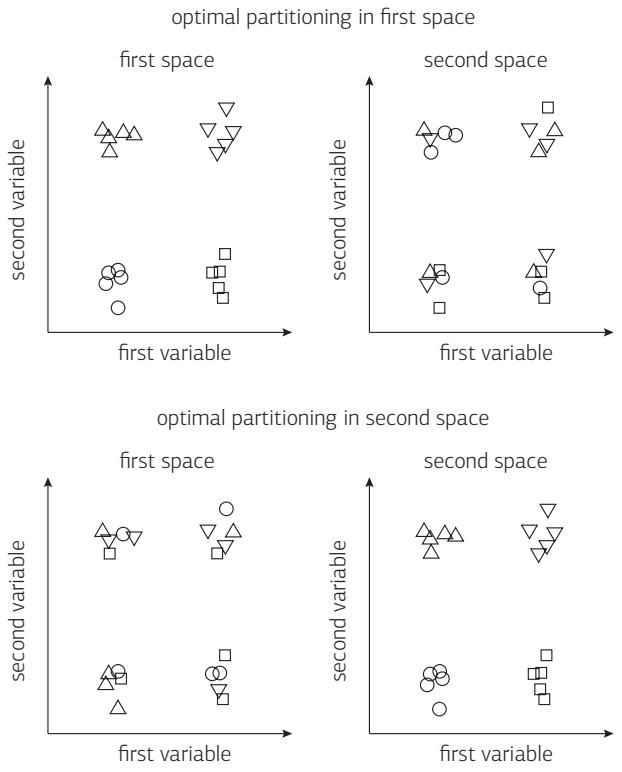


Figure 3.10 Points to be clustered for first testcase. The upper two plots show the optimal partitioning in the first space, the lower two plots show the optimal partitioning in the second space. Both pairs of plots show the points in the first/second space in their left/right plot.

inspecting these partitionings it can be seen that the reason for this behavior is the use of the medoid instead of the centroid. If one cluster contains solutions from all four optimal clusters, the centroid lies in the center of the solutions, whereas the medoid has to be one of the actual solutions and therefore is far from the centroid, which in turn causes problems when calculating the SD Index. The SD index therefore cannot be used for the optimization.

For the remaining indices, the number of function evaluations after which both optima have been found without the local heuristic is shown in the

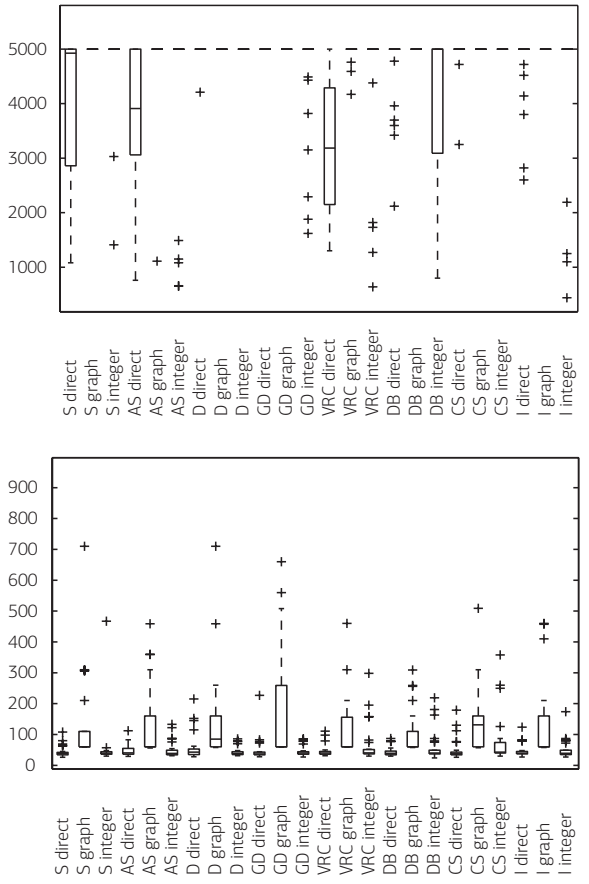


Figure 3.11 Number of function evaluations (smaller is better) after which both optima have been found for different validity index / representation pairs, without local heuristic (upper plot) and with local heuristic (lower plot).

left plot of Figure 3.11. It can be seen that no combination reaches both optima within 5000 function evaluations in all runs, which is an indication for a low convergence speed. Nevertheless, the VRC, S and AS Index with direct representation as well as the DB Index with integer representation seem to work better than the remaining combinations.

To tackle the slow convergence speed we now add the local heuristic. The number of function evaluations for reaching the known optima is shown in the right plot of Figure 3.11. As can be seen from the figure, all combinations (with the exception of the GD and DB index with integer representation that did not reach both optima in 5000 function evaluations in one out of all 30 runs) reach both optima within 800 function evaluations. Moreover, it can be seen that the direct and the integer representation find the optima faster than the graph representation.

2nd Testcase: Irregular Clusters

In the last section the known optimal clusters were all of the same size and both spaces had the same optimal number of clusters (i.e. four). The goal of this section is to see (a) how PAN performs if the best clusters are of different sizes and (b) whether PAN struggles with cases where the optimal number of clusters is quite different in the two spaces. The corresponding problem is shown in Figure 3.12. Note that in the first space there are 3 clusters with 2, 5, and 13 solutions each, whereas in the second space there is a more regular structure with 8 clusters of 2 solutions each, and one cluster with 4 solutions.

When looking at the results it has been found that the VRC Index, the I Index as well as the SD Index all find suboptimal partitionings that have a better value than the known optimal ones and therefore cannot be used for the optimization. For the remaining validity indices, the number of function evaluations after which both optima have been found is shown in the left plot of Figure 3.13. It can be seen that the direct representation is faster for the S, D and AS index, and not worse in the other indices than the graph and the integer representation.

3rd Testcase: Larger Dataset

Considering the results from the previous two testcases it was found that the direct representation works better than the other two representations. Also, the VRC, I and SD Index cannot be used, because their optimal partitionings are known to be suboptimal. The remaining indices seem to perform satisfactorily, so we tested these indices with direct representation

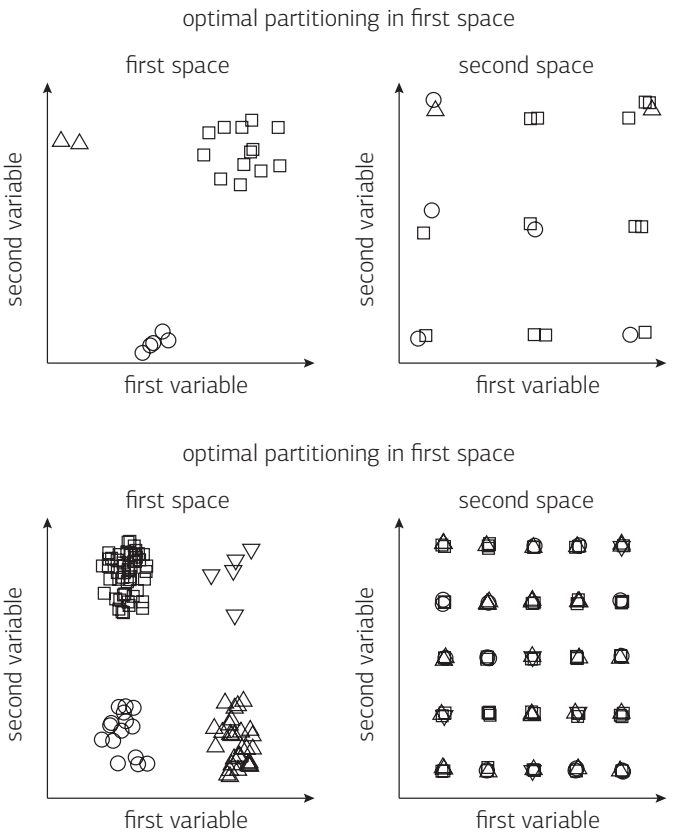


Figure 3.12 Points to be clustered for second (upper two plots) and third (lower two plots) testcase. Both pairs of plots show the optimal partitioning in the first space (left plots), with the corresponding partitioning in the second space in the right plot of the respective plot pair.

on a larger dataset. The dataset under consideration is shown in Figure 3.12. In the first space, there are four distinct clusters with 5, 15, 30 and 50 solutions each. In the second space, there are 25 evenly spaced clusters with 4 solutions each.

The results are shown in the right plot of Figure 3.13. It can be seen that PAN both with the D and the GD index do not find both optima within 5000 function evaluations. Also, the S index and the CS index perform

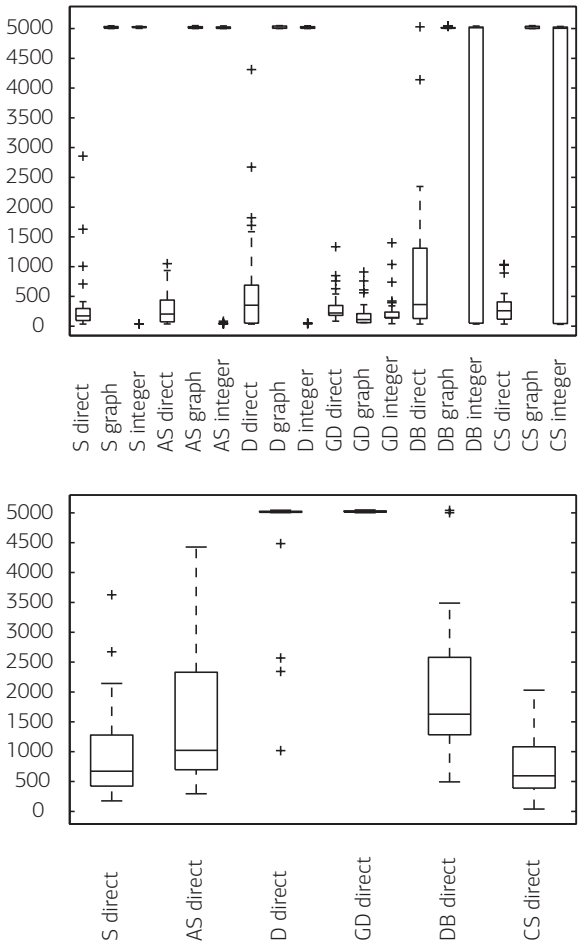


Figure 3.13 Number of function evaluations (smaller is better) after which both optima have been found for all representation / validity index pairs, with local heuristic.

slightly better than the AS index and the DB index. Finally, it has been found that the S index takes much longer to compute than the remaining indices, as it needs to calculate the pairwise distances between all solutions.

Testcase Summary

In this section we considered several artificial datasets in order to find a

good configuration for PAN. First, it could be observed that the speedup proposed in Section 3.4.2 decreases the required number of function evaluations to reach the known optimal partitionings significantly. Second, it was found that the SD, the VRC and the I index sometimes find suboptimal partitionings that have a better partitioning goodness value than the known optimal partitionings. Also, the D and the GD index do not reach the optimal partitioning in a reasonable number of function evaluations for larger datasets. And finally, the direct representation has a better performance than the graph and the integer representation, especially on larger datasets. In conclusion, it was found that a good combination for PAN is to use direct representation with either the S, the AS, the DB or the CS index.

3.4.4 · Results

This section first compares the proposed algorithm with the standard approach of iteratively applying the k-medoids algorithm. Then, the method is applied first to a knapsack problem and then to a real-world bridge construction problem and its results are qualitatively inspected.

Comparison with k-medoids

The goal of this section is to validate the multi-objective approach. To this end we compare the achieved hypervolume with the hypervolume obtained by the standard method of applying k-medoids iteratively. In more detail, this is done in the following way: We apply k-medoids several times for all possible cluster numbers. Each time we cluster the solutions twice, once in decision space and once in objective space and check whether the optimized partitionings satisfy the constraints (i.e. they contain at least two clusters, where each cluster must at least contain 2 solutions). For all partitionings that satisfy the constraints, we calculate the cluster goodness values in decision and in objective space. Finally, to compare the resulting population with PAN, we reduce the number of achieved solutions to the population size used with PAN, using PANs selection procedure.

We compared k-medoids with PAN on the 3rd testcase, where PAN uses direct representation and the same indices tested in Section 3.4.3. PANs

	PAN	k-medoids
S index	0.68 ± 0.27	0.36 ± 0.19
AS index	0.80 ± 0.23	0.22 ± 0.13
D index	0.20 ± 0.32	0.10 ± 0.02
GD index	0.52 ± 0.29	0.12 ± 0.04
DB index	0.63 ± 0.25	0.29 ± 0.11
CS index	0.75 ± 0.18	0.34 ± 0.17

Table 3.2 Mean and standard variation of achieved hypervolume values of PAN and the iterated k-medoids algorithm for six indices. For each index, all achieved hypervolume values were normalized such that the minimum/maximum hypervolume got the values 0/1.

population size again is 10 and it is run for 5000 function evaluations. The iterative k-medoids algorithm, on the other hand, is applied for all cluster numbers in the interval $[2, 50]$, with $\lceil 5000/49 \rceil = 103$ restarts for each cluster number. This way, both PAN and the iterative k-medoids algorithm use the same number of calls to the actual k-medoid algorithm.

The corresponding hypervolume values for the different validity indices are shown in Table 3.2 and Figure 3.14. According to a Kruskal-Wallis test performed on the data as described in [19], with the Conover-Inman procedure, Fisher's least significant difference method performed on ranks and a significance level of 1%, PAN is always significantly better than the k-medoids algorithm, except for the D index that has many outliers. This indicates that some partitionings found by PAN cannot be achieved by using k-medoids. Instead, slight variations of partitionings produced by k-medoids might have a high gain in one space, but at the same time not much loss in the other space.

Application to Knapsack Problem

First, we applied PAN to a simple biobjective knapsack problem. Here, we consider a problem with 150 items where each item i has two randomly chosen profits p_i^1 and p_i^2 and weights w_i^1, w_i^2, where p_i^1, p_i^2, w_i^1, and w_i^2 are chosen uniformly and at random in the interval $[10, 100]$. The problem can therefore be viewed as a selection problem, where a subset of the 150

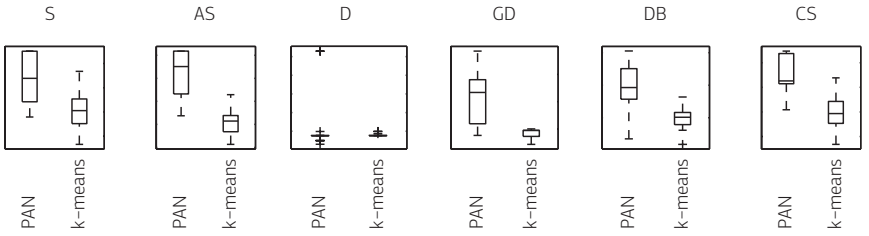


Figure 3.14 Achieved Hypervolume (larger is better) of PAN and the iterative k-medoids algorithm, for six selected indices. Visualization of the data shown in Table 3.2.

items has to be selected which is evaluated in two separate knapsacks, where each item has a different profit and weight in each knapsack. Each solution $x = \{x_1, x_2, ..., x_{150}\} \in \{0,1\}^{150}$ is a binary string of length 150, saying for each item whether it is selected or not. The biobjective knapsack problem is a constrained problem, where for each feasible solution x, $\sum_{i=1}^{150} x_i \cdot w_i^1 \leq 0.2 \cdot \sum_{i=1}^{150} w_i^1$ and $\sum_{i=1}^{150} x_i \cdot w_i^2 \leq 0.2 \cdot \sum_{i=1}^{150} w_i^2$ must hold, i.e. the total weight of all selected items in each knapsack must not exceed 20 percent of the total weight of all items of that knapsack. The objectives then are the sum of profits of each knapsack, i.e. the first objective is to maximize $\sum_{i=1}^{150} x_i \cdot p_i^1$, and the second objective is to maximize $\sum_{i=1}^{150} x_i \cdot p_i^2$. These objectives can be transformed easily into minimization problems using the following formula:

$$f_1(x) = \sum_{i=1}^{150} p_i^1 - \sum_{i=1}^{150} x_i \cdot p_i^1$$
$$f_2(x) = \sum_{i=1}^{150} p_i^2 - \sum_{i=1}^{150} x_i \cdot p_i^2$$

We used the integer programming problem solver CPLEX to generate the exact Pareto-optimal front for one instance of this knapsack problem with 150 items. The resulting front contained 138 Pareto-optimal solutions that were clustered using PAN. We applied PAN using the AS, CS and DB indices, using direct representation, a population size of 20 and running PAN for 100 000 function evaluations. We found that the AS index has a tendency to produce many small clusters, whereas the CS produces two small and one large cluster. The DB index, on the other hand, produces a

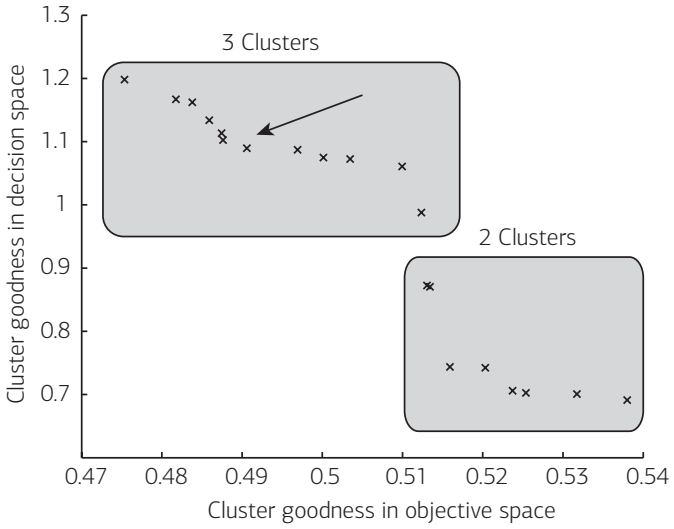


Figure 3.15 Partitionings resulting from one PAN run on the knapsack problem using the Davies Bouldin index. All found partitionings either had two or three clusters (as indicated). The chosen partitioning that will be inspected in more detail is indicated with an arrow.

few clusters of reasonable size. We only show the results of the DB index in the following. When looking more closely at the found partitionings shown in Figure 3.15, it was found that they can be classified in two templates, one that contains three clusters and one that contains two clusters. All partitionings are similar to one of these two partitioning templates. We look at a partitioning with three clusters in the following.

To visualize one solution in decision space, the profits of the chosen and discarded items are plotted. Note that there are 17 items that are selected in all Pareto-optimal solutions, and 75 items that are never selected in any of the Pareto-optimal solutions. The profits of these items are not plotted, although a decision maker might certainly look at them to learn more about the problem at hand. To interpret differences and similarities of clusters, only the 58 items that are selected in some solutions, but not in others are plotted. To plot a whole cluster, the cluster medoid and the solution furthest from the medoid are calculated and the items that are selected/not selected

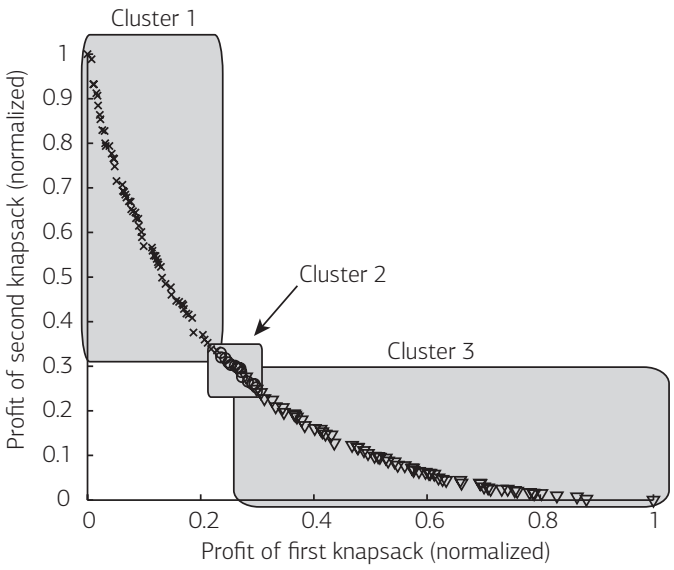


Figure 3.16 Chosen partitioning with its three clusters (crosses, circles and squares) in objective space.

in those two representative solutions are plotted. Also, it is indicated which items are selected in all/no solutions of that cluster.

The results are shown in Figures 3.16 and 3.17. As can be seen, there are two large clusters and one small cluster, where the two large clusters cover the two extremal regions of the Pareto-front, and the small cluster covers the middle region. When looking at the decision space as shown in Figure 3.17, the connection between selected items and location of the solution on the front can be seen. Cluster 1 contains the solutions with the highest profit in the first knapsack, and at the same time with the lowest profit in the second knapsack (remember that the profits in Figure 3.16 are transformed to yield a minimization problem). Several items are selected / not selected in all solutions of this first cluster, and the selected solutions all have a good profit in the first knapsack. In the third cluster, the opposite holds, namely the items selected in all solutions mainly have a good profit in the second knapsack, leading to solutions with a good overall

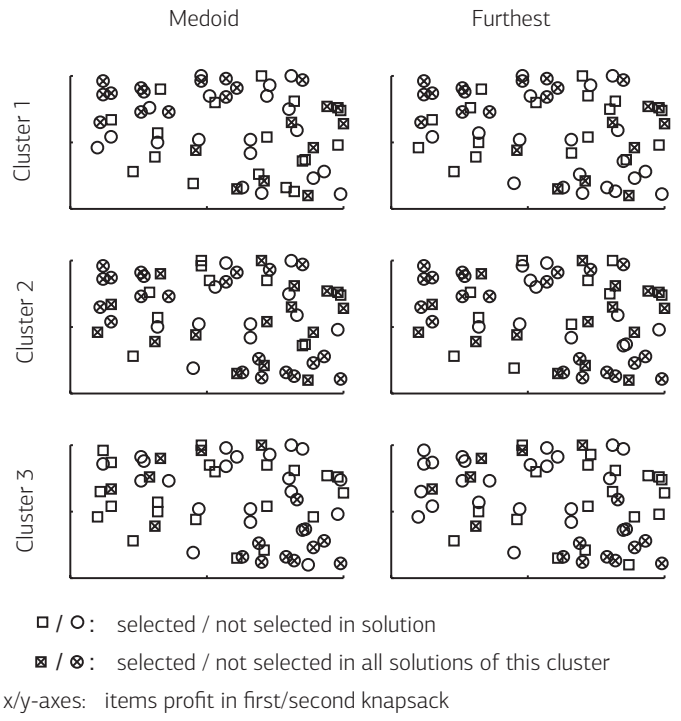


Figure 3.17 For each cluster (rows), the medoid and the solution furthest from the medoid are plotted. Each plot shows the knapsack items selected / not selected in that specific solutions, plus the items selected / not selected in all solutions of that cluster.

profit in the second knapsack. Finally, the middle cluster contains solutions that selected items from the whole range of profits in both knapsacks. As for the difference between cluster medoids and solutions furthest from the medoid, it can be noted that the Hamming distance between the medoid and furthest solution are 12, 8 and 12 items for the first, second and third cluster, respectively. For the whole dataset, the Hamming distance between solutions varies between 2 and 43 items (remember that from the 150 items, only 58 are not selected/deselected in all solutions), with a mean distance of 15.22 ± 7.78.

Application to Bridge Construction Problem

We also applied our algorithm to a real-world problem. As a problem, we selected the bridge construction problem which is described in Appendix B, where the goal is to build a truss bridge that can carry a fixed load. We generated a set of optimized bridges using DIOP (see Section 2.4) for 100 000 function evaluations and with a population size of 100. We chose DIOP because a standard multi-objective evolutionary algorithm produces a set of similar-looking bridges that are not interesting to cluster. DIOP on the other hand optimizes the bridges for structural diversity, while having constraints on the bridge's objective values. After deleting duplicates, i.e. bridges with decision or objective space[5] distance zero, 98 solutions remain for the partitioning. Note that these bridges are optimized for diversity, i.e. there should be no natural clusters of bridges. Therefore, the clustering task actually is hard, and no trivial partitioning can be expected. The different indices handle the situation in different ways. The S index was not tested as it takes considerably longer to compute than the other three indices. Also, the AS index that has been proposed to solve the speed problem of the S index can be used instead. The AS index itself handles the problem of a non-trivial dataset by generating a large number of small clusters. The CS index, on the other hand, finds a few good small clusters, and one large cluster that contains the remaining bridges. The DB index is even more extreme and generates the minimum number of clusters, i.e. two, where one cluster is large and the other small. We here only show the results of the CS index.

We clustered all the bridges in the given set using the direct representation and the CS index, with a population size of 20 and for 80 000 function evaluations. The resulting partitionings are shown in Figure 3.18. In the following, one of the partitionings, indicated with an arrow in the figure, will be inspected more closely. The chosen partitioning is shown in Figure 3.19 and consists of a total of 8 clusters, out of which 7 are small clusters with either two or three bridges, and one large cluster containing all the remaining bridges. When inspecting the small clusters it can be seen that

[5] Duplicates in objective space are also deleted as some cluster validity indices cannot handle duplicates in either space.

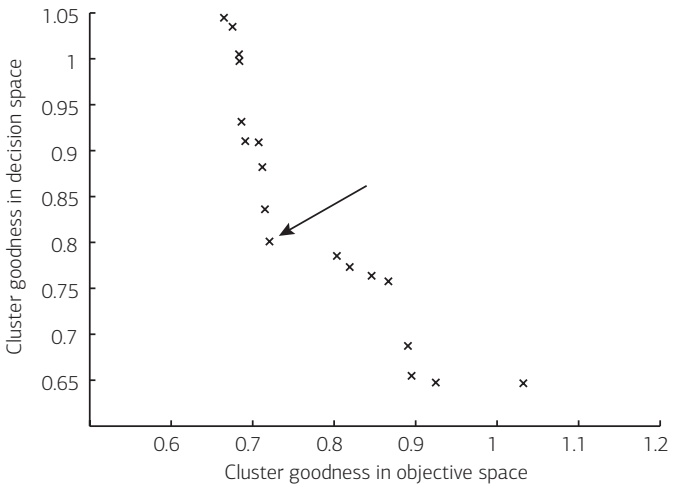


Figure 3.18 Partitionings found by PAN on the bridge dataset (measures have to be minimized).

they indeed contain similar looking bridges (one example is given in the upper right corner of the figure). The large cluster, on the other hand, contains many different looking bridges, though without the most distant looking bridges. Apparently, these bridges could not be put into smaller clusters without impeding the CS measure. When inspecting the covered objective space area of the smaller clusters, it can be seen that the area they cover is quite different. One extreme is the cluster with a length of the longest connection of 10 to 12 meters, and a weight between 800 and 1 100 kg. The other extreme is represented by the three clusters that all map to a point in objective space that has a length of the longest connection of approximately 21, and a weight of approximately 580 kg. Overall, a visualization of the whole front as shown in Figure 3.19 is a much more intuitive way of extracting information from the front than just plotting the objectives, or by cluttering the picture with plotting all 98 bridges.

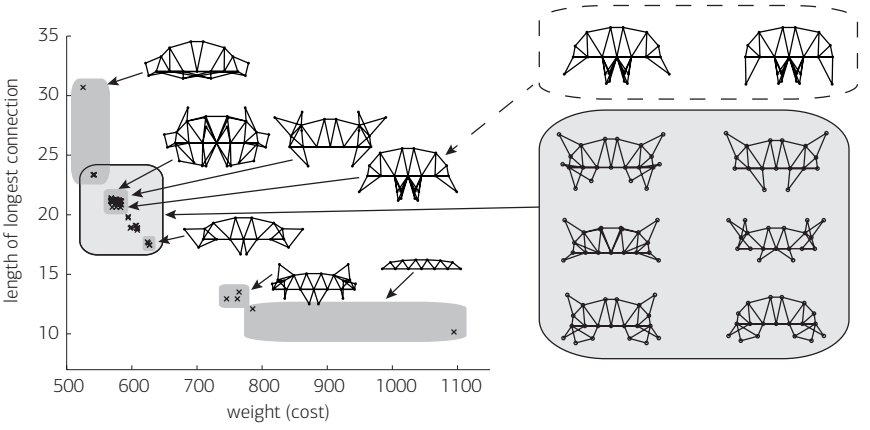


Figure 3.19 Partitioning achieved by PAN. The partitioning consists of 7 small clusters of 2 or 3 bridges each, and one large cluster of 83 bridges. The objective space values of the bridges are shown on the left, where the small clusters are indicated by a dark gray box. Each small cluster is represented by a random bridge out of that cluster. For one of the clusters, both bridges are shown (dashed box). The large cluster is indicated by a box with a black edge. 6 random bridges out of that cluster are shown on the right lower part.

3.4.5 · PAN Summary

In this section we cluster a set of solutions, such that clusters that are compact and well separated in both decision and objective space are generated. To this end, we formally define this clustering problem as a biobjective optimization problem, and designed PAN, a multi-objective evolutionary algorithm, to solve the problem. We tested several standard cluster validity indices for their use as optimization goals, and several representations found in the literature to represent a partitioning. Applying all representation / validity index combinations to cluster several artificial datasets with known optimal partitionings helped identifying the strengths and weaknesses of the different representations and validity indices, such that a combination that reliably produces good partitionings could be chosen.

PAN was then compared to the standard clustering approach of repeatedly using the k-medoids clustering algorithm for all possible number of clusters. It has been found that the partitionings found by PAN achieve a higher

hypervolume in terms of decision and objective space goodness than the partitionings found by the k-medoids algorithm. When applying PAN to a knapsack problem, the relation between selected items and achieved profits could be visualized. Also, the method was applied to a real-world truss bridge optimization problem, where a front containing 98 bridges could be visualized in a compact manner by representing each cluster by a representative bridge and by dividing the objective space into regions to which the particular clusters map. In conclusion, it has been found that the proposed method is able to adequately cluster the solutions, such that the clusters contain similar designs, and are located in compact regions in objective space.

In the future, a measure to quantify the goodness of different clustering tradeoffs should be developed. That way, validity indices could not only be compared according to the extreme Pareto-optimal partitionings (that are either best in decision or objective space), but also according to their tradeoffs between the two extreme partitionings. Also, there might be some user preferences, for example a maximum number of clusters that the user can handle, or the user might value cluster compactness more than cluster separations. PAN could therefore be adapted to incorporate such preferences. Furthermore, PAN could be extended to provide some help in picking one partitioning out of the set of partitionings which is produced.

3.5 · Comparison of Approaches

This chapter proposes two approaches to analyze Pareto-sets or approximations thereof. The first approach, MANA, assumes that the decision space is binary, and requires that there are two objectives and that the set of solutions to be analyzed only contains non-dominated solutions. The second approach, PAN, is applicable to problems with any number of objectives, with any type of decision space, as long as there is a distance measure to quantify the dissimilarity of two solutions, and it can handle sets that contain dominated solutions.

Both approaches were applied to the biobjective knapsack problem, and they both discovered the inherent relation between decision space and objective space values of solutions to the knapsack problem, i.e. that solutions which achieve a high profit in the first/second knapsack mainly contain items with a high profit in the first or second knapsack, respectively. On the other hand, there are several differences between the results of the two approaches. MANA yields a hierarchical clustering, whereas PAN yields one single partitioning, whose number of clusters is best with respect to the chosen cluster validity index. Also, MANA discovers subsets of items that are selected in a large number of solutions and clusters the solutions using these subsets, whereas PAN clusters solutions that are similar to each other in terms of the Hamming distance of the selected items. Here, the main difference is that PAN clusters solutions with *similar* selected items, whereas MANA clusters solutions with *equal* selected items.

Finally, PAN was applied to a bridge construction problem, which cannot be decoded into a binary decision space, and therefore, MANA cannot be used to analyze the resulting front. In conclusion, PAN is a general algorithm that can be applied to a wide variety of problems. MANA, on the other hand, aims at problems with binary decision spaces, and is useful if blocks of decision variables that are set to one are of interest.

4

Bounding the Effectiveness of the Hypervolume Indicator

In this chapter, we study bounds for the α-approximate effectiveness of non-decreasing $(\mu + \lambda)$-archiving algorithms that optimize the hypervolume. A $(\mu + \lambda)$-archiving algorithm defines how μ individuals are to be selected from a population of μ parents and λ offspring. It is non-decreasing if the μ new individuals never have a lower hypervolume than the μ original parents. An algorithm is α-approximate if for any optimization problem and for any initial population, there exists a sequence of offspring populations for which the algorithm achieves a hypervolume of at least $1/\alpha$ times the maximum hypervolume.

Bringmann and Friedrich [12] have proven that all non-decreasing, locally optimal $(\mu+1)$-archiving algorithms are $(2+\varepsilon)$-approximate for any $\varepsilon > 0$. We extend this work and substantially improve the approximation factor by generalizing and tightening it for any choice of λ to $\alpha = 2 - (\lambda - p)/\mu$ with $\mu = q \cdot \lambda - p$ and $0 \leq p \leq \lambda - 1$. In addition, we show that $1 + \frac{1}{2\lambda} - \delta$, for

$\lambda < \mu$ and for any $\delta > 0$, is a lower bound on α, i.e. there are optimization problems where one cannot get closer than a factor of $1/\alpha$ to the optimal hypervolume.

4.1 · Motivation and Background

When optimizing multiple conflicting objectives, there usually is no single best solution. Instead, there are incomparable tradeoff solutions, where no solution is strictly better than any other solution. *Better* in this case refers to Pareto-dominance, i.e. one solution is said to be better than another, or dominate it, if it is equal or better in all objectives, and strictly better in at least one objective. The set of non-dominated solutions is called the Pareto-optimal set. Usually, this Pareto-optimal set can contain a large number of solutions, and it is infeasible to calculate all of them. Instead, one is interested in finding a relatively small, but still *good* subset of this Pareto-optimal set.

It is not a priori clear how a good subset should look like, i.e. how the goodness of a subset can be measured. One of the most popular measures for subset quality is the hypervolume indicator, which measures the volume of the dominated space. Therefore, one possibility to pose a multi-objective optimization problem is to look for a solution set $\mathcal{P}^*$ of fixed size, which maximizes the hypervolume.

Algorithms that optimize the hypervolume face several problems. First, the number of possible solutions can become very large, so it is not possible to select from all solutions. Second, even if all solutions are known and the non-dominated solutions can be identified, the number of subsets explodes and not all of them can be enumerated for comparison.

In this chapter, we consider $(\mu + \lambda)$-Evolutionary Algorithms, or $(\mu + \lambda)$-EAs. They iteratively improve a set of solutions, where the set is named *population* and the iteration is denoted as *generation*. In particular, they maintain a population of size μ, generate λ offspring from the μ parents and then select μ solutions from the μ parents and the λ offspring that are

to survive into the next generation. Note that we here only consider non-decreasing algorithms, i.e. algorithms whose hypervolume cannot decrease from one generation to the next.

Several questions arise in this setting. First, what are upper and lower bounds on the hypervolume that a population of a fixed size will achieve? Is it possible to prove that a set of size μ with the maximal hypervolume can be found, without explicitly testing all possible sets? To answer these questions, the term *effectiveness* has been defined. An algorithm is effective if for any optimization problem[1] and for any initial population[2], there is a sequence of offspring[3] which leads to the population with maximum hypervolume. Obviously, $(\mu + \mu)$-EAs are always effective: We just choose the first set of offspring to be exactly the population with the maximal hypervolume and then we select this set as the new population. It has also been shown by Zitzler et al.[124] that $(\mu + 1)$-EAs, on the other hand, are ineffective. Recently, it has been shown by Bringmann and Friedrich [12] that all $(\mu + \lambda)$-EAs with $\lambda < \mu$ are ineffective.

Bringmann and Friedrich then raised the follow-up question: If it is not possible to reach the optimal hypervolume for all optimization problems and all initial populations, is it at least possible to give a lower bound on the achieved hypervolume? To this end, they introduced the term *α-approximate effectiveness*. An algorithm is α-approximate if for any optimization problem and for any initial population there is a sequence of offspring with which the algorithm achieves at least $1/\alpha \cdot I_H^{\max}$, where $I_H^{\max}$ is the maximum achievable hypervolume of a population of size μ. They proved in their paper that a $(\mu + 1)$-EA is 2-approximate and conjectured that for larger λ, a $(\mu + \lambda)$-EA is $O(1/\lambda)$-approximate.

On the other hand, we might also be interested in upper bounds on the achievable hypervolume. Bringmann and Friedrich [12] have found an opti-

[1] We only consider finite search spaces here, such that mutation operators exist which produce offspring with a probability larger than zero. Note that any search space coded on a computer is finite.

[2] Note that the term *for any initial population* implies that at any point during the algorithm, there exists a sequence of offspring with which an effective algorithm can achieve the optimal hypervolume.

[3] Note that the term *there is a sequence of offspring* assumes that we are given variation operators that produce any sequence of offspring with probability greater than zero.

mization problem where no algorithm can achieve more than $1/(1+0.1338(1/\lambda-1/\mu)-\varepsilon)$ of the optimal hypervolume, i.e. there is no $(1+0.1338(1/\lambda-1/\mu)-\varepsilon)$-approximate archiving algorithm for any $\varepsilon > 0$.

Why is knowledge of the bounds of the α-approximate effectiveness useful? Assume that we are using an exhaustive mutation operator, which produces any offspring with a probability larger than zero. Therefore, the probability of generating an arbitrary sequence of offspring is also larger than zero. The $\frac{1}{2}$-approximate effectiveness of $(\mu+1)$-EAs now tells us that if we execute the evolutionary algorithm for a sufficiently large number of generations, we will end up with a population that has at least half of the maximal hypervolume. In case of a $(\mu + \mu)$-EA, on the other hand, we know that we will finally achieve a population with maximum hypervolume, i.e. $\alpha = 1$. We are therefore interested in deriving bounds on the effectiveness of evolutionary algorithms.

This chapter extends the work of Bringmann and Friedrich by (a) computing the α-approximate effectiveness of $(\mu + \lambda)$-EAs for general choices of λ, (b) tightening the previously known upper bound on α, and (c) tightening the previously known lower bound on α. The results for (a) and (b) are based on the theory of submodular functions, see [31]. For (c) we show that for $\lambda < \mu$, there exist optimization problems where any $(\mu+\lambda)$-EA does not get closer than a factor of $1/\alpha$ to the optimal hypervolume with $\alpha = 1 + \frac{1}{2\lambda} - \delta$, for any $\delta > 0$.

The chapter is organized as follows: The next section presents the formal setting, including the definition of the hypervolume, the algorithmic setting, definitions for the effectiveness and approximate effectiveness and an introduction into submodular functions. In Section 4.3 we determine an upper bound on α for general choices of μ and λ, thereby giving a quality guarantee in terms of a lower bound of the achievable hypervolume. Finally in Section 4.4, we will determine a lower bound on α for general choices of μ and λ.

4.2 · Preliminaries

We here follow the notation introduced in Section 1.1, and shortly revisit the relevant concepts. Consider a multi-objective minimization problem with a decision space $\mathcal{X}$ and an objective space $\mathcal{Y} \subseteq \mathbb{R}^m = \{f(x)|x \in \mathcal{X}\}$, where $f : \mathcal{X} \to \mathcal{Y}$ denotes a mapping from the decision space to the objective space with m objective functions $f = \{f_1, ..., f_m\}$ which are to be minimized.

The underlying preference relation is weak Pareto-dominance, where a solution $a \in \mathcal{X}$ weakly dominates another solution $b \in \mathcal{X}$, denoted as $a \preceq b$, if and only if solution a is better or equal than b in all objectives, i.e. iff $f(a) \leqslant f(b)$, or equivalently, iff $f_i(a) \leq f_i(b), \forall i \in \{1, ..., m\}$. In other words, a point $p \in \mathcal{X}$ weakly dominates the region $\{y \in \mathbb{R}^m : f(p) \leqslant y\} \subset \mathbb{R}^m$.

4.2.1 · Hypervolume Indicator

The hypervolume indicator of a given set $\mathcal{P} \subseteq \mathcal{X}$ is the volume of all points in $\mathbb{R}^m$ which are dominated by at least one point in $\mathcal{P}$ and which dominate at least one point of a reference set $\mathcal{R} \subset \mathbb{R}^m$.[4] Roughly speaking, the hypervolume measures the size of the dominated space of a given set. Sets with a larger hypervolume are considered better. More formally, the hypervolume indicator can be written as

$$I_H(\mathcal{P}) := \int_{y \in \mathbb{R}^m} A_{\mathcal{P}}(y)\, dy$$

where $A_{\mathcal{P}}(y)$ is called the *attainment function* of set $\mathcal{P}$ with respect to a given reference set $\mathcal{R}$, and is defined as follows:

$$A_{\mathcal{P}}(y) = \begin{cases} 1 & \text{if } \exists p \in \mathcal{P}, r \in \mathcal{R} : f(p) \leqslant y \leqslant r \\ 0 & \text{else} \end{cases}$$

[4] No assumptions on the reference set have to be made, as our results have to hold for any objective space, including the one only containing solutions that dominate at least one reference point. If that set is empty, all algorithms are effective, as the hypervolume is always zero.

1: **function** EA(μ, λ, g)
2: $\quad\mathcal{P}^0 \leftarrow$ initialize with μ random solutions
3: $\quad$**for** $t = 1$ to g **do**
4: $\quad\quad\mathcal{O}^t \leftarrow$ generate λ offspring
5: $\quad\quad\mathcal{P}^t \leftarrow$ select μ solutions from $\mathcal{P}^{t-1} \cup \mathcal{O}^t$
6: $\quad$**return** $\mathcal{P}^g$

Algorithm 15 General $(\mu + \lambda)$-EA framework: μ denotes the population size; λ the offspring size; the algorithm runs for g generations.

The goal of a $(\mu + \lambda)$-EA is to find a population $\mathcal{P}^* \subseteq \mathcal{X}$ of size μ with the maximum hypervolume:

$$I_H(\mathcal{P}^*) = \max_{\mathcal{P} \subseteq \mathcal{X}, |\mathcal{P}|=\mu} I_H(\mathcal{P}) = I_{H,\mu}^{\max}(\mathcal{X})$$

4.2.2 · Algorithmic Setting

The general framework we are considering here is based on a $(\mu+\lambda)$ Evolutionary Algorithm (EA) as shown in Algorithm 15. The selection step of Line 5 is done by a $(\mu+\lambda)$-archiving algorithm[5]. We here assume that the archiving algorithm is non-decreasing, i.e. $I_H(\mathcal{P}^t) \geq I_H(\mathcal{P}^{t-1})$, $1 \leq t \leq g$. We use the following formal definition (as given in [12]) to describe an archiving algorithm:

Definition 4.1 (archiving algorithm): *A $(\mu + \lambda)$-*archiving algorithm *A is a partial mapping $A : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ such that for a μ-population $\mathcal{P}$ and a λ-population $\mathcal{O}$, $A(\mathcal{P}, \mathcal{O})$ is a μ-population and $A(\mathcal{P}, \mathcal{O}) \subseteq \mathcal{P} \cup \mathcal{O}$.*

Using this definition, the for-loop in Algorithm 15 can be described as follows, see also [12]:

Definition 4.2 (population at generation t): *Let $\mathcal{P}^0$ be a μ-population and $\mathcal{O}^1, ..., \mathcal{O}^N$ a sequence of λ-populations. Then*

$$\mathcal{P}^t := A(\mathcal{P}^{t-1}, \mathcal{O}^t) \quad \textit{for all } t = 1, ..., N$$

[5] We use the term *archiving algorihm* here to be compliant with [12]. It does not mean that we keep a separate archive in addition to the population $\mathcal{P}^t$.

We also define

$$\begin{aligned} A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^t) &:= A(A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^{t-1}), \mathcal{O}^t) \\ &= A(...A(A(\mathcal{P}^0, \mathcal{O}^1), \mathcal{O}^2), ..., \mathcal{O}^t) \\ &= \mathcal{P}^t \quad \textit{for all } t = 1, ..., N \end{aligned}$$

As mentioned above, we only consider non-decreasing archiving algorithms which are defined as follows, see also [12]:

Definition 4.3 (non-decreasing archiving algorithm): *An archiving algorithm A is non-decreasing, if for all inputs $\mathcal{P}$ and $\mathcal{O}$, we have*

$$I_H(A(\mathcal{P}, \mathcal{O})) \geq I_H(\mathcal{P})$$

4.2.3 · Effectiveness and Approximate Effectiveness

Following Bringmann and Friedrich [12], we here assume a worst-case view on the initial population and a best-case view on the choice of offspring. This means that we would like to know for any optimization problem, starting from any initial population, whether there exists a sequence of offspring populations such that the EA is able to find a population with the maximum possible hypervolume. If so, the archiving algorithm is called *effective*:

Definition 4.4 (effectiveness): *A $(\mu + \lambda)$-archiving algorithm A is* effective*, if for all finite sets $\mathcal{X}$, all objective functions f and all μ-populations $\mathcal{P}^0 \subseteq \mathcal{X}$, there exists an $N \in \mathbb{N}$ and a sequence of λ-populations $\mathcal{O}^1, ..., \mathcal{O}^N \subseteq \mathcal{X}$ such that*

$$I_H(A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^N)) = I^{max}_{H,\mu}(\mathcal{X})$$

Similarly, we use the following definition for the approximate effectiveness, which quantifies the distance to the optimal hypervolume that can be achieved:

Definition 4.5 (approximate effectiveness): *Let $\alpha \geq 1$. A $(\mu + \lambda)$-archiving algorithm A is* α-approximate *if for all finite sets $\mathcal{X}$, all objective functions*

f and all μ-populations $\mathcal{P}^0 \subseteq \mathcal{X}$, there exists an $N \in \mathbb{N}$ and a sequence of λ-populations $\mathcal{O}^1, ..., \mathcal{O}^N$ such that

$$I_H(A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^N)) \geq \frac{1}{\alpha} I_{H,\mu}^{max}(\mathcal{X})$$

Of course, an effective archiving algorithm is 1-approximate. Here, we are interested in deriving bounds on α for any choice of μ and λ.

4.2.4 · Submodular Functions

The theory of submodular functions has been widely used to investigate problems where one is interested in selecting optimal subsets of a given size. But what exactly is a submodular function? At first, they map subsets of a given base set to real numbers, just like the hypervolume indicator defined above. In addition, submodular functions show a diminishing increase when adding points to sets that become larger. In other words, let us define the set function $z : 2^{\mathcal{X}} \rightarrow \mathbb{R}$, where $2^{\mathcal{X}}$ is the power set of the decision space. Then the contribution of a point $s \in \mathcal{X}$ with respect to set $\mathcal{A} \subset \mathcal{X}$ is $c(s, \mathcal{A}) = z(\mathcal{A} \cup \{s\}) - z(\mathcal{A})$. When z is a submodular function, the contribution $c(s, \mathcal{A})$ gets smaller when $\mathcal{A}$ becomes larger. More formally, a submodular function z is defined as follows:

$$\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{X}, \forall s \in \mathcal{X} \backslash \mathcal{B} : z(\mathcal{A} \cup \{s\}) - z(\mathcal{A}) \geq z(\mathcal{B} \cup \{s\}) - z(\mathcal{B})$$

i.e. if set $\mathcal{A}$ is contained in set $\mathcal{B}$, the contribution of adding a point s to $\mathcal{A}$ is larger or equal than the contribution of adding s to $\mathcal{B}$. A submodular function is non-decreasing if it is monotone in adding points:

$$\forall \mathcal{B} \subseteq \mathcal{X}, \forall s \in \mathcal{X} \backslash \mathcal{B} : z(\mathcal{B} \cup \{s\}) \geq z(\mathcal{B})$$

Now, we show that the hypervolume indicator as defined above is non-decreasing and submodular.

Theorem 4.6: *The hypervolume indicator $I_H(\mathcal{P})$ is non-decreasing submodular.*

Proof. At first, we define the contribution of a solution s to a set $\mathcal{B}$ as

$$I_H(\mathcal{B} \cup \{s\}) - I_H(\mathcal{B}) = \int_{y \in \mathbb{R}^m} C(\mathcal{B}, s, y)\, dy$$

with

$$C(\mathcal{B}, s, y) = A_{\mathcal{B}\cup\{s\}}(y) - A_{\mathcal{B}}(y)$$

Using the definition of the attainment function A we find

$$C(\mathcal{B}, s, y) = \begin{cases} 1 & \text{if } (\exists r \in \mathcal{R} : f(s) \leqslant y \leqslant r) \wedge (\nexists p \in \mathcal{B} : f(p) \leqslant y) \\ 0 & \text{else} \end{cases}$$

As $C(\mathcal{B}, s, y)$ is non-negative, the hypervolume indicator is non-decreasing.

Consider two arbitrary sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{X}$ with $\mathcal{A} \subseteq \mathcal{B}$, and an arbitrary solution $s \in \mathcal{X}$, $s \notin \mathcal{B}$. To prove that the hypervolume indicator is submodular, we have to show that

$$I_H(\mathcal{A} \cup s) - I_H(\mathcal{A}) \geq I_H(\mathcal{B} \cup s) - I_H(\mathcal{B})$$

or equivalently

$$\int_{y \in \mathbb{R}^m} C(\mathcal{A}, s, y)\, dy \geq \int_{y \in \mathbb{R}^m} C(\mathcal{B}, s, y)\, dy \tag{4.1}$$

for $\mathcal{A} \subseteq \mathcal{B}$, $s \notin \mathcal{B}$.

To this end, we will show that for all $y \in \mathbb{R}^m$ the inequality $C(\mathcal{A}, s, y) \geq C(\mathcal{B}, s, y)$ holds. As $C(\cdot, \cdot, \cdot)$ can only assume the values 0 and 1, we have to show that for all $y \in \mathbb{R}^m$, $s \notin \mathcal{B}$ we have

$$C(\mathcal{A}, s, y) = 0 \quad \Rightarrow \quad C(\mathcal{B}, s, y) = 0$$

Following the definition of C, there are the following three cases where $C(\mathcal{A}, s, y) = 0$:

1. $(\nexists r \in \mathcal{R} : y \leqslant r)$: In this case, we also have $C(\mathcal{B}, s, y) = 0$ as the condition is the same for $C(\mathcal{A}, s, y)$ and $C(\mathcal{B}, s, y)$.

2. $(f(s) \not\leqslant r)$: Again, we find $C(\mathcal{B}, s, y) = 0$ as the condition is the same for $C(\mathcal{A}, s, y)$ and $C(\mathcal{B}, s, y)$.
3. $(\exists p \in \mathcal{A} : f(p) \leqslant y)$: In other words, there exists a solution $p \in \mathcal{A}$ in $\mathcal{A}$ which weakly dominates y. But as $\mathcal{A} \subseteq \mathcal{B}$, we also have $p \in \mathcal{B}$ and therefore, $(\exists p \in \mathcal{B} : f(p) \leqslant y)$. Therefore, we find $C(\mathcal{B}, s, y) = 0$.

As a result, (4.1) holds and the hypervolume indicator is submodular. □

4.3 · Upper Bound on the Approximate Effectiveness

In this section, we will provide quality guarantees on the hypervolume achieved by an EA in terms of the α-approximate effectiveness, i.e. we will provide an upper bound on α for all population sizes μ and offspring set sizes λ.

In the previous section, we showed that the hypervolume is non-decreasing submodular. Nemhauser, Wolsey and Fisher [75] have investigated interchange heuristics for non-decreasing set functions and showed approximation properties in case of submodular set functions. We will first show that the interchange heuristic in [75] is execution-equivalent to the previously defined $(\mu + \lambda)$-EA framework. Then, the approximation properties for the R-interchange heuristics are used to determine upper bounds on α.

The heuristic described in [75] is shown in Algorithm 16 where we deliberately changed the variable names to make them fit to the notations introduced so far. It makes use of the difference between sets, which is defined as follows: Given two sets $\mathcal{A}$ and $\mathcal{B}$, the difference between $\mathcal{A}$ and $\mathcal{B}$ is $\mathcal{A} - \mathcal{B} = \{x : x \in \mathcal{A} \wedge x \notin \mathcal{B}\}$, i.e. the set of all solutions which are contained in $\mathcal{A}$ but not in $\mathcal{B}$.

The heuristic in Algorithm 16 is of a very general nature. No assumptions are made about the starting population $\mathcal{P}^0$, and the method of searching for $\mathcal{P}^t$. For example, we can set the function $z(\mathcal{P}) = I_H(\mathcal{P})$ and then choose the following strategy for Line 5:

1. Determine a set $\mathcal{O}^t$ of offspring of size λ.

```
1: function heuristic(μ, λ)
2:     P^0 ← initialize with an arbitrary set of size μ
3:     t ← 1
4:     while true do
5:         determine a set P^t of size μ with |P^t − P^{t−1}| ≤ λ such that z(P^t) > z(P^{t−1})
6:         if no such a P^t exists then
7:             break
8:         t ← t + 1
9:     return P^G ← P^{t−1}
```

Algorithm 16 Interchange heuristic: μ is the size of the final set; λ the maximum number of elements which can be exchanged.

2. Select μ solutions from $\mathcal{P}^{t-1} \cup \mathcal{O}^t$ using an archiving algorithm A, i.e. $\mathcal{S} = A(\mathcal{P}^{t-1}, \mathcal{O}^t)$.
3. Execute the above two steps until $I_H(\mathcal{S}) > I_H(\mathcal{P}^{t-1})$ and then set $\mathcal{P}^t = \mathcal{S}$, or until no such $\mathcal{S}$ can be found.

Following Algorithm 16, the above steps need to guarantee that a set $\mathcal{P}^t$ with $I_H(\mathcal{P}^t) > I_H(\mathcal{P}^{t-1})$ is found if it exists. For example, we can use an exhaustive offspring generation, i.e. every subset of size λ of the decision space $\mathcal{X}$ can be determined with a probability larger than zero. Moreover, the archiving algorithm A must be able to determine an improved subset of $\mathcal{P}^{t-1} \cup \mathcal{O}^t$ if it exists. In other words, we require from A that $I_H(A(\mathcal{P}, \mathcal{O})) > I_H(\mathcal{P})$ if there exists a subset of $\mathcal{P} \cup \mathcal{O}$ of size μ with a larger hypervolume than $I_H(\mathcal{P})$. For example, A may in turn remove all possible subsets of size λ from $\mathcal{P}^{t-1} \cup \mathcal{O}^t$ and return a set that has a better hypervolume than $\mathcal{P}^{t-1}$. Note that this instance of the interchange heuristic can be easily rephrased in the general $(\mu + \lambda)$-EA framework of Algorithm 15 with an unbounded number of generations.

Nemhauser et al. [75] have proven the following result for the interchange heuristic:

Theorem 4.8: *Suppose z is non-decreasing and submodular. Moreover, define the optimization problem $z^* = \max_{\mathcal{P} \subseteq \mathcal{X}, |\mathcal{P}| \leq \mu} z(\mathcal{P})$. If $\mu = q \cdot \lambda - p$ with q a positive integer, and p integer with $0 \leq p \leq \lambda - 1$, then*

$$\frac{z^* - z(\mathcal{P}^G)}{z^* - z(\emptyset)} \leq \frac{\mu - \lambda + p}{2\mu - \lambda + p}$$

where $z(\mathcal{P}^G)$ is the value of the set obtained by Algorithm 16 and $z(\emptyset)$ is the value of the empty set.

We have shown that the hypervolume indicator is non-decreasing submodular. Therefore, if we set the function $z(\mathcal{P}) = I_H(\mathcal{P})$ and note that $I_H(\emptyset) = 0$, we can easily obtain the following bound on the approximation quality of Algorithm 16:

Proposition 4.9: *If $\mu = q \cdot \lambda - p$ with an integer $0 \leq p \leq \lambda - 1$, then*

$$I_H(\mathcal{P}^G) \geq \frac{1}{2 - \frac{\lambda - p}{\mu}} \cdot I_{H,\mu}^{\max}(\mathcal{X}) \tag{4.2}$$

This bound can be compared to the definition of the approximate effectiveness, see Definition 4.5, i.e. it bounds the achievable optimization quality in terms of the hypervolume if a certain algorithm structure is used. But whereas Definition 4.5 and the corresponding value of $\alpha = 2 + \varepsilon$ from [12] is related to Algorithm 15, the above bound with $\alpha = 2 - \frac{\lambda - p}{\mu}$ is related to Algorithm 16.

We will now show that the improved approximation bound of $\alpha = 2 - \frac{\lambda - p}{\mu}$ is valid also in the case of Algorithm 15, thereby improving the results in [12].

Theorem 4.10: *Suppose a non-decreasing $(\mu + \lambda)$-archiving algorithm which satisfies in addition*

$$\exists \mathcal{S} \ : \ (\mathcal{S} \subset \mathcal{P} \cup \mathcal{O}) \wedge (|\mathcal{S}| = \mu) \wedge (I_H(\mathcal{S}) > I_H(\mathcal{P})) \quad \Rightarrow \quad I_H(A(\mathcal{P}, \mathcal{O})) > I_H(\mathcal{P})$$

Then for all finite sets $\mathcal{X}$, all objective functions f and all μ-populations $\mathcal{P}^0 \subseteq \mathcal{X}$ the following holds: For any run of an instance of Algorithm 16, one can determine a sequence of λ-populations $\mathcal{O}^1, ..., \mathcal{O}^N$ such that

$$I_H(A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^N)) = I_H(\mathcal{P}^G)$$

Proof. The proof uses the special instance of Algorithm 16 that has been introduced above. Line 5 is implemented as follows: (1) Determine a set $\mathcal{O}^t$ of offspring of size λ using an exhaustive generation, i.e. each subset of $\mathcal{X}$ is determined with non-zero probability. (2) Use the archiving algorithm A to determine a set $\mathcal{S} = A(\mathcal{P}^{t-1}, \mathcal{O}^t)$. (3) Repeat these two steps until $I_H(\mathcal{S}) > I_H(\mathcal{P}^{t-1})$ or no such $\mathcal{S}$ can be found. Due to the required property of A, no such $\mathcal{S}$ can be found if it does not exist.

Algorithm 16 yields as final population $\mathcal{P}^G = \mathcal{P}^{t-1}$ which can be rewritten as $\mathcal{P}^{t-1} = A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^{t-1})$ The sets of offspring $\mathcal{O}^i$ are generated as described above. Using $N = t-1$ yields the required result $I_H(A(\mathcal{P}^0, \mathcal{O}^1, ..., \mathcal{O}^N)) = I_H(\mathcal{P}^G)$. □

As a direct consequence of the execution equivalence between Algorithm 15 and Algorithm 16 according to the above theorem, the Definition 4.5 and (4.2), we can state the following result:

Proposition 4.12: *A non-decreasing $(\mu + \lambda)$-archiving algorithm $A(\mathcal{P}, \mathcal{O})$, which yields a subset of $\mathcal{P} \cup \mathcal{O}$ of size μ with a better hypervolume than that of $\mathcal{P}$ if there exists one, is $(2 - \frac{\lambda - p}{\mu})$-approximate where $\mu = q \cdot \lambda - p$ with an integer $0 \leq p \leq \lambda - 1$.*

It is interesting to note two special cases of the above proposition:

1. $\mu = \lambda$: In this case, we have a $(\mu+\mu)$-EA. It holds that $p = 0$ and therefore, the formula evaluates to $\alpha = 1$, which means that this algorithm actually is effective. This corresponds to the obvious result mentioned in Section 4.1.
2. $\lambda = 1$: In this case, we have a $(\mu+1)$-EA. It holds that $p = 0$ and $q = \mu$ and therefore, the formula evaluates to $\alpha = 2 - \frac{1}{\mu}$, which is tighter than the bound of Bringmann and Friedrich [12].

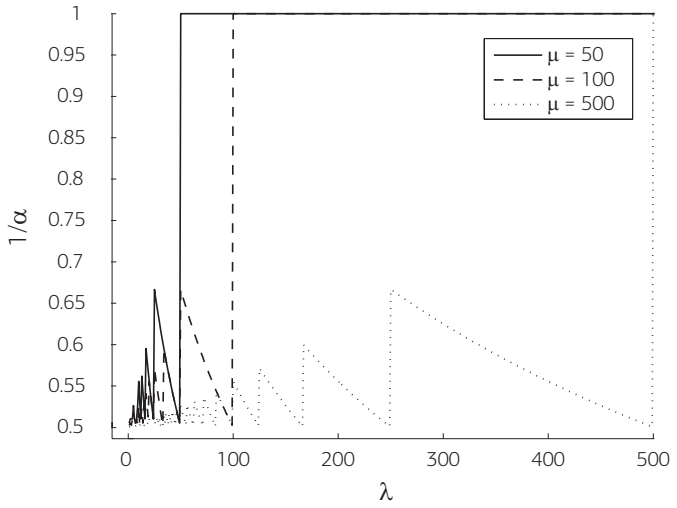


Figure 4.1 Quality guarantees for the hypervolume achieved by a $(\mu + \lambda)$-EA. For a given μ and a given λ, there is a sequence of offspring such that at least $\frac{1}{\alpha} \cdot I_{H,\mu}^{max}(\mathcal{X})$ can be achieved, irrespective of the optimization problem and the chosen initial population.

Figure 4.1 shows the relation between λ and α for several settings of μ. As can be seen, it is a zigzag line which corresponds to the modulo-like definition of p and q. The local maxima of each line are located where μ is an integer multiple of λ.

4.4 · Lower Bound on the Approximate Effectiveness

In the previous section we gave an upper bound on α. In this section, on the other hand, we will give a lower bound on α. This lower bound is tight for $\mu = 2$, i.e. is equal to the upper bound. To find this bound, we will show that there exist optimization problems and initial populations, such that any non-decreasing archiving algorithm will end up with a hypervolume that is at most $1/(1 + \frac{1}{2\lambda})$ of the optimal hypervolume. Whereas a first particular example has been shown in [124], a more general lower bound was shown in [12], where Bringmann and Friedrich found a problem where

any non-decreasing archiving algorithm ends up with a hypervolume that is at most $1/(1+0.1338(1/\lambda - 1/\mu) - \varepsilon)$ of the optimal hypervolume, for any $\varepsilon > 0$. The new bound substantially tightens the result of [12], but relies on the general definition of the hypervolume indicator which uses a reference set $\mathcal{R}$ instead of a single reference point.

Theorem 4.13: *Let $\lambda < \mu$. There is no α-approximate non-decreasing $(\mu+\lambda)$-archiving algorithm for any $\alpha < 1 + \frac{1}{2\lambda}$.*

Proof. We proof this theorem by finding a population $\mathcal{P}^0 = \{s_0, ..., s_{\mu-1}\}$ whose hypervolume indicator $I_H(\mathcal{P}^0)$ cannot be improved by any non-decreasing $(\mu+\lambda)$-archiving algorithm, i.e. it is locally optimal. At the same time, the optimal population $\mathcal{P}^* = \{o_0, ..., o_{\mu-1}\}$ has a hypervolume indicator value of $I_H(\mathcal{P}^*)$ which satisfies $I_H(\mathcal{P}^*) = (1 + \frac{1}{2\lambda} - \delta) I_H(\mathcal{P}^0)$ for any $\delta > 0$.

The setting we are considering for the proof is shown in Figure 4.2. There are $2 \cdot \mu$ points, where the initial population is set to $\mathcal{P}^0 = \{s_0, ..., s_{\mu-1}\}$ and the optimal population would be $\mathcal{P}^* = \{o_0, ..., o_{\mu-1}\}$. We consider a setting with multiple reference points $\{r_0, ..., r_{2\mu-2}\}$, such that the areas contributing to the hypervolume calculation are A_i (areas only dominated by the initial population), B_i (areas only dominated by the optimal population), and C_i and D_i (areas dominated by one solution of the initial population and one solution of the optimal population), see Figure 4.2. The objective space is the union of all points, i.e. $\mathcal{Y} = \mathcal{P}^0 \cup \mathcal{P}^*$.

In our example, we set these areas as follows, assuming $\lambda < \mu$:

$$A_i = \varepsilon \text{ for } 0 \leq i < \mu \quad , \quad B_i = \begin{cases} \varepsilon & \text{for } 0 \leq i < \lambda \\ 1 & \text{for } \lambda \leq i < \mu \end{cases}$$

$$C_i = \sum_{i-\lambda \leq j < i} B_{j \bmod \mu} \quad , \quad D_i = \sum_{i+1 \leq j < i+1+\lambda} B_{j \bmod \mu}$$

Note that for any choice of areas A_i, B_i, C_i, and D_i, corresponding coordinates can be found for all s_i and o_i and r_i by using the following recursions:

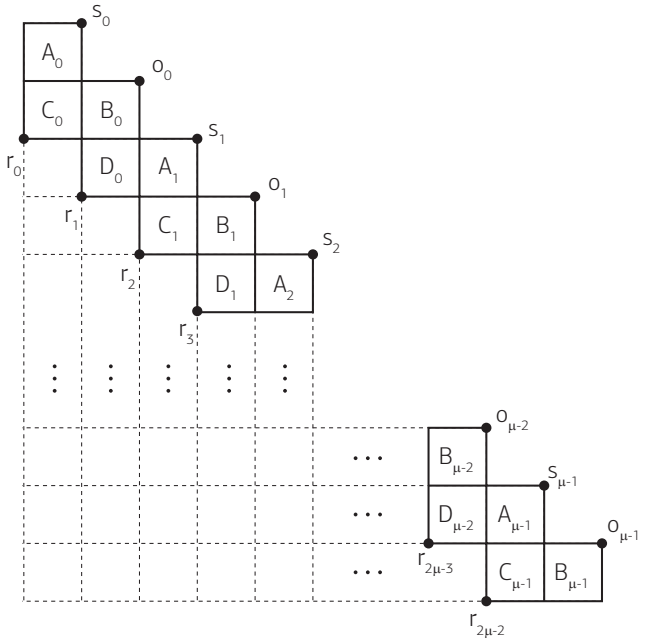


Figure 4.2 Schematic drawing of the example setting in the proof of Theorem 4.13.

$$s_i^x = o_{i-1}^x + \frac{A_i}{s_i^y - o_i^y}, \quad o_i^x = s_i^x + \frac{B_i}{o_i^y - s_{i+1}^y}$$

$$s_i^y = o_{i-1}^y - \frac{C_{i-1}}{s_{i-1}^x - o_{i-2}^x}, \quad o_i^y = s_i^y - \frac{D_{i-1}}{o_{i-1}^x - s_{i-1}^x}$$

$$r_i^x = \begin{cases} o_{i/2-1}^x \; i \text{ even} \\ s_{(i-1)/2-1}^x \; i \text{ odd} \end{cases}, \quad r_i^y = \begin{cases} o_{i/2+1}^y \; i \text{ even} \\ s_{(i-1)/2}^y \; i \text{ odd} \end{cases}$$

where s_i^x, o_i^x, s_i^y, o_i^y and r_i^y, r_i^y are the x-axis and y-axis coordinates of s_i, o_i and r_i, respectively. While s_0^x, s_0^y, and r_0^x with $r_0^x < s_0^x$ can be chosen arbitrarily, the coordinates for o_0^y and s_1^y are set as follows:

$$o_0^y = s_0^y - \frac{A_0}{s_0^x - r_0^x}, \quad s_1^y = o_0^y - \frac{C_0}{s_0^x - r_0^x}$$

Furthermore, $r_{2\mu-2}^y$ and $o_{\mu-1}^x$ are set as follows:

$$r_{2\mu-2}^y = o_{\mu-1}^y - \frac{C_{\mu-1}}{s_{\mu-1}^x - o_{\mu-2}^x}, \quad o_{\mu-1}^x = s_{\mu-1}^x + \frac{B_{\mu-1}}{o_{\mu-1}^y - r_{2\mu-2}^y}$$

First, we want to show that for the example, $\mathcal{P}^0$ is a local optimum, i.e. $I_H(\mathcal{P}^0)$ cannot be improved by any non-decreasing $(\mu+\lambda)$-archiving algorithm. To do so consider a λ-population $\mathcal{O} \subset \mathcal{Y}$ and a μ-population $\mathcal{P}^1 \subset \mathcal{P}^0 \cup \mathcal{O}$. In order for $\mathcal{P}^0$ to be a local optimum, we have to show that $I_H(\mathcal{P}^0) \geq I_H(\mathcal{P}^1)$.

Note that for the rest of the proof, we will always use the indices modulo μ without writing it explicitly. Put differently, we will write A_i,B_i,C_i, and D_i as a short form of $A_{i \bmod \mu}$,$B_{i \bmod \mu}$,$C_{i \bmod \mu}$, and $D_{i \bmod \mu}$.

The hypervolume of the initial population can be written as $I_H(\mathcal{P}^0) = I_H - \sum_{0 \leq i < \mu} B_i = I_H - (\mu - \lambda) - \lambda\varepsilon$, where I_H is the hypervolume of all solutions, i.e. $I_H = I_H(\mathcal{P}^0 \cup \mathcal{P}^*)$. Similarly, we can write $I_H(\mathcal{P}^1) = I_H - \sum_{i:s_i,o_i \notin P^1} C_i - \sum_{i:s_{i+1},o_i \notin P^1} D_i - \sum_{i:o_i \notin \mathcal{P}^1} B_i - \sum_{i:s_i \notin \mathcal{P}^1} A_i$. Using these expressions, we get the following set of equivalent inequalities:

$$\begin{aligned}
I_H(\mathcal{P}^0) &\geq I_H(\mathcal{P}^1) \\
I_H - (\mu - \lambda) - \lambda\varepsilon &\geq I_H - \textstyle\sum_{i:s_i,o_i \notin P^1} C_i - \sum_{i:s_{i+1},o_i \notin P^1} D_i \\
&\quad - \textstyle\sum_{i:o_i \notin \mathcal{P}^1} B_i - \sum_{i:s_i \notin \mathcal{P}^1} A_i \\
(\mu - \lambda) + \lambda\varepsilon &\leq \textstyle\sum_{i:s_i,o_i \notin P^1} C_i + \sum_{i:s_{i+1},o_i \notin P^1} D_i \\
&\quad + ((\mu - \lambda) + \lambda\varepsilon - \textstyle\sum_{i:o_i \in \mathcal{P}^1} B_i) + \sum_{i:s_i \notin \mathcal{P}^1} A_i
\end{aligned}$$

$$\sum_{i:o_i \in \mathcal{P}^1} B_i \leq \sum_{i:s_i,o_i \notin P^1} C_i + \sum_{i:s_{i+1},o_i \notin P^1} D_i + \sum_{i:s_i \notin \mathcal{P}^1} A_i \tag{4.3}$$

To prove this inequality (4.3), we need to consider all possible μ-populations $\mathcal{P}^1 \subset \mathcal{P}^0 \cup \mathcal{O}$, i.e. the results of all possible $(\mu+\lambda)$-archiving algorithms.

To go from $\mathcal{P}^0$ to $\mathcal{P}^1$, λ solutions s_i of the initial set are discarded and the same number of solutions o_i from the optimal set are added. We call these discarded s_i and added o_i *affected* solutions.

In the following, we consider *blocks* of affected solutions. To this end, we first mark all solutions in $\mathcal{P}^0 \cup \mathcal{P}^*$ that are either removed from $\mathcal{P}^0$ or added to $\mathcal{P}^0$ when going from $\mathcal{P}^0$ to $\mathcal{P}^1$. This set of marked solutions is then partitioned into the minimal number of subsets, such that each subset contains all solutions in index range $[i, i+k]$. Depending on whether the first and last solutions in such a subset are from set $\mathcal{P}^0$ or $\mathcal{P}^*$ we call it an (s,s)-, (s,o)-, (o,s)- or (o,o)-block, respectively. For example, an (o,s)-block with index range $[2,5]$ contains solutions $\{o_2, s_3, o_3, s_4, o_4, s_5\}$. The rationale is that non-neighboring solutions do not influence each other, as they do not dominate any common area. As for the blocks, there are two cases which will be considered separately.

Blocks of even length: There are two types of blocks of even length: Those starting with an added solution from the optimal set, i.e. (o,s)-blocks, and those starting with a discarded solution from the initial set, i.e. (s,o)-blocks. The first case can be formalized as follows: The (o,s)-block with index range $[i, i+k]$ exists iff $(o_l \in \mathcal{P}^1, i \leq l < i+k) \wedge (o_{i+k} \notin \mathcal{P}^1) \wedge (s_i \in \mathcal{P}^1) \wedge (s_l \notin \mathcal{P}^1, i+1 \leq l < i+k+1)$. For this block, (4.3) evaluates to:

$$\begin{aligned}
\textstyle\sum_{i:o_i \in \mathcal{P}^1} B_i &\leq \textstyle\sum_{i:s_i, o_i \notin P^1} C_i + \sum_{i:s_{i+1}, o_i \notin P^1} D_i + \sum_{i:s_i \notin \mathcal{P}^1} A_i \\
\textstyle\sum_{i \leq l < i+k} B_l &\leq C_{i+k} + 0 + \textstyle\sum_{i+1 \leq l < i+k+1} A_l \\
\textstyle\sum_{i \leq l < i+k} B_l &\leq \textstyle\sum_{i+k-\lambda \leq l < i+k} B_l + k\varepsilon \\
0 &\leq \textstyle\sum_{i+k-\lambda \leq l < i} B_l + k\varepsilon
\end{aligned}$$

The last step is true because we know that $k \leq \lambda$. As all B_l as well as ε are larger than zero, (4.3) holds.

The second case can be formalized as follows: The (s,o)-block with index range $[i, i+k]$ exists iff $(o_{i-1} \notin \mathcal{P}^1) \wedge (o_l \in \mathcal{P}^1, i \leq l < i+k) \wedge (s_l \notin \mathcal{P}^1, i \leq l < i+k) \wedge (s_{i+k} \in \mathcal{P}^1)$. For this block, (4.3) evaluates to:

$$\begin{aligned}
\textstyle\sum_{i:o_i\in\mathcal{P}^1} B_i &\leq \textstyle\sum_{i:s_i,o_i\notin P^1} C_i + \sum_{i:s_{i+1},o_i\notin P^1} D_i + \sum_{i:s_i\notin\mathcal{P}^1} A_i \\
\textstyle\sum_{i\leq l<i+k} B_l &\leq 0 + D_{i-1} + \textstyle\sum_{i\leq l<i+k} A_l \\
\textstyle\sum_{i\leq l<i+k} B_l &\leq \textstyle\sum_{i\leq l<i+\lambda} B_l + k\varepsilon \\
0 &\leq \textstyle\sum_{i+k\leq l<i+\lambda} B_l + k\varepsilon
\end{aligned}$$

Again, we can see that the last inequality holds, and therefore, (4.3) holds.

Blocks of odd length: Such blocks consist of either a set of discarded solutions that enclose a set of added solutions or vice versa, i.e. (s,s)- or (o,o)-blocks. Due to $|\mathcal{P}^0| = |\mathcal{P}^1|$, the number of added solutions from the optimal set must be equal to the number of discarded solutions from the initial set. Directly following this, we know that for each block of discarded solutions enclosing added solutions, there must be another block of added solutions enclosing discarded solutions and vice versa. These two types of blocks can be formalized as follows: The (s,s)-block with index range $[i,i+k]$ exists iff $(o_l \in \mathcal{P}^1, i \leq l < i+k-1) \wedge (o_{i-1}, o_{i+k-1} \notin \mathcal{P}^1) \wedge (s_l \notin \mathcal{P}^1, i \leq l < i+k)$. The (o,o)-block with index range $[j,j+p]$ exists iff $(o_l \in \mathcal{P}^1, j \leq l < j+p) \wedge (s_l \notin \mathcal{P}^1, j+1 \leq l < j+p) \wedge (s_j, s_{j+p} \in \mathcal{P}^1)$. Also, we know that $1 \leq k, p \leq \lambda$ and $k+p \leq \lambda+1$. Considering both of these blocks, (4.3) evaluates to:

$$\begin{aligned}
\textstyle\sum_{i:o_i\in\mathcal{P}^1} B_i \leq\ & \textstyle\sum_{i:s_i,o_i\notin P^1} C_i + \sum_{i:s_{i+1},o_i\notin P^1} D_i \\
& + \textstyle\sum_{i:s_i\notin\mathcal{P}^1} A_i \\
\textstyle\sum_{i\leq l<i+k-1} B_l + \sum_{j\leq l<j+p} B_l \leq\ & C_{i+k-1} + D_{i-1} + \textstyle\sum_{i\leq l<i+k} A_l \\
& + \textstyle\sum_{j+1\leq l<j+p} A_l \\
\textstyle\sum_{i\leq l<i+k-1} B_l + \sum_{j\leq l<j+p} B_l \leq\ & \textstyle\sum_{i+k-1-\lambda\leq l<i+k-1} B_l + \sum_{i\leq l<i+\lambda} B_l \\
& + (k+p-1)\varepsilon \\
\textstyle\sum_{j\leq l<j+p} B_l \leq\ & p \leq \textstyle\sum_{i+k-1-\lambda\leq l<i+\lambda} B_l + (k+p-1)\varepsilon \\
p \leq\ & \lambda\varepsilon + \lambda - k + 1 + (k+p-1)\varepsilon
\end{aligned}$$

The second last step can be done because we know that at most λ of the B_l's are set to ε and therefore, at least $\lambda - k + 1 \geq p$ of the B_l's remain which are set to 1. Also, because of $p \leq \lambda - k + 1$, the last inequality holds and with it (4.3) holds.

Combinations of blocks: As stated before, only neighboring solutions in $\mathcal{Y} = \mathcal{P}^0 \cup \mathcal{P}^*$ share a common dominated area. From the definition of the different types of blocks it can be seen that there are no adjacent blocks, because in this case, the two blocks would be combined into one. Therefore, each pair of blocks is separated by at least one solution from $\mathcal{Y}$ which is not affected by the transition from $\mathcal{P}^0$ to $\mathcal{P}^1$. As a result, the changes in hypervolume when going from $\mathcal{P}^0$ to $\mathcal{P}^1$ can be considered separately for each block. We have shown that for any block, (4.3) holds. From this we can conclude that $I_H(\mathcal{P}^0) \geq I_H(\mathcal{P}^1)$ and therefore, $\mathcal{P}^0$ is a local optimum.

Now that we've done the first part of the proof, i.e. showing that any non-decreasing $(\mu+\lambda)$-archiving algorithm will not be able to escape from $\mathcal{P}^0$, we would like to calculate how far the hypervolume of $\mathcal{P}^0$ is from the maximum achievable hypervolume. In other words, we would like to calculate $\frac{I_H(\mathcal{P}^*)}{I_H(\mathcal{P}^0)}$. The hypervolume of the initial population evaluates to:

$$\begin{aligned} I_H(\mathcal{P}^0) &= \textstyle\sum_{0\leq l<\mu} C_l + \sum_{0\leq l<\mu} D_l + \sum_{0\leq l<\mu} A_l \\ &= \textstyle\sum_{0\leq l<\mu} \left(\sum_{l-\lambda\leq j<l} B_j + \sum_{l+1\leq j<l+1+\lambda} B_j \right) + \mu\varepsilon \\ &= \textstyle\sum_{0\leq l<\mu} \left(\sum_{l-\lambda\leq j<l+1+\lambda} B_j - B_l \right) + \mu\varepsilon \\ &= (2\lambda+1) \textstyle\sum_{0\leq l<\mu} B_l - \sum_{0\leq l<\mu} B_l + \mu\varepsilon \\ &= 2\lambda \textstyle\sum_{0\leq l<\mu} B_l + \mu\varepsilon \end{aligned}$$

The hypervolume of the optimal population, on the other hand, can be calculated as follows:

$$\begin{aligned} I_H(\mathcal{P}^*) &= \textstyle\sum_{0\leq l<\mu} C_l + \sum_{0\leq l<\mu} D_l + \sum_{0\leq l<\mu} B_l \\ &= \textstyle\sum_{0\leq l<\mu} \sum_{l-\lambda\leq j<l+1+\lambda} B_j \\ &= (2\lambda+1) \textstyle\sum_{0\leq l<\mu} B_l \end{aligned}$$

Both sets of equations make use of $\sum_{0\leq l<\mu} \sum_{l-\lambda\leq j<l+1+\lambda} B_j = (2\lambda+1) \sum_{0\leq l<\mu} B_l$. This is due to the fact that the inner sum of the left-hand term consists of $2\lambda+1$ summands. Because all indices are taken modulo μ, we see that each B_j is summed up $2\lambda+1$ times in the whole term.

Finally, this leads us to the following result, which holds for any $\delta > 0$ if $\varepsilon \to 0$ and $\lambda < \mu$:

$$\begin{aligned}\frac{I_H(\mathcal{P}^*)}{I_H(\mathcal{P}^0)} &= \frac{(2\lambda+1)\sum_{0\leq l<\mu} B_l}{2\lambda\sum_{0\leq l<\mu} B_l+\mu\varepsilon} \\ &= 1 + \tfrac{1}{2\lambda} - \delta\end{aligned}$$

Note that in the case of $\lambda = \mu$, the equation evaluates to $\frac{I_H(\mathcal{P}^*)}{I_H(\mathcal{P}^0)} = 1$, which is very natural, since for $\mu = \lambda$, any non-decreasing $(\mu+\lambda)$-archiving algorithm is effective. □

We may also interpret the above result in terms of the more practical interchange heuristic shown in Algorithm 16. One can conclude that for $z(\mathcal{P}) = I_H(\mathcal{P})$, i.e. we use the hypervolume indicator for archiving, we may end up with a solution that is not better than $1/\alpha$ times the optimal hypervolume with $\alpha > 1 + \frac{1}{2\lambda}$, even after an unlimited number of iterations.

4.5 · Summary

In this chapter, we investigated the α-approximate effectiveness of $(\mu+\lambda)$-EAs that optimize the hypervolume. The value of α gives a lower bound on the hypervolume which can always be achieved, independent of the objective space and the chosen initial population. While it is obvious that for $\mu = \lambda$, α is equal to 1, Bringmann and Friedrich have shown that for $\lambda = 1$, α is equal to 2. This chapter strictly improves the currently known bound and finds that for arbitrary λ, the approximation factor α is equal to $2 - \frac{\lambda-p}{\mu}$, where $\mu = q \cdot \lambda - p$ and $0 \leq p \leq \lambda - 1$.

Furthermore, we improve the available lower bound on α for the general definition of the hypervolume indicator, i.e. $\alpha > 1 + \frac{1}{2\lambda}$. Upper and lower bounds only match for a population size of $\mu = 2$. It might be possible to further tighten the lower bound by extending the worst case construction in the proof of Theorem 4.13 to higher dimensions of the objective space.

5

Conclusions

Optimization problems can either be single-objective or multi-objective . In the case of a single-objective problem, there is one solution or possibly a set of solutions that has the best objective value, whereas in the case of multi-objective problems with conflicting objectives, there is no single best solution, but a set of tradeoff solutions, the so-called Pareto-front. Usually, the goal of optimizing algorithms is to find one or all of the best solutions in single-objective problems, and the whole Pareto-front or a representative subset of it in multi-objective problems.

This thesis considers the case that there are some uncertainties or simplifications in the optimization model, which make close-to-optimal solutions also interesting for the user who optimizes the problem. Moreover, we assume that the user is interested in a set of structurally diverse, close-to-optimal solutions. The present thesis (a) explores ways to generate such sets of solutions, and (b) provides methods to analyze the resulting sets of solutions.

Finally, this thesis investigates the properties of the hypervolume indicator, which is a contemporary measure to quantify the quality of a set of solutions

in terms of its objective values in multi-objective problems. The hypervolume indicator is used in all parts of this thesis, both in the algorithms that generate diverse solutions for multi-objective problems and in one of the algorithms that helps to analyze the achieved set. We investigate the effectiveness of this indicator, which states whether an algorithm can reach a set of solutions of a certain size with the best hypervolume.

5.1 · Key Results

The present thesis makes the following three main contributions.

5.1.1 · Finding Structurally Diverse Close-To-Optimal Sets of Solutions

We propose three evolutionary algorithms to find structurally close-to-optimal sets of solutions. The first algorithm, NOAH, aims at single-objective problems, where the user can specify a certain *barrier* value v, where all solutions that are better than v are acceptable to the user. NOAH then finds a set of solutions of high diversity, where each solution has an objective value which is better or equal to v. The second algorithm, DIOP, extends the idea of NOAH to multi-objective problems. It finds a maximally diverse set of solutions that lie within a user-defined distance of an approximation of the Pareto-optimal front. Finally, the third algorithm, DIVA, integrates the diversity information into the hypervolume indicator, and finds a set of solutions which maximizes this adapted indicator.

5.1.2 · Analyzing Given Sets of Solutions

Once a set of solutions is returned by the optimizer, the user has to analyze it in order to pick a preferred solution or learn about the problem. This thesis focuses on the latter, and proposes two algorithms to analyze a given set in terms of what types of solutions lead to what objective values. The first algorithm, MANA, tackles biobjective problems with binary decision spaces. It finds so-called modules of decision variables, i.e. sets of decision variables that are set to 1 in as many solutions as possible. The output is

a hierarchical clustering, where on each level of the hierarchy, the clusters contain only neighboring solutions in objective space, and can be annotated with the modules that are contained in the solutions of this cluster. The second algorithm, PAN, is aimed at problems with 2 or more objectives, and with arbitrary decision spaces. It solves the biobjective problem of clustering solutions both in decision and objective space. The output is a partitioning of the solutions into clusters, where solutions within the clusters have low pairwise distances both in objective and decision space, and solutions in different clusters have a high pairwise distance both in objective and decision space.

5.1.3 · Bounding the Effectiveness of the Hypervolume Indicator

Evolutionary Algorithms (EAs) usually have a fixed population size. Instead of finding the whole Pareto-front, they aim at finding a good subset of that front. One way to do so is to pick the subset with the best hypervolume. This thesis investigates the effectiveness of $(\mu+\lambda)$-EAs. An algorithm is effective if for any optimization problem, and for any initial population, there is a sequence of offspring such that the algorithm can reach the set of solutions with the best hypervolume. If such a sequence of offspring does not exist, the set with the best hypervolume cannot be reached, and the question arises how far the best achievable hypervolume is from the optimal hypervolume. We present both upper and lower bounds on this achievable hypervolume which are tighter than the bounds previously known in the literature.

5.2 · Discussion and Future Work

This thesis showed that explicitly maintaining structural diversity during optimization can lead to considerably more diverse, but still close-to-optimal solutions. However, some questions still remain unanswered, which might be interesting to discuss in the future. The first question concerns the diversity measure itself, and the requirements the measure should fulfill. We listed

three requirements, originally stated by Solow and Polasky [98], that a diversity measure should satisfy. These three requirements are that (a) diversity should increase if a solution is added to the set, (b) adding duplicates should not change diversity, and (c) increasing the distance between two or more solutions should increase the diversity. It is not clear whether these three requirements are sufficient. For example they make no statement about what type of distribution of points is preferred, or what the optimal distribution of points would be. Moreover, it is not clear whether requirements (a) and (b) are necessary or wanted. These two requirements indicate that the diversity should measure how well the points cover a given space. Adding a duplicate does not increase coverage, and adding a new solution never decreases coverage. The problem is that with these requirements, only sets with the same number of solutions can be compared fairly, as the diversity can only increase by adding solutions. For comparing sets of solutions with different numbers of solutions, the measure should decrease when adding duplicates and solutions very similar to solutions already present in the set. Even if it is decided that the three requirements are what is wanted, a measure needs to be found that satisfies these requirements. Up to date the author is not aware of any measure that fulfills all requirements for general decision spaces and general distance metrics.

For analyzing sets of solutions, more information about the problem may be included, as this would enable algorithms to automatically extract problem specific design principles. Also, interactive methods could be developed that guide the user through the set of solutions, where the user can focus on the area he or she is most interested in.

Finally the work on the hypervolume effectiveness may be extended to find tight bounds for populations with more than two solutions. Also, a different scenario could be considered. In this thesis, we assume a best case view on the offspring generation, which in reality has a very low probability of occurring. Therefore it would be interesting to investigate average offspring sequences, or at least give some indication of the probability that an offspring sequence might occur in practice.

Appendix

A · Reference Algorithm: Greedy Hypervolume Selection

In this section we introduce the Standard Multi-objective Evolutionary Algorithm (sMOEA) that is needed in order to quantify the results of the proposed diversity-optimizing multi-objective evolutionary algorithms DIOP and DIVA. Since in this thesis we use the hypervolume indicator as a measure of set quality in objective space, we propose to use an algorithm that explicitly optimizes the hypervolume.

The framework of the sMOEA is given in Algorithm 17. We want to compare selection strategies, therefore, the problem-dependent variation procedure will be the same for all compared algorithms. For selection, several things have to be taken into account. First, it is possible that the population is a multiset, i.e. it contains duplicate solutions. Note that solutions are unique in decision space, but not necessarily in objective space, as two different solution can map to the same objective values. So the term *duplicate* means that two solutions are equal in decision space. We assume that two dissimilar solutions $a, b \in \mathcal{X}$ are always better than two duplicates $a, a \in \mathcal{X}$, i.e. duplicates will only be selected if only duplicates are left to select from. One way to achieve this is using the method described in Algorithm 18.

Second, any selection strategy should respect the given preference relation, i.e. Pareto dominance. This means that a dominated solution should only be selected if its dom-

```
1: function sMOEA(μ, λ, R, g)
2:     Initialize population P^1 randomly with μ solutions
3:     i = 1
4:     for g generations do
5:         O^i := VARIATE(P^i, λ)
6:         P^{i+1} := SELECT(P^i ∪ O^i, μ)
7:                                          (Only take new population if not worse)
8:         if I_H(P^{i+1}, R) < I_H(P^i, R) then
9:             P^{i+1} := P^i
10:        i = i + 1
11:    return P^i
```

Algorithm 17 Framework of sMOEA. Input parameters: population size μ, offspring size λ, reference set $\mathcal{R}$; minimization of objective functions is done for g generations

function DuplicateSelection($\mathcal{P}$, s)
  define $\mathcal{R}$ to be the underlying set of elements of the multiset $\mathcal{P}$
  define $m : \mathcal{R} \to \mathbb{N}_{\geq 1}$ as a function giving the multiplicity of each element in $r \in \mathcal{R}$, i.e. $m(r)$ is the number of times r appears in $\mathcal{P}$
  generate duplicate sets $\{\mathcal{D}_1, ..., \mathcal{D}_n\}$, where $\mathcal{D}_i = \{r \in \mathcal{R} : m(r) \geq i\}$
  $\mathcal{S} = \emptyset$
  $i = 1$
  while $|\mathcal{S} \cup \mathcal{D}_i| \leq s$ **do**
    $\mathcal{S} = \mathcal{S} \cup D_i$ (multiset union)
    $i = i + 1$
  $\mathcal{S}' = \mathcal{D}_i$
  $s' = s - |\mathcal{S}|$
  return $\mathcal{S}, \mathcal{S}', s'$

Algorithm 18 Duplicate selection. Input parameters: population of n solutions $\mathcal{P}$, number of solutions to select s. Returns selected solutions $\mathcal{S}$, solutions to select from $\mathcal{S}'$ and number of remaining solutions to select s'.

function NondominatedSortingSelection($\mathcal{P}$, s)
  extract nondominated fronts $\{\mathcal{F}_1, ..., \mathcal{F}_n\}$, where $\mathcal{F}_i = \{p \in \mathcal{P} : \nexists y \in \mathcal{P} \setminus \cup_{j=1}^{i-1} \mathcal{F}_i \text{ s.t. } y \prec p\}$
  $\mathcal{S} = \emptyset$
  $i = 1$
  while $|\mathcal{S} \cup \mathcal{F}_i| \leq s$ **do**
    $\mathcal{S} = \mathcal{S} \cup \mathcal{F}_i$
    $i = i + 1$
  $\mathcal{S}' = \mathcal{F}_i$
  $s' = s - |\mathcal{S}|$
  return $\mathcal{S}, \mathcal{S}', s'$

Algorithm 19 Nondominated sorting selection. Input parameters: population of n solutions $\mathcal{P}$, number of solutions to select s. Returns selected solutions $\mathcal{S}$, solutions to select from $\mathcal{S}'$ and number of remaining solutions to select s'.

inating solutions are selected as well. To achieve this, the well-known nondominated sorting strategy [29] as described in Algorithm 19 can be used.

Finally, an ideal hypervolume-optimizing selection strategy would return the subset $\mathcal{P}'$ of size s which has the maximum hypervolume, i.e. $\mathcal{P}' = \text{argmax}_{\mathcal{P}'' \subseteq \mathcal{P}, |\mathcal{P}''| = s} I_H(\mathcal{P}'')$. Unfortunately, there is no easy way to determine this best set $\mathcal{P}'$. Also, testing all possible subsets of size s is infeasible due to combinatorial explosion. One common way

1: **function** GreedyHypervolumeSelection($\mathcal{P}$, s)
2: **while** $|\mathcal{P}| > s$ **do**
3: $\forall p_i \in \mathcal{P} : c(p_i, \mathcal{P}) = I_H(\mathcal{P}) - I_H(\mathcal{P} \backslash \{p_i\})$
4: $p_{min} = \text{argmin}_{p_i \in \mathcal{P}}\, c(p_i, \mathcal{P})$
5: $\mathcal{P} := \mathcal{P} \backslash \{p_{min}\}$
6: **return** $\mathcal{P}$

Algorithm 20 Greedy hypervolume selection strategy. Input parameters: population of n solutions $\mathcal{P}$, number of solutions to select s

1: **function** select($\mathcal{P}$, μ)
2: $\{\mathcal{P}', \mathcal{S}', s'\} =$DupcliateSelection$(\mathcal{P}, \mu)$
3: $\{\mathcal{P}'', \mathcal{S}'', s''\} =$NondominatedSortingSelection$(\mathcal{S}', s')$
4: $\mathcal{P}''' =$GreedyHypervolumeSelection$(\mathcal{S}'', s'')$
5: **return** $\mathcal{P}' \cup \mathcal{P}'' \cup \mathcal{P}'''$

Algorithm 21 Reference selection strategy. Input parameters: population of $\mu + \lambda$ solutions $\mathcal{P}$, number of solutions to select μ

to deal with this is to iteratively throw away the solution with the lowest contribution to the hypervolume, which is also called greedy hypervolume selection, see Algorithm 20.

Using these methods, we can finally describe the reference environmental selection scheme used in this thesis, see Algorithm 21. It first handles the duplicates and extracts a subset containing only dissimilar solutions. Second, it prefers non-dominated solutions over dominated ones by using non-dominated sorting, which returns a subset only containing non-dominated solutions. Finally, to select from this set of non-dominated solutions, the greedy hypervolume selection strategy is used. Note that because a greedy selection scheme is used, it is possible that the hypervolume of the new population is lower than the hypervolume of the parent population. In this case, the new population is discarded and the parent population is used instead.

B · Bridge Optimization Problem

This thesis uses a bridge construction problem as a real-world optimization problem. This problem is well suited as a test problem because (a) bridges are easy to visualize, (b) bridge evaluation is fast and (c) the decision space of possible bridges is complex and large. The bridge construction problem considered in this thesis is inspired by [3],

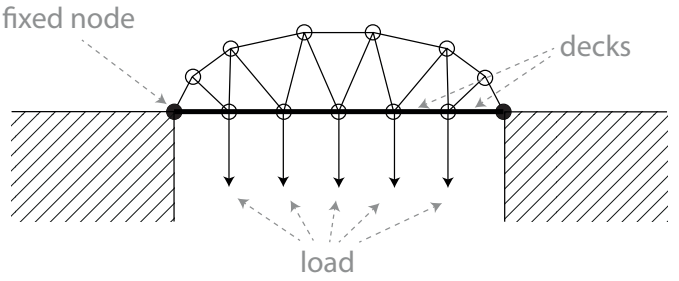


Figure B.1 Example bridge. One of the two fixed nodes, the five applied loads and two of the six decks are indicated using dotted arrows.

where the goal is to build a truss bridge that can carry a fixed load. An example bridge can be seen in Figure B.1. Each bridge basically is a set of nodes, with connections between certain node pairs. All bridges have to be built in the following framework: First, there are 2 fixed nodes, shown as black circles in the figure, to which the bridge is connected. Note that in accordance with standard truss analysis, the fixed node on the left side of the bridge is fixed both in horizontal and vertical direction, whereas the fixed node on the right side of the bridge is fixed only in the vertical direction. Each bridge has a set of 6 horizontal connections, called the decks, over which the traffic goes. The traffic is modeled as a fixed load which is applied to the five non-fixed nodes between these decks, shown as arrows in the figure. In this thesis we assume that a good bridge will be symmetric. Therefore we reduce the search space to symmetric bridges only, i.e. bridges whose left half is identical to the mirrored right half of the bridge. We do so by starting with randomized symmetric bridges and then ensuring that each change we apply to the bridge is also mirrored to the other side.

The optimization algorithm needs to be able to modify existing bridges in order to create new bridges from old ones. So how do we represent and modify existing bridges? We use a so-called direct representation, i.e. the bridges are directly stored as a set of nodes, and a list of pairs of nodes between which there is a connection. To create new bridges from existing ones, each bridge can be modified through mutation, or two bridges can be recombined. To do mutation, either the nodes or the connections of the bridge can be modified. If a node is modified, three elementary operations can be made: a node can be added, removed, or moved. If a node is removed, the node is deleted from the node list and all connections to or from that node are also deleted. If a node is added, an existing connection (not the decks though) is randomly selected and split by adding a node

somewhere in between the end nodes of the connection, removing the old connection and then reconnecting both end nodes to the newly inserted node by inserting two new connections. To move a node, a random node is selected and moved both in horizontal and vertical direction by adding a random number distributed according to a two dimensional Gaussian distribution. Modifying connections is straightforward. Either a random existing connection (except the decks) is selected and removed, or a connection is added between two nodes which have not yet been connected.

To recombine two parent bridges, we use an adaptation of one-point crossover which is illustrated in Figure B.2. Note that because bridges are symmetric, the one-point crossover is actually a two-point crossover with mirrored cut points. First, a cut position is chosen randomly, shown as a vertical line. A second cut is calculated by mirroring the first cut. Both parent bridges are cut at those two positions, and the parts between the cuts are swapped in order to generate two offspring. Hence the first offspring bridge for example consists of the outer part of the first parent bridge and the inner part the second parent bridge. Now, there might have been certain connections in both parent bridges which have been destroyed by the cutting. For each connection that was cut, one end node is retained in the offspring, whereas the other end node is not there anymore. To repair such a connection, all available nodes in the offspring are considered and the one node which is closest to the removed end node (the one which is not available anymore in the offspring) will be used as the new end node of the connection.

The optimization algorithm also needs to be able to create random bridges to generate the initial population. As randomly generating nodes and connecting them in a random manner is likely to lead to unstable bridges, we propose the following approach: We always start with a (stable) Warren truss, and then we randomly move the nodes of the bridge's top horizontal connections in order to introduce some variation. A warren truss (left) and a random bridge generated from it (right) are shown in the upper row of Figure B.2. If a random bridge is unstable, new bridges are generated repeatedly until a stable one is found.

Bridges are evaluated according to two criteria: weight and the length of the longest connection. We assume that nodes are weight-free, and the total weight is solely determined by the weights of the connections. We chose the bridge weight as the first objective because under a few assumptions, the weight relates linearly to the cost of the bridge through the material cost. These assumptions are that there are no additional cost for nodes, and no fixed cost for each connection. The second objective is the length of the longest connections. We chose this objective because in a real-world scenario, long connections might be more difficult to transport than short connections, and they may be difficult to produce.

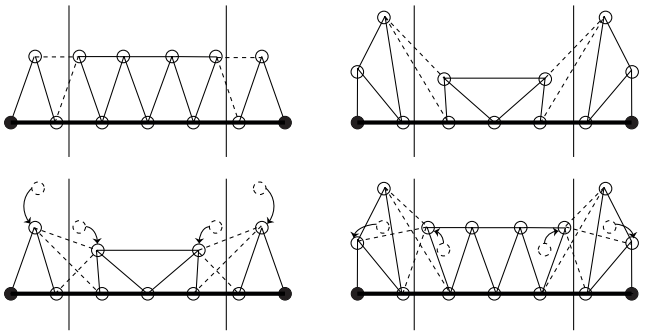


Figure B.2 Example for the recombination of two parent bridges (upper row). Cuts are shown as vertical lines. Connections that will be destroyed by the cut are shown as dashed lines in the parents. For both offspring (bottom row), four connections have been cut and need to be reinserted. The corresponding inserted connections are shown as dashed lines in the children. Also, the original nodes (dashed circles) are shown, as well as the offspring nodes to which they are closest (indicated by arrows).

The weight of the bridge is calculated as follows: First, it is checked whether the bridge is stable. To do so, we use an approach presented in [85].[1] If the bridge is not stable, it is discarded. If it is stable, the force on each connection is calculated. Then, the minimum diameter of each connection is calculated. It is chosen such that the connection can withstand the force applied to it, a decision which only depends on the material's yield strength. Note that the forces on the connections can only be calculated if the truss is statically determinate, meaning the equation $2 \cdot j = m + r$ holds, where j is the number of nodes, m is the number of connections and r is the number of reaction components. As this cannot be guaranteed in our algorithms, a method which is called the force method [85] is used. The force method allows deflection of the connections, which makes it possible to calculate the forces of a statically indeterminate truss. To apply the force method, knowledge of the connection diameters is necessary. As the diameters of the bridges in turn depend on the forces, we decided to calculate the forces with each connection diameter set to 1m^2, then to calculate the true connection area, and then recalculate the forces of the bridge with the new diameters. If the bridge is stable with the connection diameters of 1m^2, but unstable with the true connection areas, it is treated as unstable and discarded. Also, if the forces with the true diameters deviates

[1] Rahami's Matlab code which we used in this thesis is available on *http://www.mathworks.com/matlabcentral/fileexchange/14313-truss-analysis*

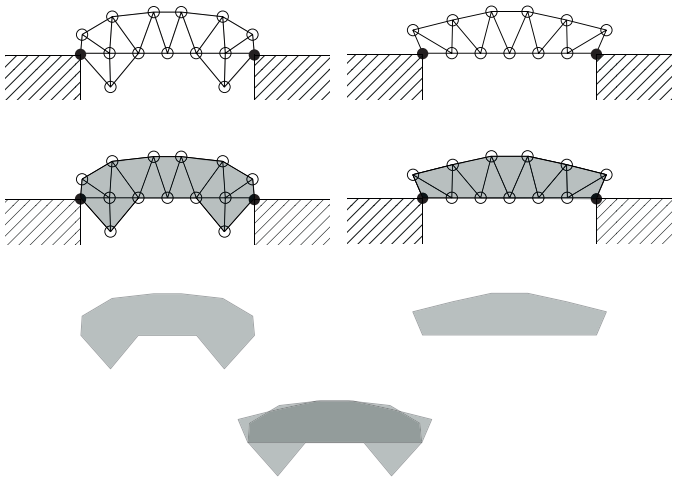


Figure B.3 Distance calculation between two bridges (top row). Bridge areas are shown in gray. The distance is visualized in the bottom row, as the lighter gray area.

more than 1% from the forces of the $1m^2$ diameters, the bridge is treated as unstable and discarded. Once the diameters of the connections have been calculated, the weight of the connection can be calculated using this diameter, the length of the connection and the density of the chosen material.

As mentioned in Section 1.1, a distance measure in decision space is needed, such that similar looking bridges have a low distance and dissimilar looking bridges have a large distance. But how can the distance between two bridges be measured? We decided to go for a visual measure, based on the shape of each bridge. We define the shape of a bridge as the area enclosed by its outermost connections, see Figure B.3 for an example. The area difference between the shapes of two bridges then is the distance between the two bridges.

To generate a set of optimized bridges we used the following specifications: The bridges must support 6 decks, where each deck has a length of 10 meters. Therefore the bridge has to cover a distance of 60 meters. For the load we assume that the bridge must be able to carry two 40 ton trucks, a load of 80 tons therefore is applied to each node between the decks. The bridge has to use steel as a material, with a yield strength of 400 MPa, an elasticity (young's modulus) of 300 GPa and a density of 7.8 g/cm^3. For the variation process in the evolutionary algorithm, we use a recombination probability

of 0.5 and a mutation probability of 1.0, and during mutation, we randomly select with equal probability one of the elementary mutations, i.e. add a connection, remove a connection, add a node, remove a node or move a node. When adding a node, as explained before, an existing connection is removed and replaced by a new node which is connected to the end nodes of the removed connection. The location of the inserted node is chosen randomly in the rectangle area spanned by the end nodes of the removed connection. When moving a node, a Gaussian is added in each dimension with mean zero and standard deviation 3. Whenever infeasible bridges are generated during mutation, we use a repeat strategy that repeatedly tries to do the selected elementary mutation on the parent until either a maximal number of tries, in our case 100, is reached (in which case the parent is returned) or a feasible bridge is found. Furthermore we use a repair strategy prior to the distance calculation which iteratively removes all connections on which there is no force, and all nodes which are the end nodes of less than two connections.

Note that this bridge problem has many similarities to the one proposed in [3]. One difference is that the load is a constraint that determines the connection thickness, where in [3] the load was an objective and the thickness was an optimization parameter. A second difference is that in [3] connections are not allowed to cross the middle of the bridge, as only one half of the bridge was stored, and this half was simply mirrored to the other side to create the whole bridge. In this thesis, the whole bridge is stored, therefore connections can go from the left half to the right half, as long as they have a mirrored counterpart. Also, while [3] only used mutation, this thesis proposes a recombination operator to be used on the bridges.

C · Singular Matrix for Solow-Polasky Diversity Measure

The matrix shown in Figure C.1 is a distance matrix we encountered during a run of the E/E-architecture optimization problem (see Section 1). The distance values are integers, because the measure has to quantify the difference between two partitionings, and is related to the number of differently clustered elements.

Note that there is no violation of the triangle inequality in this matrix. Remember that the Solow-Polasky should yield a value between 1 and the number of solutions, i.e. 17 in this case, which indicates the number of different species found in the population. However, using $\theta = 0.15$, the normalized distance matrix M is close to singular, and the value of the Solow-Polasky measure is 1442.4. Note that the value of the Solow-Polasky measure depends on the value of θ. Figure C.2 shows the influence of θ on the measure,

0	10	9	8	8	9	5	7	4	10	7	7	4	10	6	10	7
10	0	14	11	8	13	12	14	8	3	12	10	10	10	6	7	3
9	14	0	9	11	6	4	2	11	11	4	10	7	7	11	10	13
8	11	9	0	12	11	11	11	8	10	7	7	10	4	6	7	11
8	8	11	12	0	11	11	11	10	8	9	7	4	10	8	10	5
9	13	6	11	11	0	10	8	11	13	10	4	7	7	13	7	11
5	12	4	11	11	10	0	6	9	12	8	6	9	9	9	9	9
7	14	2	11	11	8	6	0	11	13	6	8	9	9	13	11	11
4	8	11	8	10	11	9	11	0	8	7	11	8	9	2	9	8
10	3	11	10	8	13	12	13	8	0	9	10	10	7	6	7	3
7	12	4	7	9	10	8	6	7	9	0	6	9	9	9	9	9
7	10	10	7	7	4	6	8	11	10	6	0	3	3	10	3	7
4	10	7	10	4	7	9	9	8	10	9	3	0	6	7	6	7
10	10	7	4	10	7	9	9	9	7	9	3	6	0	7	3	10
6	6	11	6	8	13	9	13	2	6	9	10	7	7	0	7	6
10	7	10	7	10	7	9	11	9	7	9	3	6	3	7	0	10
7	3	13	11	5	11	9	11	8	3	9	7	7	10	6	10	0

Figure C.1 Distance matrix for 17 solutions of the E/E-architecture problem. The element in the i-th row and the j-th column is the distance between the i-th and the j-th solution.

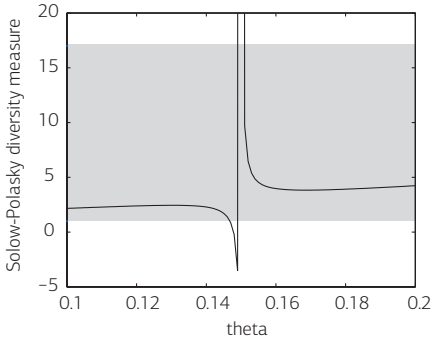


Figure C.2 Change of the Solow-Polasky measure calculated on the above distance matrix with different values of θ. The gray area indicates the expected range (between 1 and 17 for the given distance matrix) of values for the Solow-Polasky measure.

around the critical value of $\theta = 0.15$. Note that the critical range of θ, i.e. where the diversity measure is smaller than 1 or larger than 17, is quite narrow, e.g. choosing $\theta = 0.14$ or $\theta = 0.16$, again leads to a measure value in the expected area, i.e. between 1 and 17.

Bibliography

1 T. Aittokoski, S. Ayramo, and K. Miettinen. Clustering aided approach for decision making in computationally expensive multiobjective optimization. *Optimization Methods Software*, 24:157–174, April 2009.

2 E. Altshuler and D. Linden. Design of a wire antenna using a genetic algorithm. *Journal of Electronic Defense*, 20:50–52, 1997.

3 J. Bader. *Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods*. PhD thesis, ETH Zurich, Switzerland, 2010.

4 S. Bandaru and K. Deb. Towards automating the discovery of certain innovative design principles through a clustering-based optimization technique. *Engineering Optimization*, 43:911–941, 2011.

5 S. Bandyopadhyay and U. Maulik. Nonparametric genetic clustering: Comparison of validity indices. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-31-1:120–125, 2001.

6 D. Beasley, D. Bull, and R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1:101–125, 1993.

7 J. C. Bezdek and N. R. Pal. Cluster validation with generalized dunn's indices. In *Conference on Artificial Neural Networks and Expert Systems (ANNES '95)*, 1995.

8 S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa—a platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, 2003.

9 J. Bourgain. On lipschitz embeddings of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

10 H. J. Bremermann, M. Rogson, and S. Salaff. Global properties of evolution processes. In *Natural Automata and Useful Simulations*, pages 3–41. Spartan Books, 1966.

11 K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In *International Symposium on Algorithms and Computation (ISAAC 2008)*, volume 5369 of *LNCS*, pages 436–447, 2008.

12 K. Bringmann and T. Friedrich. Convergence of hypervolume-based archiving algorithms i: Effectiveness. In *Genetic and Evolutionary Computation Conference (GECCO 2011)*, 2011.

13 D. Brockhoff, D. K. Saxena, K. Deb, and E. Zitzler. On handling a large number of objectives a posteriori and during optimization. In *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pages 377–403. Springer, 2007.

14 L. T. Bui, J. Branke, and H. A. Abbass. Multiobjective optimization for dynamic environments. In *Congress on Evolutionary Computation (CEC 2005)*, 2005.

15 P. R. Bushel, R. D. Wolfinger, and G. Gibson. Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes. *BMC Systems Biology*, 1, 2007.

16 T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

17 M. Calonder, S. Bleuler, and E. Zitzler. Module identification from heterogeneous biological data using multiobjective evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN IX)*, 2006.

18 C. H. Chou, M. C. Su, and E. Lai. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7:205–220, 2004.

19 W. J. Conover. *Practical Nonparametric Statistics*. John Wiley, 3rd edition, 1999.

20 S. Das, A. Abraham, and A. Konar. *Metaheuristic Clustering*. Springer, 2009.

21 D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence*, PAMI-1:224–227, 1979.

22 K. A. de Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

23 K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

24 K. Deb and S. Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In *International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-99)*, 1999.

25 K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Third International Conference on Genetic Algorithms*, 1989.

26 K. Deb and A. Srinivasan. Innovization: Innovative design principles through optimization. Technical report, KanGAL, Indian Institute of Technology Kanpur, 2006.

27 K. Deb and A. Srinivasan. Innovization: Innovating design principles through optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2006)*, 2006.

28 K. Deb and S. Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062–1087, 2008.

29 K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature (PPSN VI)*, 2000.

30 J. J. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

31 J. Edmonds. Submodular functions, matroids and certain polyhedra. In *Combinatorial Structures and their Applications*. Gordon and Breach, 1971.

32 A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

33 M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *National Academy of Sciences of the United States of America*, 95(25): 14863–14868, 1998.

34 E. Falkenauer. *Genetic Algorithms and Groupping Problems*. Wiley, 1998.

35 F. R. Gantmacher. *The Theory of Matrices*. Chelsea Publishing Company, 1959.

36 K. J. Gaston and J. I. Spicer. *Biodiversity: An Introduction*. Wiley-Blackwell, 2nd edition, 2004.

37 D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

38 D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Second International Conference on Genetic Algorithms and their Application*, 1987.

39 D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

40 M. Halkidi and M. Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. In *IEEE International Conference on Data Mining*, 2001.

41 M. Halkidi, Michalis M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. In *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2000)*, 2000.

42 M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *ACM SIGMOD Record*, 31:19–27, 2002.

43 J. Handl. Multiobjective clustering around medoids. In *Congress on Evolutionary Computation (CEC 2005)*, 2005.

44 J. Handl and J. Knowles. Improvements to the scalability of multiobjective clustering. In *Congress on Evolutionary Computation (CEC 2005)*, 2005.

45 J. Handl and J. Knowles. Exploiting the trade-off - the benefits of multiple objectives in data clustering. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, 2005.

46 J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11:56–76, 2007.

47 B. Hardung. *Optimisation of the Allocation of Functions in Vehicle Networks*. PhD thesis, Universität Erlangen, 2006.

48 G. Harik. Finding multimodal solutions using restricted tournament selection. In *Sixth International Conference on Genetic Algorithms*, 1995.

49 J. A. Hartigan. Direct clustering of a data matrix. *Journal of the Amercian Statistical Association*, 67 (337):123–129, 1972.

50 C. Haubelt, S. Mostaghim, J. Teich, and A. Tyagi. Solving hierarchical optimization problems using moeas. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, 2003.

51 J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1975.

52 M. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management*. Springer Berlin / Heidelberg, 2010.

53 E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro. Evolving clusters in gene-expression data. *Information Sciences*, 176(13):1898–1927, 2006.

54 E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155, 2009.

55 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.

56 E. J. Hughes. Radar waveform optimization as a many-objective application benchmark. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, 2007.

57 J. Izsák and L. Papp. A link between ecological diversity indices and measures of biodiversity. *Ecological Modelling*, 130:151–156, 2000.

58 Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303 – 317, 2005.

59 R. Jornsten, Y. Vardi, and C.-H. Zhang. A robust clustering method and visualization tool based on data depth. In *In Statistical data analysis based on the L1norm and related methods (Neuch□tel, 2002)*, 2002.

60 S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

61 L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis based on the L1 Norm*, pages 405–416, 1987.

62 L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

63 D. Kundu, K. Suresh, S. Ghosh, S. Das, A. Abraham, and Y. Badr. Automatic clustering using a synergy of genetic algorithm and multi-objective differential evolution. In *Hybrid Artificial Intelligence Systems (HAIS 2009)*, 2009.

64 Z. Kutalik, J. S. Beckmann, and S. Bergmann. A modular approach for integrative analysis of large-scale gene-expression and drug-response data. *Nature Biotechnology*, 26(5):531–539, 2008.

65 M. Křivánek and J. Morávek. NP-hard Problems in Hierachical-Tree Clustering. *Acta Informatica*, 23 (3):311–323, 1986.

66 X. Li, J. Zheng, and J. Xue. A diversity metric for multi-objective evolutionary algorithms. In *International Conference on Advances in Natural Computation (ICNC 2005)*, 2005.

67 J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

68 S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

69 S. W. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving From Nature (PPSN II)*, 1992.

70 G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

71 D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *Tenth National Conference on Artificial Intelligence*, 1992.

72 R. Moritz, T. Ulrich, and L. Thiele. Evolutionary exploration of e/e-architectures in automotive design. In *International Conference on Operations Research*, 2011.

73 J. N. Morse. Reducing the size of the nondominated set: Pruning by clustering. *Computers and Operations Research*, 7:55–66, 1980.

74 M. Narayanan, A. Vetta, E. E. Schadt, and J. Zhu. Simultaneous clustering of multiple gene expression and physical interaction datasets. *PLoS Computational Biology*, 6, 2010.

75 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – i. *Mathematical Programming*, 14:265–294, 1978.

76 S. Obayashi. Pareto Solutions of Multipoint Design of Supersonic Wings Using Evolutionary Algorithms. *Adaptive Computing in Design and Manufacture V*, 2002.

77 S. Obayashi and D. Sasaki. Visualization and data mining of pareto solutions using self-organizing map. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, 2003.

78 S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE Transactions on Evolutionary Computation*, 4:182–187, 2000.

79 Y. J. Park and M. S. Song. A genetic algorithm for clustering problems. In *Proceedings of the Third Annual Conference on Genetic Programming*, 1998.

80 A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Conference on Evolutionary Computation (CEC 1996)*, 1996.

81 K. S. Pollard and M. J. van der Laan. Statistical inference for simultaneous clustering of gene expression data. *Mathematical Biosciences*, 176:99–121, 2002.

82 A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

83 M. Preuss, B. Naujoks, and G. Rudolph. Pareto set and emoa behavior for simple multimodal multiobjective functions. In *Parallel Problem Solving From Nature (PPSN IX)*, 2006.

84 A. Pryke, S. Mostaghim, and A. Nazemi. Heatmap visualization of population based multi objective algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2006)*, 2006.

85 H. Rahami, A. Kaveh, and Y. Gholipour. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures*, 30(9):2360–2369, 2008.

86 I. Rechenberg. *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* PhD thesis, 1973.

87 C. Ricotta and M. Moretti. Quantifying functional diversity with graph-theoretical measures: advantages and pitfalls. *Community Ecology*, 9:11–16, 2008.

88 M. A. Rosenman and J. S. Gero. Reducing the pareto optimal set in multicriteria optimization. *Engineering Optimization*, 8:189–206, 1985.

89 P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

90 G. Rudolph, B. Naujoks, and M. Preuss. Capabilities of emoa to detect and preserve equivalent pareto subsets. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, 2007.

91 A. Saha and K. Deb. A bi-criterion approach to multimodal optimization: Self-adaptive approach. In *Simulated Evolution and Learning (SEAL 2010)*, 2010.

92 K. Sastry, D. E. Goldberg, and X. Llorà. Towards billion-bit optimization via a parallel estimation of distribution algorithm. In *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

93 O. Schütze, A. Lara, C.A. Coello Coello, and M. Vasile. Computing approximate solutions of scalar optimization problems and applications in space mission design. In *Conference on Evolutionary Computation (CEC 2010)*, 2010.

94 H. P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, 1981.

95 W. Sheng, X. Liu, and M. Fairhurst. A niching memetic algorithm for simultaneous clustering and feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):868–879, 2008.

96 H. Shimodaira. A diversity-control-oriented genetic algorithm (dcga): Performance in function optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2000)*, 2000.

97 O. M. Shir, M. Preuss, B. Naujoks, and M. Emmerich. Enhancing decision space diversity in evolutionary multiobjective algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2009)*, 2009.

98 A. R. Solow and S. Polasky. Measuring bilological diversity. *Environmental and Ecological Statistics*, 1:95–103, 1994.

99 G. Squillero and A. P. Tonda. A novel methodology for diversity preservation in evolutionary algorithms. In *Conference Companion on Genetic and Evolutionary Computation Conference (GECCO 2008)*, 2008.

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

101 K. Sugimura, S. Jeong, S. Obayashi, and T. Kimura. Kriging-model-based multi-objective robust optimization and trade-off-rule mining using association rule with aspiration vector. In *Congress on Evolutionary Computation (CEC 2009)*, 2009.

102 H. A. Taboada and D. W. Coit. Data clustering of solutions for multiple objective system reliability optimization problems. *Quality Technology and Quantitative Management*, 4:191–210, 2007.

103 L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design space exploration of network processor architectures. In *Network Processor Design 2002: Design Principles and Practices*. Morgan Kaufmann, 2002.

104 A. Toffolo and E. Benini. Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation*, 11(2):151–167, 2003.

105 S. Tsutsui, Y. Fujimoto, and A. Ghosh. Forking genetic algorithms: Gas with search space division schemes. *Evolutionary Computation*, 5(1):61–80, 1997.

106 T. Ulrich. Pareto-set analysis: Biobjective clustering in decision and objective spaces. *Journal of Multi-Criteria Decision Analysis*.

107 T. Ulrich and L. Thiele. Maximizing population diversity in single-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2011)*, 2011.

108 T. Ulrich and L. Thiele. Bounding the effectiveness of hypervolume-based (mu + lambda)-archiving algorithms. In *Learning and Intelligent Optimization Conference (LION 6)*, 2012.

109 T. Ulrich, D. Brockhoff, and E. Zitzler. Pattern identification in pareto-set approximations. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, 2008.

110 T. Ulrich, J. Bader, and L. Thiele. Defining and optimizing indicator-based diversity measures in multiobjective search. In *Parallel Problem Solving From Nature (PPSN XI)*, 2010.

111 T. Ulrich, J. Bader, and E. Zitzler. Integrating decision space diversity into hypervolume-based multiobjective search. In *Genetic and Evolutionary Computation Conference (GECCO 2010)*, 2010.

112 R. K. Ursem. Diversity-guided evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2002)*, 2002.

113 D. A. Van Veldhuizen and G. B. Lamont. Multiobjective optimization with messy genetic algorithms. In *ACM Symposium on Applied Computing*, 2000.

114 M. Vasile and P. De Pascale. Preliminary design of multiple gravity-assist trajectories. *Journal of Spacecraft and Rockets*, 43(4):794–805, 2006.

115 M. L. Weitzman. On diversity. *The Quarterly Journal of Economics*, 107(2):363–405, 1992.

116 L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

117 R. Xu and D. C. Wunsch. *Clustering*. Wiley, 2009.

118 X. Yu and M. Gen. *Introduction to Evolutionary Algorithms*. Springer, 2010.

119 A. Zhou, Q. Zhang, and Y. Jin. Approximating the set of pareto optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 13:1167–1189, 2009.

120 E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature (PPSN VIII)*, 2004.

121 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.

122 E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, 2002.

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

124 E. Zitzler, L. Thiele, and J. Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14:58–79, 2010.

Curriculum Vitae

Personal Information

Tamara Emiliana Ulrich
Born September 2, 1983 in Schwyz, Switzerland
Citizen of Switzerland

Education

2008–2012 Doctoral student at Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland

2002–2008 Master studies in information technology and electrical engineering at ETH Zurich, Switzerland

2002 Matura at Gymnasium Immensee