University of Applied Sciences and Arts Northwestern Switzerland
School of Business

Master Thesis

Graphical User Interface For Multi-Objective Decision Support

Author:
Benjamin Keller

Supervisor:
Dr. Thomas Hanne

A master thesis presented to the School of Business of the University of Applied Sciences and Arts Northwestern Switzerland in partial fulfillment of the requirements for the degree of Master of Science in Business Information Systems

July 2016

Multi-objective problems arise in various fields of application. Generally there is not a single optimal solution to these kind of problems but a set of Pareto optimal solutions with relative trade-offs. Which solution is the "best" in a given situation depends on the judgment of a decision maker. To support the decision maker in the decision process software tool support is desirable. This research proposal outlines a study focused on visual representation of alternative solutions and interactive solution space navigation. The study will explore a method to narrow the solution space considered by a search algorithm to the decision maker's areas of interest. The methods to specify the decision maker's preference through a graphical user interface and how this information is considered by the search algorithm are part of the research.

According to the design science research approach a prototype of a software tool has been implemented. The application features a graphical user interface whereby a user is able to specify the preferred objective range for solutions. Two variants of a method to enable the multi-objective search algorithm to consider these preferences have been implemented and tested on example problems. The validation through experiments shows that the proposed methods are suitable to reduce computing time to find solutions in the preferred range compared to solving the problem without considering the decision maker's input.

I would like to thank Prof. Dr. Thomas Hanne for his valuable support and coaching in this thesis. It has been a very pleasant collaboration and I appreciate your effort and time. I also thank my wonderful wife who motivated and encouraged me always. Thanks for pushing me beyond my limitations. I love you. And I thank my God who makes everything possible.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

The purpose of this study is to investigate how preference information of a decision maker can be captured and used to find a most appropriate solution for a multi-objective optimization problem. Multi-objective optimization problems exist in different scientific fields as in engineering, economics or logistics. Multi-objective problems arise when two or more often conflicting objectives are to be optimized for a problem. For multi-objective problems in general a single optimal solution does not exist but rather a set of so called Pareto optimal solutions. Preference information of a human decision maker is needed to determine a "best" solution out of the possible, qualitatively indistinguishable solutions of a Pareto optimal set. To make multiple-criteria decision-making appropriate for non-experts in optimization technologies the preference information could be specified with graphical software tools.

This chapter gives an overview about the topic discussed in this study by presenting the problem statement and thesis statement, research questions, scope and limitations, significance, definition of terms and an outline of chapters.

1.2 Background Information

In multi-objective problems several objective values must be optimized simultaneously. When having several objective functions in general there is no single optimal solution. The meaning of optimum changes for these optimization scenarios as the goal is to find a good compromise in objective values. A solution of a problem is Pareto optimal if it is not dominated by another solution. A solution dominates another if it is "better" (i.e.

closer to the optimum) in at least one objective value without any other objective being "worse". In practical applications usually out of the set of feasible solutions only one is to be selected or realized. To find the most appropriate solution the input of a human decision maker is needed.

The requirements a decision maker can have on a solution are diverse. The decision maker could want to define minimum or maximum boundaries for objectives. That means he or she is only interested in solutions where selected objectives are beyond or between certain boundaries. These are rather simple requirements but there could as well be more complex demands. For example objectives could be classified as more or less important. This importance could be specified absolutely or relatively between objectives. One could also think of relation functions that define the significance of value boundaries between two or more objective dimensions. For example assuming that generally a high value for an objective O_1 is desired, a decision maker could want to express that the target level of O_1 could be decreased but only if another objective O_2 in return increases by not less than a certain amount.

To find a Pareto optimal solution set or its approximation for complex multi-objective optimization problems several kinds of search techniques can be used. Among the different methods evolutionary search algorithms such as NSGA-II or SPEA2 seem to be particularly suitable (Coello, 2001). These methods deal with a set of possible solutions (a so called population) simultaneously and can find several Pareto optimal solutions in one run rather than sequentially producing single solutions. With a population the algorithms imitate selection, mating and genetic mutation to produce solution generations. The quality of a solution is translated into a fitness value which is used for influencing the selection of candidates for mating.

1.3 Problem Statement

A decision maker dealing with alternatives resulting from multi-objective optimization must select the best possible compromise. The selection candidates are generated by a search algorithm and represent a Pareto optimal set or its approximation. The decision maker must narrow the solution set according to his or her preferences.

In practically applied multi-objective optimization the calculation of objective values can be very expensive in terms of computation time. An example is an aerospace engineering problem where one could search for wing shape design parameters with a multi-objective optimization algorithm and depend on finite element simulation to determine objective values (e.g. drag, lift, mechanical stress, fuel efficiency, etc.).

Also the number of Pareto optimal solutions can become very large. The size of the solution set possibly grows exponentially with the dimension of the problem (i.e. its number of objectives).

Using preference information to guide an optimization algorithm could lead to faster convergence to a preferred solution. This is in comparison to a non-directed search which will produce solutions which do not adhere to the decision makers preference.

1.4 Thesis Statement

It is possible to design a software tool that supports a user (decision maker) in the search for a best compromise solution in a multi-objective scenario. The tool provides this support by the following means:

- It provides a graphical user interface for navigating the solution set.
- It provides a graphical user interface for the specification of preferences.
- The preference information is used to guide the search algorithm.

1.5 Research Questions and Objectives

1.5.1 Research Questions

The main research questions of the study are:

- How should a tool be designed to support multi-criteria decision making?
- How can preference information identify a best compromise from a Pareto optimal set?

In order to answer the main research questions, the following sub-questions must be answered in advance:

- How can a tool display a Pareto optimal set of solutions?
- How can a Pareto optimal set be visualized for intuitive perception?
- How can a decision maker visually specify preference information?
- How can preference information be used to guide a search algorithm?
- How can this preference information be represented graphically?

1.5.2 Objectives

The following objectives shall be accomplished by the study:

- Show that a best compromise solution can be found faster with a tool that supports the decision maker.
- Provide a user interaction model design for visual preference specification.
- Adapt a search algorithm to take preference information into account.

Illustrate these concepts with a tool prototype ...

- ... that supports non-experts in multi-objective problem solving.
- ... that helps a decision maker to navigate a Pareto optimal set.
- ... whereby a user can visually specify preference information.
- ... that finds solutions according to the users preference.

1.6 Scope and Limitations

The study focuses on the design of visualization and navigation of multi-criteria solution sets and also on how preference information can be input to and processed by a decision-support tool. The focus is not on a specific multi-objective problem or on solution methods. However, sample data must be produced to investigate the focused subject.

1.6.1 Search Algorithm

There exists a multitude of search methods to apply to multi-objective problems. However, the study is focused on the tool and the interaction between the user and the tool. The navigation in a set of solutions is of importance and not how such solutions are generated. The underlying search algorithm is not in the research focus. Therefore, a single search algorithm is selected to be used. It is the goal of the study to produce data by solving example problems. The selection is more or less arbitrary. The selected algorithm shall be suitable and well established. Examples are NSGA-II or SPEA2. These score good benchmark values in multi-objective test problems (Zitzler et al., 1999) and implementations are easily available.

1.6.2 Preference Information

Preference information a user can input to the tool is limited to desired target levels. Relative importance of an objective can only be specified by relaxing the the target level for it and not by other means as for example with relative weights.

1.6.3 Problems

The term multi-objective problem refers to problems with two or more objectives. So called many-objective problems exist with possibly hundreds or thousands of objectives. The techniques to visualize high numbers of criteria differ from the ones suitable for just a handfull of objectives and will not be part of this study. Only problems with a maximum of ten objectives will be considered.

The complexity of problems is limited in order to focus on the methods and mechanisms of user interaction with the tool instead of architecture and implementation details of the problem to be solved. Therefore, relatively simple test problems are used to assess the functionality of the tool. It is out of scope to consider problems that require complex calculations as for example simulation or finite element methods.

1.7 Chapter Overview

Chapter 2 provides an overview of literature in the relevant fields for the study. Chapter 3 discusses the research approach and methods. In chapter 4 the design of the application prototype is presented considering the search algorithm, user constraints handling and visualization. Also the implementation of the knapsack example problem is explained. The next chapter (5) describes the experiments conducted in order to validate the proposed methods. The results of these experiments are presented in chapter 6. The final chapter 7 discusses the findings of the study and recommends aspects for further research and development.

Chapter 2

Literature Review

2.1 Introduction

This chapter gives an overview of the relevant subjects of the thesis. It is a survey of past and recent scientific activity in the fields applicable to the matter of this study (Fig. 2.1). It defines the context and base theory of this thesis and identifies the knowledge gap that this study will address.

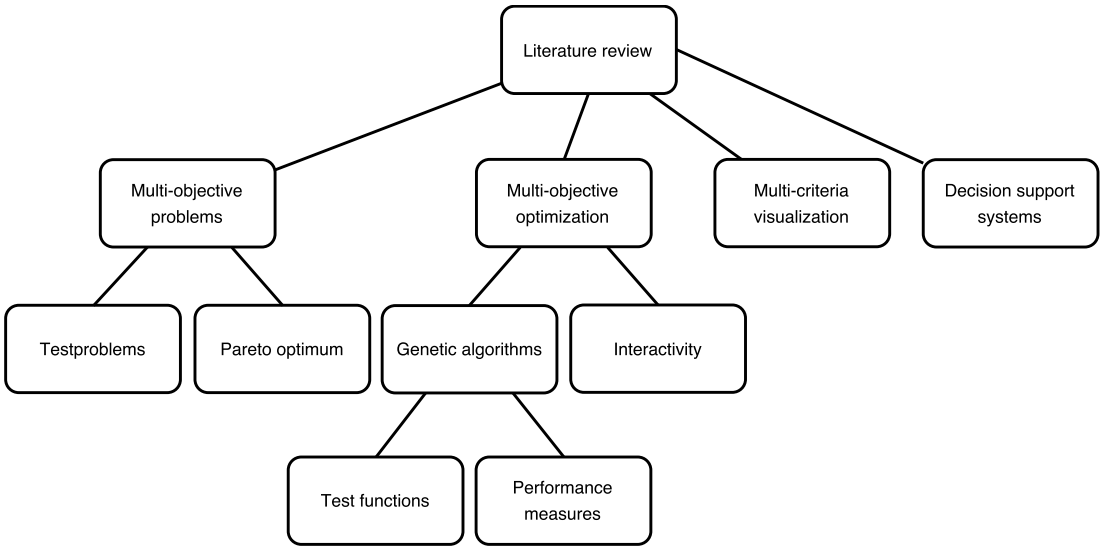


FIGURE 2.1: Literature review thematic overview

The chapter will first discuss the most notable publications regarding multi-objective problems and their optimization in general. The next section focuses on the visualization of solution sets. It is followed by a discussion of literature covering genetic search algorithms. Then previous work on interactivity with search algorithms is presented.

Finally, the summary will point out the gap in literature and the significance of the present study.

2.2 Multi-objective Problems

In many real-world planning and decision scenarios multiple conflicting objectives have to be considered simultaneously. These multi-objective (or also multi-criteria) problems arise in many different fields such as economics, engineering or logistics (Zitzler et al., 2004).

Examples include financial portfolio optimization (Hassan, 2010), electronic circuit design (Dobeš et al., 2013), freight logistics (Caramia and Dell'Olmo, 2008) or water resource planning (Major, 1977).

Osyczka (1985) defines multi-objective problems as "finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker."

Multi-objective problems define a vector $F(x)$ of k objective functions f_i to be maximized (or minimized - for simplicity reasons maximization is considered, criteria for which a minimal value is desired appropriate conversion is assumed). The objective functions are subject to constraints.

2.2.1 Pareto Optimum

With several objective function the meaning of optimum changes. Generally it is not the case, that a single solution optimizes all objective functions. Therefore a single "best" solution does not exist. Comparing two solutions is more complex. There is a "trade-off" between the objectives. In one solution objective f_1 achieves a better value than objective f_2 in another solution the value for f_2 may improve while objective f_1 deteriorates.

Considering a set $\mathcal{F}$ of k-dimensional vectors, a vector of decision variables $\boldsymbol{x}^* \in \mathcal{F}$ is *Pareto optimal* if there exists no $\boldsymbol{x} \in \mathcal{F}$ such that $f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}^*)$. This means that $\boldsymbol{x}^*$ is *Pareto optimal* if there exists no feasible decision vector $\boldsymbol{x} \in \mathcal{F}$ that would increase some criterion without causing a decrease in at least

one other criterion (Coello, 2001). In general this is true for a whole set of solutions for a given problem (the Pareto optimal set). The plot of these solutions are part of the *Pareto front*.

A vector $\boldsymbol{x}^1$ is said to *dominate* vector $\boldsymbol{x}^2$ ($\boldsymbol{x}^1 \succ \boldsymbol{x}^2$) if no component of $\boldsymbol{x}^1$ is smaller than the respective component of $\boldsymbol{x}^2$ and at least one component is greater.

2.2.2 Utopia and Nadir Vectors

The concepts of the utopia (or ideal) and the nadir (or anti-utopia) vectors are used to describe a Pareto-optimal solution set. The utopia and nadir vectors are defined for example by Trinkaus and Hanne (2005) as follows: The utopia and the nadir vector are defined by the component-wise best and worst values of the objective space, respectively.

2.3 Multi-objective Optimization Approaches and Methods

2.3.1 Scalarization

A general approach to solve multi-objective problems is to transform them into problems with a single objective through scalarization. These can then be solved with well established optimization methods such as linear programming.

The weighted sum strategy converts a multi-objective problem into an optimization problem with a single objective by assigning weights to the objectives and building the weighted sum of their functions. The weights represent the importance of the objective. The limitation of this approach is that it does not respect relative importance or trade-offs between objectives (Oliveira and Saramago, 2010).

Scalarization can also be achieved applying the global criterion method (Rao, 2009). Unlike the weighted sum method it does not allow the designer to assign proportions for each objective. The global criterion is a metric function that defines the distance to the ideal feasible solutions. The optimum solution is then found by minimizing this function.

2.3.2 Goal Programming

The goal programming method is a extension of linear programming to handle multiple objectives. In goal programming target values are assigned to the objectives. Together

with deviation variables the target values introduce new constraints to the problem. An achievement function minimizes the deviation of the objectives from the defined goal. Overviews on goal programming methods and bibliographies are provided by Schniederjans (2012), Jones and Tamiz (2002) and Jones and Tamiz (2010).

2.3.3 Genetic Algorithms

Genetic algorithms are search heuristics built on evolutionary principles such as selection, crossover and mutation. The idea of applying evolutionary principles for optimization dates back to the 1960s. A brief overview on the history of evolutionary computation and an introduction to genetic algorithms is provided by Mitchell (1996).

A characteristic feature of genetic algorithms is that they operate on a so-called population of solutions. This is of particular interest when dealing with multi-objective problems because genetic algorithms can produce multiple Pareto optimal solutions to be presented to a decision maker in a single run (Branke et al., 2008).

Populations consist of so-called individuals. Each individual is a solution candidate often described as a vector. The properties of the solution candidate constitute its chromosome or genotype. Most often the chromosome is represented as a bit string that encodes the properties of an individual in binary strings.

Genetic algorithms generally have several elements in common. From an initial population a mating population is selected. The selection is based on a fitness function. The fitness function assesses the quality of a solution candidate by assigning a score to each individual in a population. The parental population is successively replaced by an offspring population using the genetic operations of crossover and mutation. Crossover between individuals of the mating population produces child solutions that typically share properties of their "parents". According to a definable probability mutation randomly changes genes of offspring chromosomes to maintain diversity and avoid local minima (Mitchell, 1996).

Some genetic algorithms apply the concept of elitism. Elitism preserves a set of particularly fit individuals to avoid loosing their good characteristics from the gene pool. Examples of state of the art genetic algorithms using elitism are the *Nondominated Sorting Genetic Algorithm II* (NSGA-II) (Deb et al., 2002) or the *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (Zitzler et al., 2001).

2.3.4 Performance Measurement

In Okabe et al. (2003) the authors present a survey of performance indices that are used to compare multi-objective optimization solving methods and to measure the performance of multi-objective optimization algorithms that generate a set of solution as for example genetic algorithms. The term performance index (PI) is used in this paper for what is often referred to as metric in other publications. Only performance indices for the static quality of the Pareto-optimal solution sets are considered. Performance indices for measuring run-time performance, performance indices with involvement of decision maker preferences or performance indices for averaging solution sets are not considered. The performance indicators are categorized in cardinality-based PIs, accuracy PIs and distribution and spread PIs.

Distance-based accuracy PIs as proposed for example by Van Veldhuizen and Lamont (1998), Zitzler et al. (1999), Czyzak and Jaszkiewicz (1997) or Schott (1995) consider the distance values between the solution set (S) and a Pareto-optimal solution set (P) or a reference set (R). These performance indices require a given P or R. This can be difficult for real-world problems. The spread and distribution of P or R influence the value of these performance indices.

Volume-based PIs calculate the dominated area by a solution set with reference to an origin. Volume-based PIs have been proposed by Zitzler and Thiele (1999) or Wu and Azarm (2001).The sensitivity of these performance indices depends on the selection of the reference point and convex parts of the Pareto front are preferred to concave parts. For the assessment of distribution and spread of a solution performance indices are reviewed that measure either of these two characteristics and also a combination of spread and distribution information. These performance indices can be based on distance, niching or entropy.

Okabe et al. (2003) empirically compare the capability of the presented performance indices. They use both artificial solution sets and solution sets produced by solving a multi-objective optimization problem with different optimizers. The conclusion of the study is that none of the performance indices is able to represent all quality aspects for multi-objective solution sets (number of solutions, accuracy, distribution and spread). Some performance indices were even found to be misleading. Performance measurement of multi-objective optimizers by performance indices alone is not recommended. That applies particularly to real-world applications when usually only little information on the shape of the Pareto-optimal front is available.

2.4 Visualization

According to Miettinen (2014) graphical representation is a very important part of decision support systems. Miettinen (2014) categorizes visualization techniques for sets of multi-dimensional alternatives. The author defines six classes. Properties and limitations of the techniques (maximum number of alternatives, maximum number of criteria, sensitivity to ordering of criteria, suitability for pairwise comparison, ability to represent aspiration levels) are discussed. The author concludes that many possibilities for visualization exist and recommend to display the same data with various techniques. In this way a decision maker can observe a solution set from different perspectives and choose a representation pursuant to personal information processing style. Also changing order or assignment of criteria can help to gain more insight into relationships between alternatives and criteria.

Trinkaus and Hanne (2005) present a multi-criteria decision support system called *knowCube*. The focus of *knowCube* is to assist non-expert decision makers through visualization and interactivity. The system uses a radar chart (also referred to as spider web or spider chart) to represent multi-criteria vectors. Alternatives are visualized as polygons. The corner points represent criterion values on star-like arranged dimension axes. The decision maker can navigate the set of possible solutions interactively by dragging a vertex of a navigation polygon.

2.5 Interactivity

The decision maker plays an important role in interactive methods. The idea is to support the decision maker in the search for the most preferred solution (Miettinen et al., 2008) During the past 40 years interactive methods have been developed to combine numerical solution processes with subjective input from a decision maker. These methods are characterized by ongoing alternation between an objective computation phase and a subjective decision phase (Jahn, 2011).

The so-called STEM method proposed by Benayoun et al. (1971) involves a sequential exploration of solutions guided by the decision maker. With this algorithm the decision maker can learn the patterns of good solutions and relative importance of objectives. A best compromise solution is approached through cycles of computation and decision phases. In the decision phase the decision maker relaxes some satisfactory objectives to improve unsatisfactory objectives in the next cycle.

In the NIMBUS algorithm (Miettinen and Mäkelä, 1998) interactivity with the decision maker is based on the classification of objective functions. The decision maker classifies each objective function of a specific problem according to predefined classes. Based on the classification and including class specific parameters such as aspiration levels, upper bounds and weighting coefficients a new single objective optimization problem is formulated and solved. In an iterative process the decision maker can choose to calculate more alternatives and adapt classes of objectives and their parameters.

Miettinen et al. (2008) give an overview of interactive methods and on how preference information can be specified. The authors mention an important benefit of interactive methods: during the solution process the decision maker gains valuable knowledge about the problem such as its possibilities and limitations. Also by only generating Pareto optimal solutions that are of interest to the decision maker computational cost savings can be achieved. The three presented types of preference information specification include methods based on trade-off information, reference point approaches and methods using classification. Details and applications of the different types are presented with references to respective literature.

2.6 Summary

In this chapter a survey of past and recent literature in fields of research relevant for the proposed study has been presented. The covered topics include characteristics and terminology of multi-objective problems and their solution approaches with a focus on genetic algorithms, visualization and interactive approaches.

The surveyed literature indicates a gap of knowledge where it comes to the combination of interactive multi-objective optimization with visualization techniques. A method to leverage a graphical user interface for preference specification, search algorithm guidance and solution space exploration by a decision maker will potentially contribute to the knowledge in the relevant scientific fields and their research communities. The significance of the proposed study lies in its effort to study such a method through the design and evaluation of a prototype tool implementation.

Chapter 3

Research Method

3.1 Introduction

This chapter explains the research design, strategy and methodology that will be applied in the master thesis. It defines why and how the study is conducted.

3.1.1 Purpose of the study

The purpose of the study is to find principles for a visual tool that can help a decision maker to find solutions for a multi-objective problem more efficiently. The study will elaborate on methods to guide the search for solutions in an unknown solution space according to the user's preference.

3.1.2 Expected Result

The expected result is a tool that demonstrates the visual support for the user. It can be shown that a user finds a solution according to specified preferences more efficiently when the search in the solution space is guided according to this preference information compared to a non-directed search. A tool can increase the efficiency of a decision maker searching for a most preferred solution in a multi-objective problem by the usage of visual representation of alternatives and visual specification of preference information. The preference information directs the search in the solution space towards the area of interest of the decision maker.

3.2 Research Philosophy

This section discusses the meta-scientific positioning of this study based on the 'research onion' (Fig. 3.1) by Saunders and Tosey (2012).

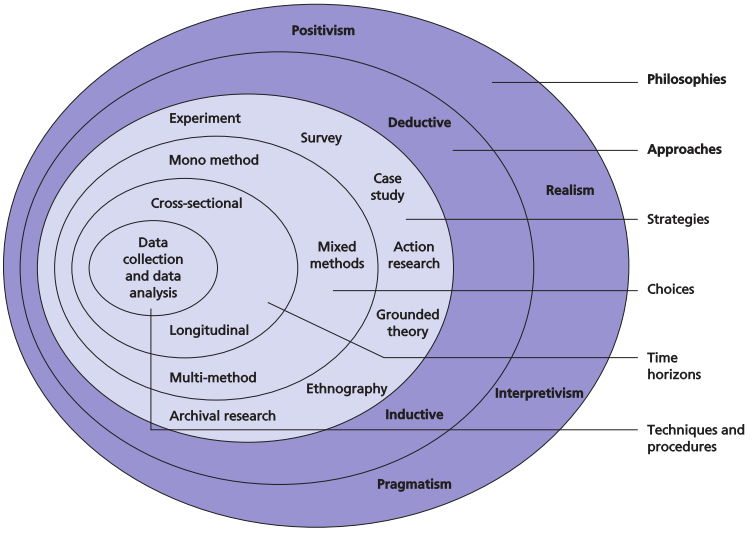


FIGURE 3.1: Research onion Saunders and Tosey (2012)

The author takes a pragmatist viewpoint on knowledge and research. It is possible to observe and describe reality from multiple viewpoints. Knowledge can be provided by observation and also by subjective meanings depending on the research question. For example on the one hand the efficiency of a decision maker in terms of time to make a decision and partly its quality can be objectively measured and compared. On the other hand the process of how to come to a decision and how valuable support in this process looks like is subjective and depends on the decision maker's preference, the type of problem and other aspects.

A design science philosophy is applied in this thesis. It is concerned with the design of a user interaction model as an artificial construct.

Mainly a deductive approach is followed. From the theory and hypothesis of how a decision maker can be supported with a tool a design is derived and a prototype tool is developed. The evaluation of the tool will verify or falsify the hypothesis by deductive reasoning. The data to evaluate the artifact is collected with quantitative and qualitative methods. The time horizon layer is not relevant for this work.

3.3 Research Design

The research strategy is based on the design science research strategy. According to Vaishnavi and Kuechler (2015) design science research creates new knowledge through the design of artifacts and through analysis of use and performance of these artifacts.

The design science research process model as described by Vaishnavi and Kuechler (2015) (Fig. 3.2) is adapted in this study.

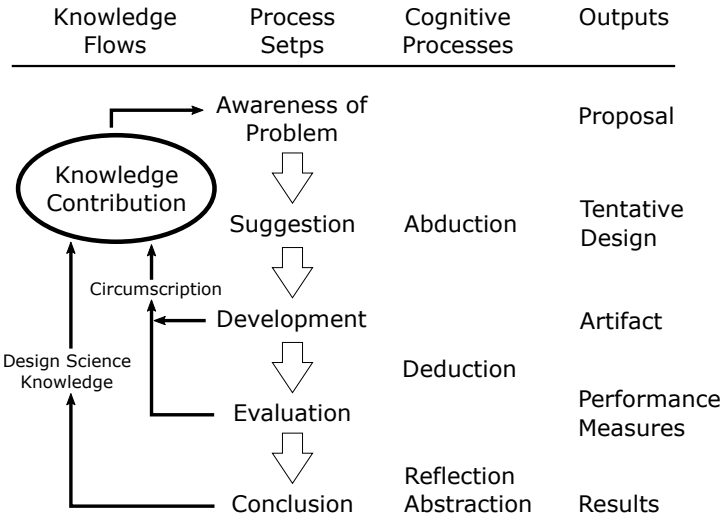


FIGURE 3.2: Design science research process model according to Vaishnavi and Kuechler (2015)

The output of the first step *Awareness of problem* is the *proposal*. It is concerned with the realization of knowledge gaps in the research field. The input of the supervisor and the literature review raised awareness of the problem dealt with in this thesis.

In the *Suggestion* phase new functionality is envisioned based on a novel configuration of existing or new and existing elements. This is a creative step and results in the *tentative design*.

The tentative design is further developed and implemented in the *Development* step. The activities depend on the kind of *artifact* created as the output of this phase. The novelty is in the design of the artifact and not in its implementation or construction.

In the *Evaluation* phase the artifact is evaluated according to defined criteria. It contains an analytic component in which hypothesis about the behavior of the artifact are made. Together with the information gained in the construction and running the artifact results are looped back to the *Suggestion* phase of a new research cycle.

3.4 Methodology

3.4.1 Concept

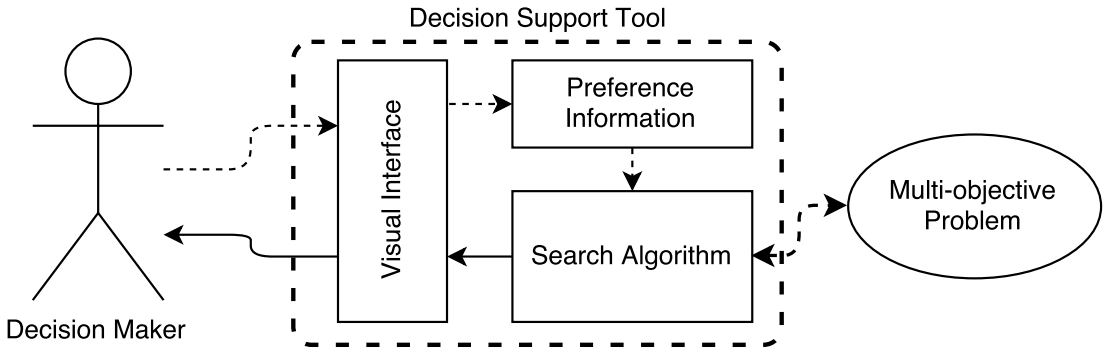


FIGURE 3.3: General concept use case

The general concept is pictured in Figure 3.3. The decision maker's task is to select a most suitable solution for a multi-objective problem. In this task the decision maker is supported by a decision support tool. The decision maker interacts with the software through a visual interface. This interface provides a graphical representation of Pareto optimal solutions (or their approximations) for the problem. These solutions are provided by the search algorithm which is applied to the problem.

With graphical elements the decision maker inputs preference information about the desired solution into the system. The search algorithm accounts for this data, narrows the considered solution space accordingly and focuses the search on the subarea of interest for the decision maker.

3.4.2 Development

A prototype of a decision support tool is developed that implements the decision concepts studied in this thesis. The concept includes the visual presentation of solution alternatives as well as graphical interface elements that allow the user to limit the search space and direct the search algorithm.

3.4.3 Evaluation

The methods implemented in the prototype tool must be validated. For validation example problems will be solved with the tool. The expectation is that a better solution will result through the limitation of the solution space using the users preference information. To be able to compare the quality of the gained solution, quality measures must

be defined and applied. How could a decision maker specify preference information in a constructed abstract test problem? The aim is to find out how a decision maker can find the ”best” solution. What does ”best” mean in an abstract test problem? Also reference data must be acquired by solving the same problem without support of the studied preference information consideration techniques.

Computational experiments on example problems with predefined values for the user input will provide data for the validation. These experiments allow a comparison of performance indicators and are expected to provide information on the influence of problem or algorithm parameters on the effectiveness of the proposed methods.

3.5 Summary

This chapter explained the research philosophy, research design and the methodology of the present study. The artifact of the design science research approach is a prototype of a decision support software tool. The developed tool will allow the decision maker to interact with a multi objective search algorithm through a visual interface. The evaluation of the implemented methods is based on computational experiments.

Chapter 4

Application Design

4.1 Introduction

This chapter describes the design and the components of the developed software tool prototype. The specifications and implementation details of the built-in multi-objective knapsack problem are presented. This problem serves as an example problem to provide actionable data to study the functionality of the application. Also the applied search algorithm with some details on the genetic operators used in the application and the methods to handle the user defined constraints are outlined. Further a description of the user interface and visualization of solutions and constraints is given.

4.2 Application Components

The application prototype consists of the following components:

1. Example problem
2. Search algorithm
3. Graphical user interface

The purpose of the example problem is to provide an actual multi-objective problem on which the search algorithm and user constraint handling concepts of the application can be applied. To be able to study the performance of the application when handling problems of different numbers of objectives, it needs to be configurable in this regard. For simplicity reasons the example problem is hard-wired into the application. That means it can only handle knapsack problems but is able to work with different configurations

thereof. A productive application of course must be able to handle arbitrary multi-objective problems. In a productive application there must be ways to define the problem to be solved, but as this is not the focus of the study the related effort to build appropriate mechanisms and application interfaces has been omitted.

The search algorithm receives the example problem and the user's preferences as inputs and produces solution candidates as output. The user's preferences are used to direct the algorithm. This is realized through constraint handling methods.

The graphical user interface is responsible to visualize the solution candidates produced by the search algorithm. It also provides the controls by which the user can specify the preferred range to direct the search algorithm. Through the visualization of solutions and preferences the user can control the algorithm and the development of the Pareto front.

4.3 Example Problem

The application requires a multi-objective problem for generating example data on which the proposed methods can be tested. A multi-objective version of the knapsack problem has been chosen for this purpose.

4.3.1 Consideration

The knapsack problem is a relatively simple multi-objective problem and due to NP-hardness it is a realistic candidate for heuristic solution methods. It also is comprehensible due to its real-life nature. For example one could imagine a robber picking as much jewelry as fits into the bag while simultaneously optimizing the items value and weight for an easy escape. Because of its comprehensibility a user can have an idea of what it means to set constraints on a certain objective. This would certainly be even stronger in a real-life problem although in general these problems are more complex and a certain domain knowledge is required to interpret objectives and constraints. Also the evaluation of solutions is quite simple whereas the number of objectives for an instance of the problem is scalable with little effort.

Furthermore the problem can be designed to have objective values in the same range for all criteria independent of the number of criteria. The problem also is scalable in terms of needed computational effort by varying problem parameters e.g. using a very large item pool.

A drawback is that the true Pareto front can only be calculated for small scale problem instances because of the NP-hardness. This however is not of specific importance as the validation can be based on quality indicators that do not need knowledge of the true Pareto front.

4.3.2 Problem Specification

This section describes the details of the implemented version of the multi-objective knapsack problem.

In a general knapsack problem a set of items and a limiting capacity of the knapsack is given. Each item has a specific profit and a constraining characteristic such as weight associated with it. The problem consists of finding the subset of items that maximizes the sum of their profits while the sum of the constraining characteristic does not exceed the capacity boundary.

The implemented multi-objective version defines a set of n items with s distinctive profits and t characteristics for each item. The knapsack also has t defined limiting capacities, one for each characteristic.

$$\begin{aligned} p_{i,j} &= \text{Profit } j \text{ of item } i \\ c_{i,k} &= \text{Characteristic } k \text{ of item } i \\ C_k &= \text{Capacity } k \end{aligned}$$

Find a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, for which

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots f_s(\mathbf{x})) \tag{4.1}$$

is maximized, where

$$f_j(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot p_{i,j} \tag{4.2}$$

subject to the capacity constraints:

$$g_k(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot c_{i,k} \leq C_k \tag{4.3}$$

Where $x_i = 1$ if item i is put into the knapsack and $x_i = 0$ if it is not.

4.3.3 Capacity definition

The definition of the capacity of the knapsack is relevant for the complexity of the problem. Considering a problem instance with a single constraining capacity, if the capacity is equal or greater than the total of the limiting characteristic of all items the problem becomes trivial. The Pareto front would consist of a single solution, namely the set off all available items.

Also when the capacity is set too small a similar situation arises. The problem becomes trivial when the capacity constraints are so low that the knapsack cannot hold a single item. Therefore the problem must be designed in a way that it lies between these two extremes. According to Martello and Toth (1990) the optimal solution of the single objective knapsack problem consists of about half of the available items when the capacity is set to half the total weight of the items.

4.3.4 Implementation

The problem instances used in this study are designed with uniformly distributed $p_{i,j}$ and $c_{i,j}$ while $0 \leq p_{i,j} \leq 1$ and $0 \leq c_{i,k} \leq 1$. The capacities are set to to half the sum of the respective characteristic of all available items.

$$C_k = \frac{1}{2} \sum_{i=1}^{n} c_{i,k} \tag{4.4}$$

4.4 Algorithm

The algorithm used to search for solutions is NSGAII. The implementation is based on the framework jMetal.NET, an implementation of jMetal in C# published under the MIT open source license. jMetal is a Java framework for multi-objective optimization with metaheuristics.

4.4.1 Problem Encoding

Solution candidates are encoded using a single binary variable. In terms of the applied metaheuristics framework a binary variable is represented through a bitstring of ones and zeros. The string has the length of n. For every item in the pool of available items the respective one or zero represents the value for x_i and defines if the item is included in the solution or not. The chromosome of the solution candidates consist of this single binary variable.

4.4.2 Algorithm Overview

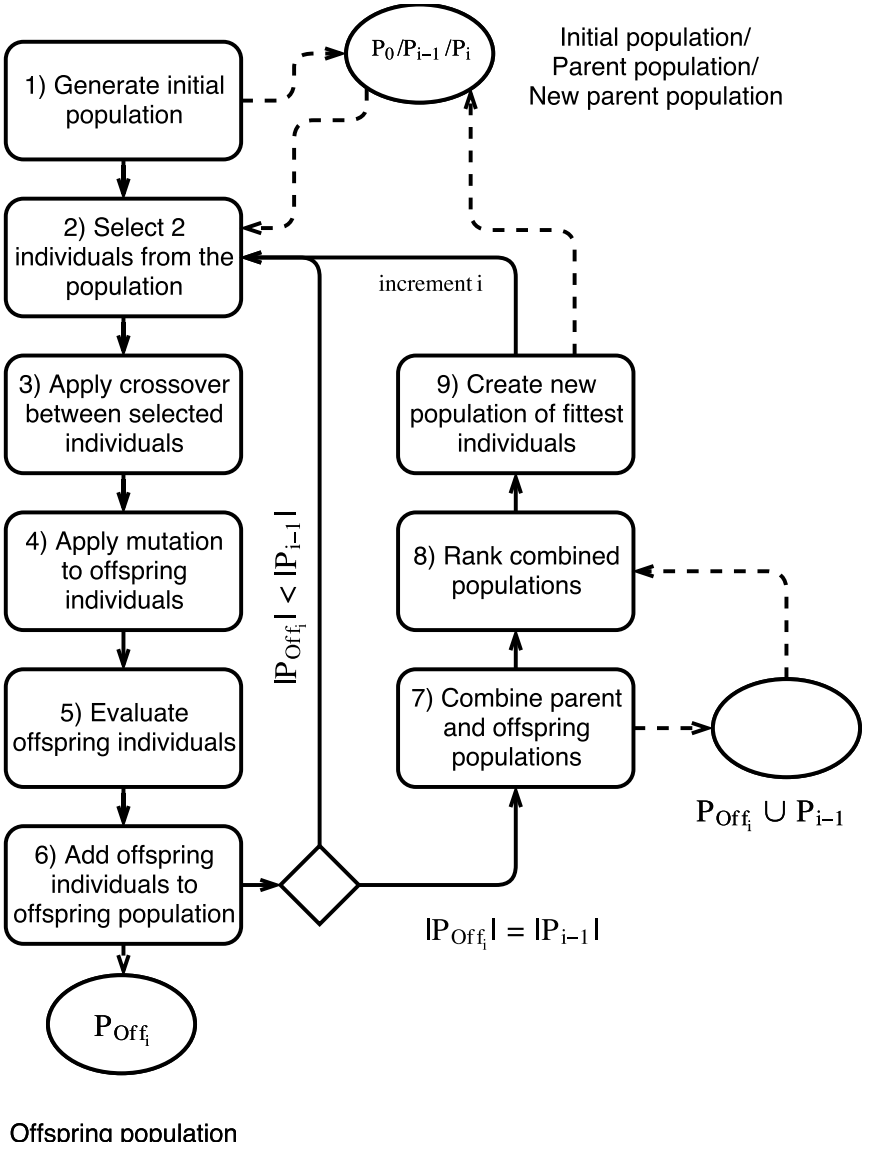


FIGURE 4.1: Overview of the NSGAII algorithm

Figure 4.1 shows the flowchart of the NSGAII algorithm. An initial population is created (1) by generating random chromosomes. From this parent population Each individual is evaluated and the fitness function is applied. From the population two individuals are selected through binary tournament (2). The crossover operator (3) takes these individuals as input and generates two offspring individuals. To both offspring individuals the mutation operator (4) is applied before evaluating (5) them and adding them to the offspring population (6). Steps (2) to (6) are repeated until the offspring population contains N (population size) individuals. When this is the case the offspring and the parent population are combined (7) and ranked (8). From this pool of individuals the N fittest constitute the new parent population.

4.4.3 Configuration and Parameters

4.4.3.1 Parameters

Population size The population size N defines how many solution candidates constitute a generation.

Mutation probability The mutation probability P_M denotes the probability for each bit of the binary variable to be flipped when the mutation operator is applied to the individual.

Crossover probability The crossover operator is applied with crossover probability P_C. If it is not applied, the crossover operator returns the unchanged parent solutions.

4.4.3.2 Selection operator

A binary tournament selection operator is used to select individuals for crossover as described as the crowded-comparison operator by Deb et al. (2002). The operator returns one individual out of two different randomly selected individuals of the population. The two individuals are then compared. If one solution dominates the other it is chosen as the return value. If the solutions are non-dominated the one with the greater crowding distance is returned. When the crowding distance is equal a winner is randomly selected. This description applies to unconstrained problems. For problems with constraints the selection operator also relies on a adjusted dominance definition that make the selection operator prefer feasible over unfeasible individuals (see 4.4.4).

4.4.3.3 Crossover operator

A single point crossover operator is applied with the set probability P_C. Two parent individuals are selected with the selection operator from the parent population. If the crossover operator is not applied, the two individuals are added to the offspring population without modification (but the mutation operator is still to be applied). If a crossover takes place the two individuals exchange a part of their chromosome. In the implemented operator a single crossover position in the bitstring is randomly selected where the chromosomes of both individuals are split. The trailing part of the bitstring is interchanged between the individuals.

4.4.3.4 Mutation operator

A bit flip mutation operator is applied to every individual in the offspring population. The operator walks every bit of the chromosome and flips it according to the set probability P_M. On average $P_M \cdot n$ bits will be flipped.

4.4.4 Constraint Handling

Two types of constraints have to be considered in this study. Firstly there are the constraints that are inherent in the problem itself due to its definition. Secondly constraints are created through the user input defining the range of objective values of interest to the decision maker.

4.4.4.1 Problem Defined Constraints

The example problem defines the constraints (4.3). It means that not every possible combination of items is also a feasible solution to the given problem. The capacity constraints have to be taken into account. However due to the randomization factor in the nature of the algorithm non-feasible individuals are created during its execution. When applying the crossover and the mutation operators individuals will be generated that violate the constraints defined by the problem. These individuals therefore represent non-feasible solutions of the problem. Although they add to the diversity of the population a decision maker is not interested in them.

4.4.4.2 User Defined Constraints

One aim of the study is to provide means to guide the search for solutions according to the decision maker's preferences. These preferences are defined as desired value ranges for the objective values that are of interest to the decision maker. One possible way to consider these preferences in the search for solutions is to define them as constraints on the problem.

The following two different methods to handle user defined constraints are applied and compared.

4.4.4.3 Constraint Handling Method A

This method is described by Deb et al. (2002) for constrained multi-objective problems. The principle of dominance is extended by methods to account for the constraints of

the problem by introducing the feasibility and a measure of the constraint violation of the solutions into the calculation of their dominance relationship. There are three possible cases when two solutions of a constrained problem are compared. A solution a constrained-dominates another solution b if:

1. Both solutions are feasible and a dominates b.
2. Both solutions are unfeasible and a has the smaller overall constraint violation.
3. Solution a is feasible and solution b is unfeasible.

With this method user defined constraints are handled in the same way as the capacity constraints originating from the problem definition. This means that a solution is regarded as "unfeasible" when it violates user defined constraints. The overall constraint violation attribute of a solution is then also calculated from capacity violation and user constraint violation. In the present implementation the deviations from either constraint were simply summarized.

4.4.4.4 Constraint Handling Method B

This method makes a distinction between hard and soft constraints. The problem defined constraints are hard constraints whereas the user defined constraints are soft constraints. According to Fig. 4.2 every solution is assigned to one of three categories . It can

(I) be feasible and within user defined constraints,

(II) be feasible but violate user defined constraints or

(III) be unfeasible.

The dominance relationship is altered in order to prefer a solution within user constraints over a feasible solution that lies outside of the users preferred range, which again dominates an unfeasible solution.

When two solutions of the same category are analyzed they are compared according to the following rules:

- When two solutions of category I are compared, the standard dominance definition is applied. The solution with the greater crowding distance is preferred if the solutions are non-dominated.

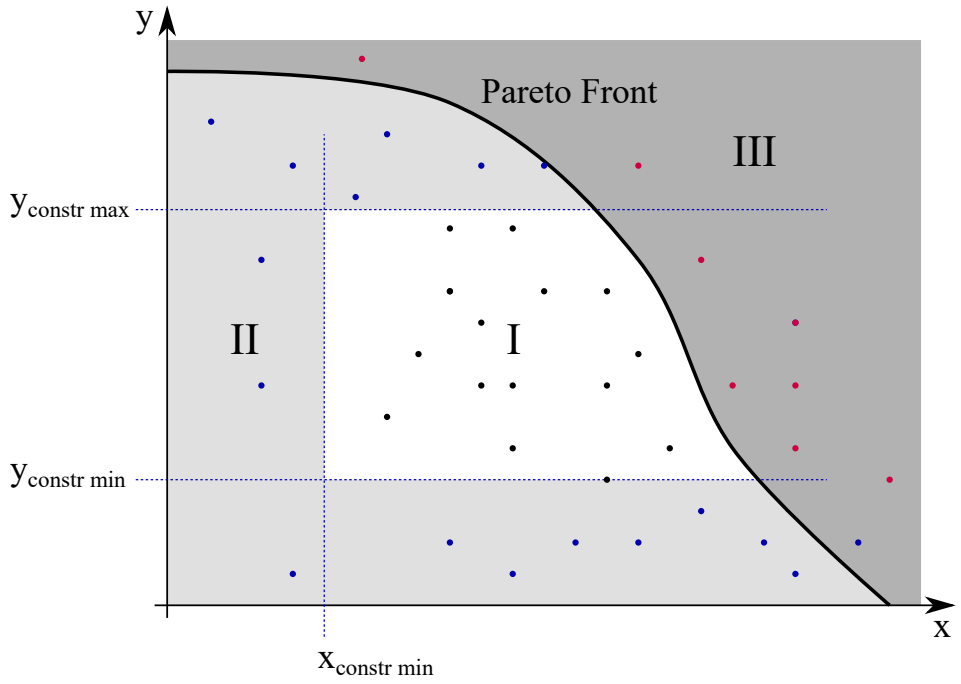


FIGURE 4.2: Visualization of three different solution categories depending on the violation of hard and soft constraints.

- When the two solutions are of category II both solutions are feasible but not within the user preferred range. In this case the solution with the smaller user constraint violation attribute dominates the other solution. In this implementation the user constraint violation attribute is calculated as the distance of the objective to the user defined boundary.
- When two solutions of category III are compared one solution dominates the other if its constraint violation attribute is smaller. This means the solution with the smaller distance to the problem constraints dominates the other.

4.5 Graphical User Interface

The user interface of the application prototype has been realized using C# and the Windows Presentation Foundation (WPF) graphical framework. The choice, although more or less arbitrary, was made for these technologies because the authors knowledge of the programming language is presumably better than of other possible alternatives. The main user interface of the application as depicted in Fig. 4.3 is composed of the interactive spider diagram, elements to control and display parameters and algorithm execution, a list of the current non-dominated solutions and a list of the user constraints.

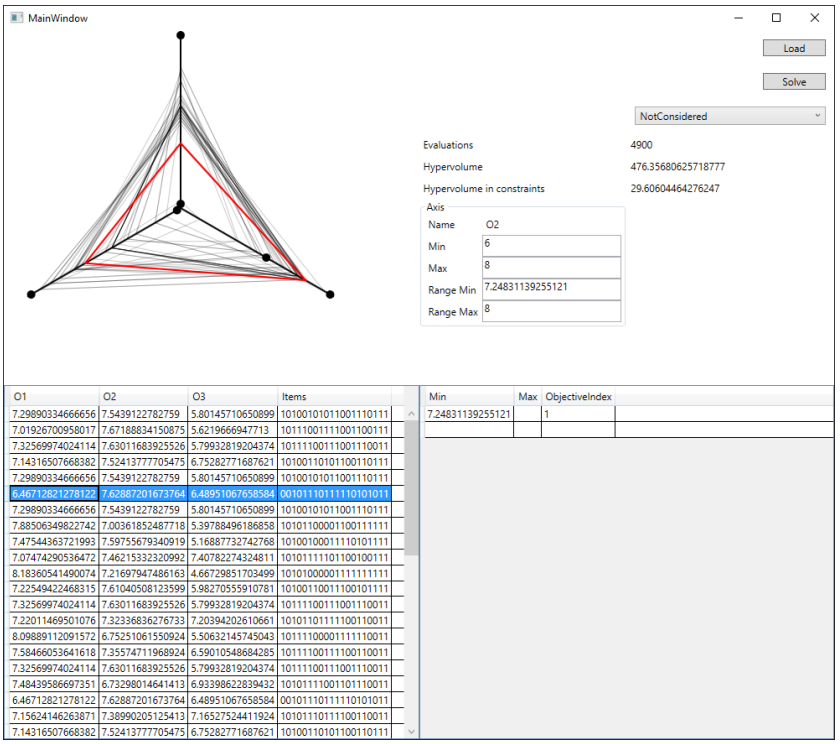


FIGURE 4.3: Example of the user interface of the application prototype.

4.5.1 Visualization with Spider Chart

The graphical user interface should support the user in the search for Pareto-optimal solutions through visual control and feedback. The visualization of solutions and user constraints is performed using a spider chart (also known as web, star or radar chart). This type of chart is suitable to display multivariate data and makes it easy to for user to locate similar solutions. The chart consists of axes originating from a common center point distributed in uniform angles. Every axis represents an objective.

Solutions are represented on the chart with polygons. The objective values of the solutions constitute points on the respective axes form the vertices of the polygon. The data points are connected with lines (edges of the polygon). The axes are individually scaled according to the range of values for the objective in the displayed set of solutions.

4.5.2 Control for User Input

The user interface was designed to allow input through direct interaction with the chart on which the solutions are displayed. For this purpose each axis of the diagram has two handles that the user can click and drag. The handles can be moved along the axis and

are used to define minimum and maximum user constraints for the objective represented by this axis.

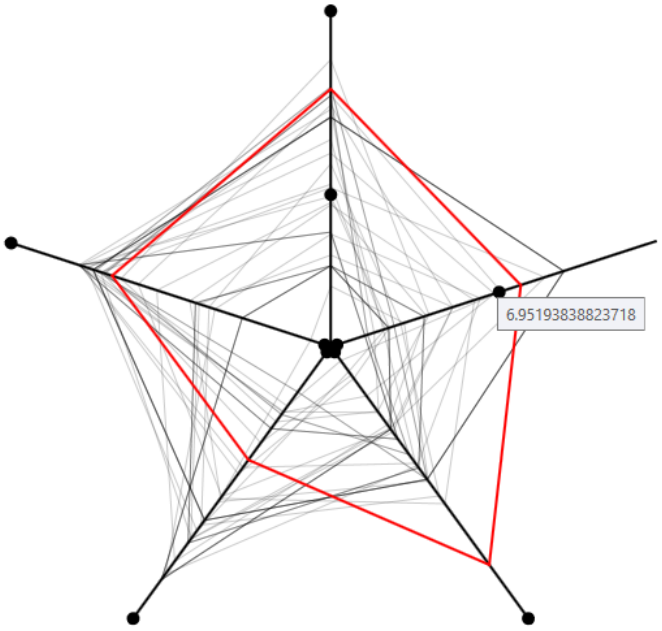


FIGURE 4.4: Spider chart user interface element for a problem with five objectives and user defined constraints on two objectives.

Fig. 4.4 shows the spider chart user interface control as it is displayed in the application. The image shows a chart for a problem with five objectives. The black dots are the draggable handles that are used to specify constraints by the user. The handle at the outer edge of the axis is used to define a maximum constraint. The handle at the center point of the axis defines a minimum constraint. Handles on the edges of the axis do not imply an active user constraint. In this picture a minimum constraint is defined for the first objective (vertical axis) and a maximum constraint is set for the second objective (axis rotated 72 degrees clockwise). A tool tip displays the value under the handle (as seen in Fig. 4.4 at objective two).

4.5.3 Use Case

The following use case description demonstrates the intended usage and interaction scenario of the application. We assume that the decision maker is required to decide on a multi-objective problem. Only one of the many possible solutions of the problem is to be put into practice. Further is assumed that the problem is available in a form which the application can interpret and generate solutions for.

The decision maker starts by loading the problem into the application and initiates the solution process. At the beginning no further user input is required. The genetic search algorithm is working on the problem and the user interface is periodically updated with the current non-dominated solution set. During this process the user can observe the development of the non-dominated front that approaches the Pareto front. Thereby the user gets an idea about the shape of the Pareto front and gains knowledge about possible ranges of the objective values.

At any time the decision maker can input preference information by setting constraints on the objective values. The user can set these constraints by dragging the respective handles on the spider web diagram that visualizes the solutions. For example the user could want to set minimum levels for certain objectives that solutions of interest must achieve. Maybe the user is interested in solutions for which a certain objectives lie in a specific range. He or she would then set a minimum and a maximum constraint on the respective objective that represent this interval. These constraints can be specified and updated while the search algorithm continues to return more solutions. The search algorithm will always take the most recent values for constraints into account.

As the solution process advances the user gets to know the solutions that lie within the preferred range and can refine and tighten the constraints. When no constraints are set the algorithm will ideally return solutions that are spread over the whole possible range of non-dominated solutions. The size of the non-dominated set is limited (theoretically maximum twice the population size in the prototype application) that means when constraints are defined the density of the non-dominated set within these constraints will increase as long as the algorithm is able to find solutions in the defined range. This way the user gets a better approximation (i.e. higher number of non-dominated solutions in the objective range) of the Pareto front in the range defined by the objective constraints. Ultimately the number of feasible solutions within constraints is reduced (at least for problems with discrete input variables such as the examined knapsack problems) as the constraints are set more and more restrictively.

4.6 Summary

This chapter described the main components of the implemented application prototype of a decision support system with a graphical user interface. It consists of a graphical interface that displays solutions and allows the user to visually specify constraints on objectives as an interactive preference input method. The genetic search algorithm NSGAII is extended with two different constraint handling methods that enable the guidance of the search through consideration of the user constraints. The application

includes an implementation of a multi-objective knapsack problem that serves as an example problem for validation.

Chapter 5

Experiments

5.1 Introduction

This chapter explains the experiments conducted to validate the presented user preference handling methods. After general considerations a section on experiment design discusses the procedure to generate the validation data as well as the applied metrics. Also the detailed parameters of the example problems and the optimization algorithm are explained and a list of experiment series is provided.

5.2 Considerations

The assumed benefit of the presented method is validated with the experiments presented in this chapter. The goal is to test the presented method for its suitability for a more efficient user directed search.

Compared to a non-directed search the proposed method is assumed to have an advantage concerning the following aspects:

- Explicitly find solutions of interest to the decision maker.
- Find more solutions in the user preferred range (density of solutions).
- Find solutions that provide a better approximation of the Pareto front within the user defined range.

Of specific importance is the number of evaluations needed to find solutions of interest to the user. Practical applications of multi-objective optimization can be very extensive

in terms of computing time. The time-consuming factor in these applications are usually not the operations of the genetic search algorithm that calculates the fitness of solutions and produces new individuals with varied chromosomes. It rather is the evaluation of these solution candidates, that means the calculation of the objective values from the new input parameter generated by the search algorithm. It is the evaluation of input parameters that can rely on computing intensive simulation as for example finite element method or other numerical analysis that calculates the objective values. Therefore a major benefit would result from a reduction of required solution evaluations to find solutions of interest.

5.3 Experiment Design

The following procedure was used to generate the data:

1. Define problem instance.
2. Define user constraints.
3. Solve problem with different constraint handling methods.
4. Record non-dominated front and metrics periodically after a number of evaluations is calculated.

The constraint handling methods are:

None The problem is solved without taking the user constraints into account. Only the constraints inherent to the problem are handled.

Method A The user constraints are treated in the same way as the constraints defined by the problem (see 4.4.4.3).

Method B The user constraints are weaker than the constraints defined by the problem. A solution within user constraints dominates solutions within problem constraints which again dominate solutions that violate problem constraints (see 4.4.4.4).

5.3.1 Metrics

The following metrics are logged as a function of the number of evaluations:

- Size of the non-dominated front

- Number of solutions within user constraints
- Number of unique solutions within user constraints
- Constrained hypervolume

The size of the non-dominated front refers to the number of solutions that this front contains. This value is not of particular interest regarding the user defined constraints but served as a reference in respect to the population size. It is therefore not further considered in the analysis.

The number of solutions within user constraints can contain duplicate solutions. As for a decision maker duplicate alternatives do not add to the available options this value is not of interest for the analysis because they do not effectively add to the real number of solutions in the user preferred range.

5.3.2 Hypervolume

The concept of the hypervolume as a quality indicator was first mentioned by Zitzler and Thiele (1998) simply as size of space covered. It is also known as the S-metric (Zitzler et al., 1999) or Lebesgue measure. Hypervolume is the n-dimensional space containing all points dominated by a set of n-dimensional points. The hypervolume is calculated relative to a reference point. According to Zitzler et al. (2007) the hypervolume metric has two important advantages: It yields a strictly better value for a solution set A than a solution set B if set A dominates B, and a set that contains all Pareto-optimal objective vectors is proven to have the maximum hypervolume measure.

5.3.2.1 Constrained Hypervolume

The user defined preferred range is of special interest. The quality of the solution set must be measured with regards to these boundaries. To measure the results in the user defined range the constrained hyper volume metric of a solution set is calculated in the following way (Fig. 5.1):

(1) Exclude all solutions that violate the user defined constraints.

(2) Set the reference point to the point $P_{ref} = (p_1, p_2, \ldots, p_k)$ where p_k is the user defined minimum boundary for objective k or 0 if no minimum boundary is defined for this objective.

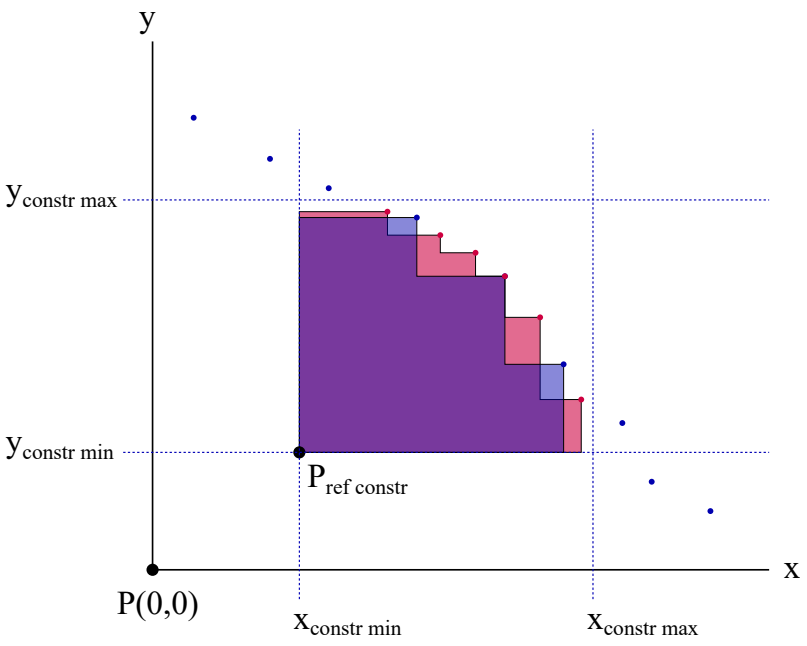


FIGURE 5.1: Visualization of constrained hypervolume of two different solution sets

5.4 Experiment Parameters

There is a significant number of input parameters to the experiments that could have an influence on the results. These parameters are defined by the example problem, the search algorithm and the user constraints.

5.4.1 Knapsack problem

Number of items n Example problems with 20, 200 and 500 items have been considered.

Number of profits s The number of objectives of the problem is defined by this value. Experiments with 3, 5 and 7 objectives have been conducted.

Number of characteristics t Number of different characteristics c (weights) of the items that together with the capacity limits constrain the problem. This value is set to $t = 1$ for all example problems.

Capacity limits C_k Capacities of the knapsack in regards to the characteristics. The single capacity limit C_1 is set to half the sum of the weight of all items.

Profit range and distribution Values that are assigned to the profits of an item. The profits in the example problem are uniformly distributed in the interval $[0.0, 1.0[$.

Characteristics range and distribution Values that are assigned to the characteristics of an item. The characteristics in the example problem are uniformly distributed in the interval $[0.0, 1.0[$.

The following 9 example problems have been used in the experiments:

		Objectives		
		3	**5**	**7**
Items	**20**	$s = 3$ $n = 20$	$s = 5$ $n = 20$	$s = 7$ $n = 20$
	200	$s = 3$ $n = 200$	$s = 5$ $n = 200$	$s = 7$ $n = 200$
	500	$s = 3$ $n = 500$	$s = 5$ $n = 500$	$s = 7$ $n = 500$

$$t = 1$$

$$C_1 = \frac{1}{2} \sum_{i=1}^{n} c_{i,1}$$

5.4.2 Algorithm

Population size Number of individuals per generation.

Selection operator Method to select parent individuals for mating. Binary Tournament as described in section 4.4.3.2.

Mutation operator Bit flip operator as described in section 4.4.3.4.

Mutation probability Set to 1/number of items. Each item correlates to one bit in the chromosome of an individual. With this value on average one bit is flipped per individual.

Crossover operator Single point crossover operator as described in section 4.4.3.3.

Crossover probability Set to 0.9.

5.4.3 User defined constraints

In a practical use case of the application a user has a visual representation of the solutions of the current non-dominated front. This visualization is constantly updated as the algorithm calculates further solutions and the user adapts and defines the constraints

according to his or her preferences dynamically as a response on the development of the Pareto front.

However, to validate the proposed method an approach with static user constraints is applied. This is in order to provide a common initial position to compare the influence of experiment parameters and the behavior of the method according to different settings.

The problem is designed so that the knapsack capacity will hold about half of all available items (n). When filling multiple knapsacks with $n/2$ items chosen at random the profit values for each objective of the packed knapsack are expected to average at $n/4$ because the profit values of the items are uniformly distributed in the interval $[0.0, 1.0[$. As the problem is a maximization problem and the items are not chosen at random but in a way to maximize the profits the objective values are expected to exceed this value.

In order to get an approximation of the Pareto front of the example problems they have been solved without setting any user constraints. Table B.1 in Appendix B lists the nadir and utopia vectors of the problems of the Pareto front approximations after 20000 evaluations. The average value of the objectives relative to the number of available items is 27,4% and 38.6% for nadir and utopia vectors respectively. This means that solutions of the Pareto front have objective values that are roughly distributed within this range.

The reason to use a maximum constraint for an objective that is subject to maximization may not be obvious. However, assuming a certain objective is not so important for the user, he or she would assign a maximum constraint on an objective to relax it. That means the user forces this objective to lower values in order to improve other objectives. If an actual improvement of other objectives can be achieved this way has to be validated.

5.4.4 Number of evaluations

The designed method is supposed to provide the decision maker with solution candidates within the preferred area of the Pareto front. The discovery of these solution, i.e. approximation of the Pareto front in the defined area, should be faster than when the algorithm does not consider the user defined preferences. 'Faster' in this context means that less evaluations are necessary to find solutions in the preferred range when comparing different methods to find these solutions.

The number of required evaluations are interesting from the perspective of usability of the method for problems of varying complexity. It is of interest if the constraints handling method is effective and providing the desired results after 1000, 10000 or 100000

evaluations. Depending on the computing time required for the evaluation of a solution this indicates whether the method is appropriate to handle problems with longer computation times or not.

Table 5.1 lists the time needed for the evaluation of multiple solutions considering different computing time to evaluate a single solution. This demonstrates that to make the search for solutions efficient it is necessary to reduce the number of evaluations needed to achieve an appropriate approximation of the Pareto front in the desired range. This is in particular the case when the evaluation takes more than a few milliseconds.

TABLE 5.1: Time needed for solution evaluation

Time per Evaluation	**Multiple Evaluations**		
	100	**1000**	**10000**
1 ms	100 ms	1 s	10 s
1 s	1 min 40 s	16 min 40 s	2 h 46 min
10 s	16 min 40 s	2 h 46 min	1 d 3 h 18 min
1 min	1 h 40 min	16 hr 40 min	6 d 22 h 39 min
10 min	16 hr 40 min	6 d 22 h 39 min	69 d 10 h

5.5 Experiment Series

This section documents the parameters applied in the different experiment series. The Table 5.2 gives an overview on which example problem configuration has been involved in which experiment series. Series B uses a population size of 500 instead of 100 and is not applied to the 20 objectives problems because an impact of this paramater is not expected based on the non-dominated solution sets resulting in series A.

TABLE 5.2: Experiment series conducted for the different example problems

		Objectives		
		3	**5**	**7**
Items	**20**	A	A	A
	200	A, B	A, B	A, B
	500	A, B	A, B	A, B

The calculation of the hypervolume turned out to be unsuitable for certain experiments. For seven objectives and solution sets of more than about 200 individuals and five objectives and solution sets exceeding 500 individuals the computation of the value was too time consuming. The execution time for 1000 evaluations of knapsack problem

solutions consumed about 1000 miliseconds of computing time. For the studied example problems this value did not vary significantly with the number of objectives and it is unrelated to the number of individuals in the non-dominated set. Table 5.3 lists example values for the calculation time of the hypervolume for a seven objective problem. It was not further examined why the computation time varies significantly for similar number of solutions. In general an exponential relationship between number of solutions and calculation time was observed. This behaviour was experienced with the jMetal.NET hypervolume implementation and computing resources of an Intel® Core™ i7-6500, 2.5 GHz. This situation can only occur for experiments with population sizes of more than 100 or 150 individuals, respectively. This is because the solution set returned by the algorithm is the non-dominated front of the union of the current parent and offspring population. The number of solutions in this solution set could theoretically become twice as large as the population size if all returned solutions are non-dominated. The metric is not calculated when the non-dominated front is larger than 300 solutions (five objectives experiments) or 200 solutions (seven objective experiments).

Table 5.3: Computation time of hypervolume for a seven objective problem

# Solutions	Calculation time [ms]
7	0
33	251
57	1853
146	68066
165	241681
176	39968
183	196144
205	42466
228	436516
261	280979
263	35740

5.5.1 Experiment Series A

Population size

The population size is set to $N = 100$.

Number of evaluations

Metrics are calculated in the interval of 100 evaluations in the range of 0 to 2000 evaluations and in the interval of 1000 evaluations in the range of 0 to 20000 evaluations.

User constraints

A.1

Minimum constraint for objective 0 set to $0.33 \cdot n$ (number of items).

A.2

Minimum constraint for objective 0 set to $0.33 \cdot n$ (number of items).
Minimum constraint for objective 1 set to $0.33 \cdot n$ (number of items).

A.2

Minimum constraint for objective 0 set to $0.33 \cdot n$ (number of items).
Maximum constraint for objective 1 set to $0.30 \cdot n$ (number of items).

5.5.2 Experiment Series B

Population size

The population size is set to $N = 500$.

Number of evaluations

Metrics are calculated in the interval of 1000 evaluations in the range of 0 to 20000 evaluations.

User constraints

B.1

Minimum constraint for objective 0 set to $0.33 \cdot n$ (number of items).
Minimum constraint for objective 1 set to $0.33 \cdot n$ (number of items).

B.2

Minimum constraint for objective 0 set to $0.33 \cdot n$ (number of items).
Maximum constraint for objective 1 set to $0.30 \cdot n$ (number of items).

5.6 Summary

This chapter detailed the experiments and metrics used to validate the constraint handling methods. The quality of the non-dominated set within the user defined constraints is assessed with a constrained hypervolume measure. Also the number of unique solutions within constraints is used for the comparison of the two constraint handling methods. Multiple experiment series are conducted on nine example knapsack problems with varying number of objectives and items.

Chapter 6

Results

6.1 Introduction

In this chapter the results of the experiments defined in the previous chapter are presented and discussed. The data generated in the experiments is presented in the form of diagrams that have been generated with the LaTeX package pgfplots (Feuersänger, 2011).

6.2 Expectations

It is expected that the hypervolume and number of unique solutions within constraints achieve higher values when either of the two constraint handling methods is applied compared to when the user constraints are not considered by the search algorithm. Probably constraint handling method B will have an advantage over A as the dominance relationship definition is hierarchical and favors feasible solutions outside the user constraints over solutions that are infeasible due to the problem definition.

It is assumed that the consideration of the user constraints will have an impact on the nadir and utopia vectors of the returned solution sets. Of particular interest is the influence of maximum constraints. The studied example problems are maximization problems. It is of interest if a maximum constraint on an objective that is maximized enables a relaxation behavior. Can such a constraint be used to limit one objective in order to achieve higher values on other objectives that are to be maximized?

6.3 Experiment Series A

The figures for the comparison of hypervolume and number of unique solutions within constraints in this section are arranged in grids of nine cells. The first, second and third row show the results for example problems with three, five and seven objectives, respectively. The knapsack problems of the first column have 20 items, those of the second 200 and those of the third 500 items. When referring to individual plots in a figure the plots are numbered from 1 to 9 from left to right and top to bottom according to Fig. 6.1.

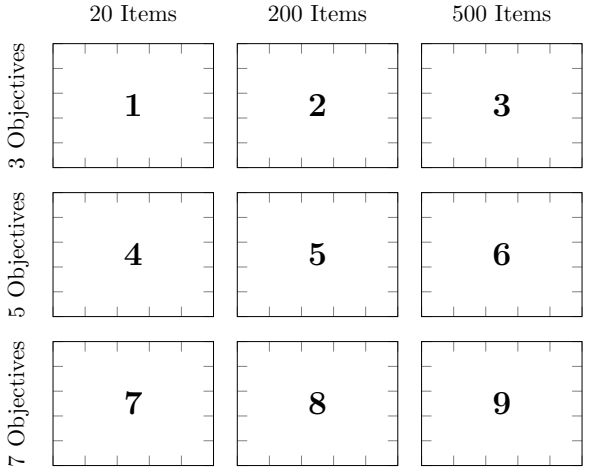


FIGURE 6.1: Numbering of individual plots.

There are two figures showing the same metric for differently scaled x-axes. The first figure shows the development of the values in the interval from 0 to 2000 evaluations in steps of 100, the second figure shows the value for the range of 0 to 20000 in intervals of 1000 evaluations.

6.3.1 Experiments A.1

The experiment Series A.1 defined a minimum constraint on objective 0 with a value of $0.33 \cdot n$.

Fig. 6.2 shows the development of the constrained hypervolume in the range from 0 to 2000 evaluations. For the plots 3,6 and 9 the hypervolume is zero because the algorithm did not yet find any solutions that comply with the constraint. For the simple experiment with three objectives and 20 items there is no significant advantage of one of the methods. The hypervolume levels out after about 1400 evaluations. However, the slight

difference in the value indicates that the different runs did not return identical solution sets. Method A returns a steeper curve for plots 5 and 8. This method returns solutions complying with the user preferences after a significant lower count of evaluations compared to the other methods.

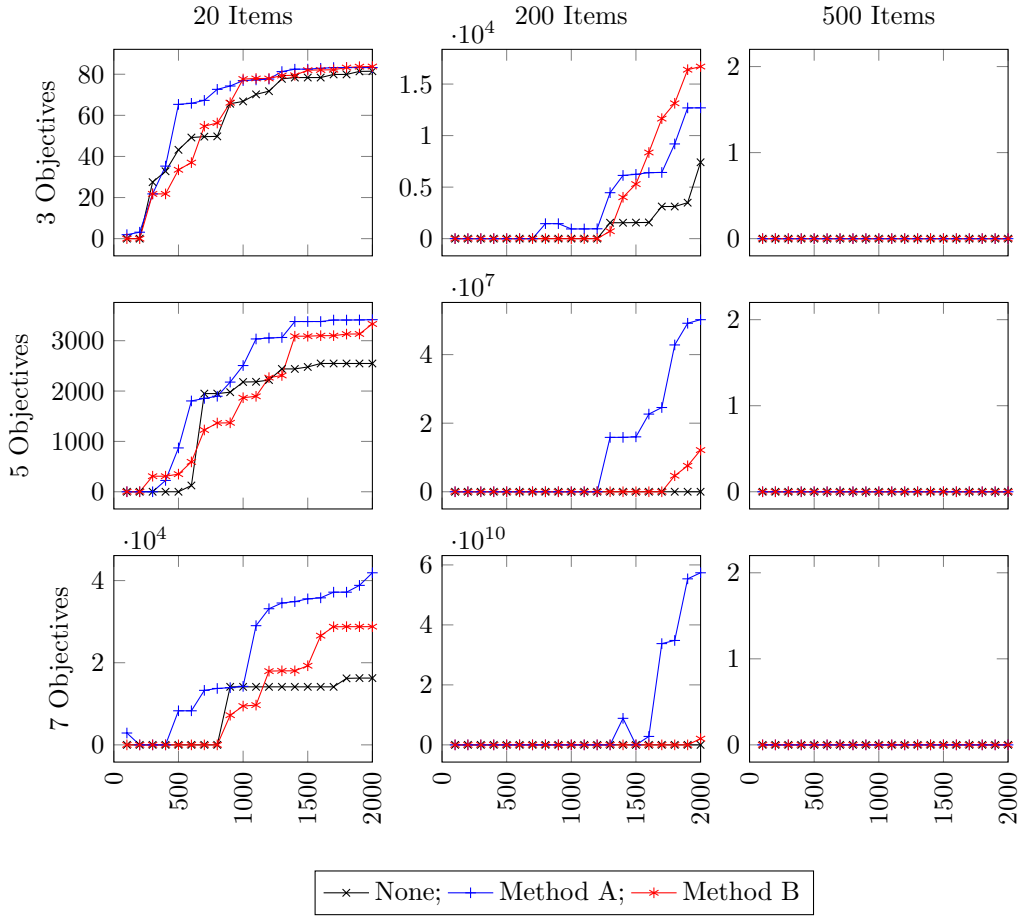


FIGURE 6.2: Development of constrained hypervolume as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$.

Fig. 6.3 shows the development of the number of unique solutions within user constraints in the range from 0 to 2000 evaluations. The plots 3,6 and 9 show a line at 0 solutions that complies with the plots of the hypervolume in the same range (Fig. 6.2). The search algorithm did not return solutions in the constrained range. The problems with 20 items (plots 1, 4, 7) show an increasing advantage of the constraint handling methods over not considering the constraint with an increasing number of objectives. Interestingly, in plot 1 the run without consideration of the constraint yields more solutions but the hypervolume metric is slightly lower than for the runs with method A and B. In plots 4 and 7 the constraint handling methods perform better while there is no clear advantage

of one method over the other considering hypervolume and number of solutions. For the problems with 200 items method a clearly yields more solutions.

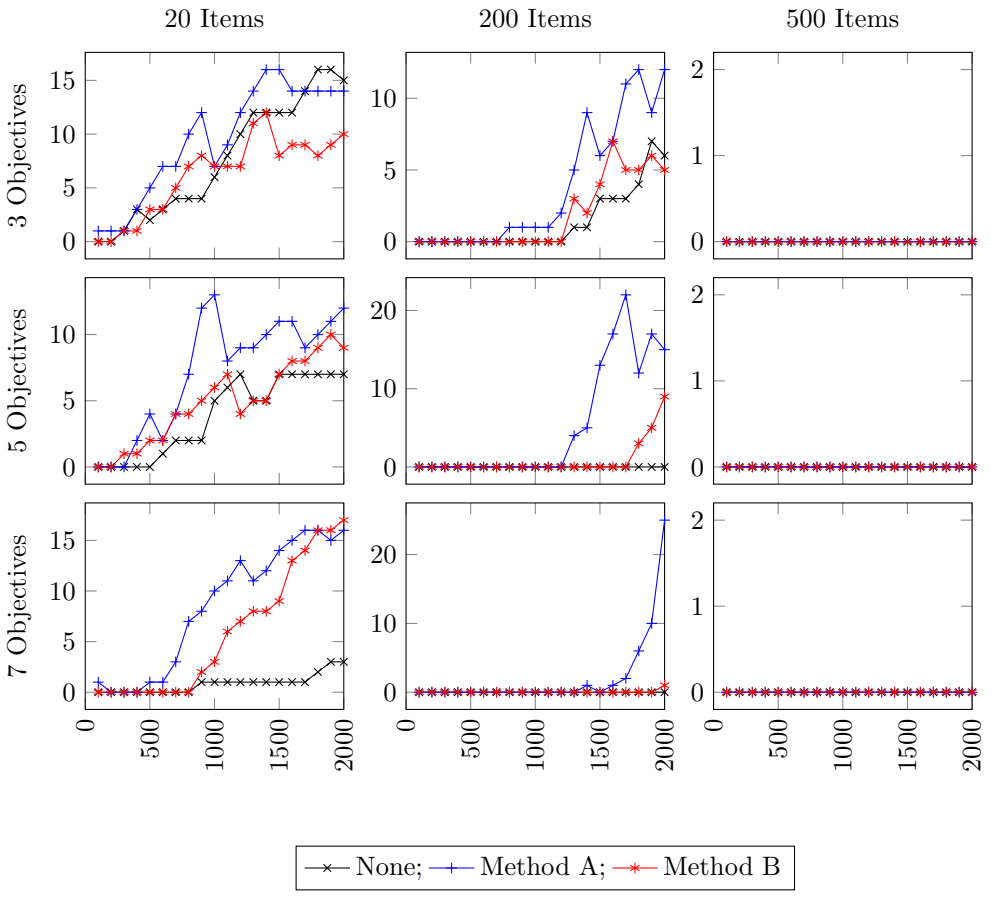


FIGURE 6.3: Development of number of unique solutions within constraints as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$.

Fig. 6.4 shows the development of the hypervolume metric for up to 20000 evaluations. The value levels for problems with 20 items for all runs except for the seven objective problem for the run without constraint consideration. From the collected data it does not become clear why the algorithm behaves in this way. It could be possible that the non-dominated front reached the true Pareto-front in the constrained range. However, the corresponding plots for the number of solutions (Fig. 6.5) shows still some variation that would be visible in this plot with a changed y-axis scale. The experiments with 200 and 500 items show a clear advantage of the constraint handling methods. For the 200 items problems method B seems to perform slightly better than method A. The latter yields bigger hypervolume measures for the 500 items problems. Except in plot 7 all hypervolume curves increase continuously.

Fig. 6.5 shows the development of the number of solutions for up to 20000 evaluations.

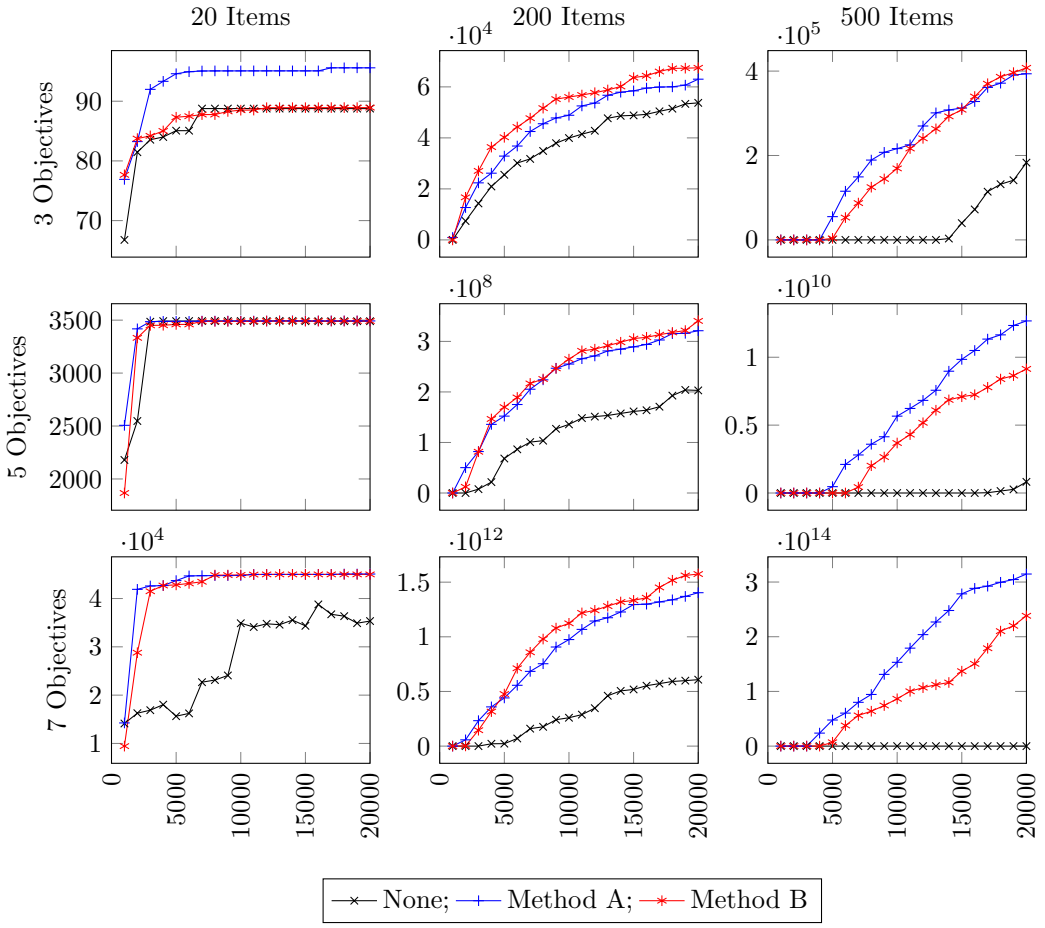


FIGURE 6.4: Development of constrained hypervolume as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$.

The plots 6 to 9 show that methods A and B perform comparably well against the experiment with no constraint consideration. For the other problems the number of solutions is quite similar. Although there is a general tendency for the increase of the number of solutions found it is not really continuous. The value evens out at about 100 solutions in the experiments 5, 6, 8 and 9. This could be due to the population size that is set to 100 in this experiment series.

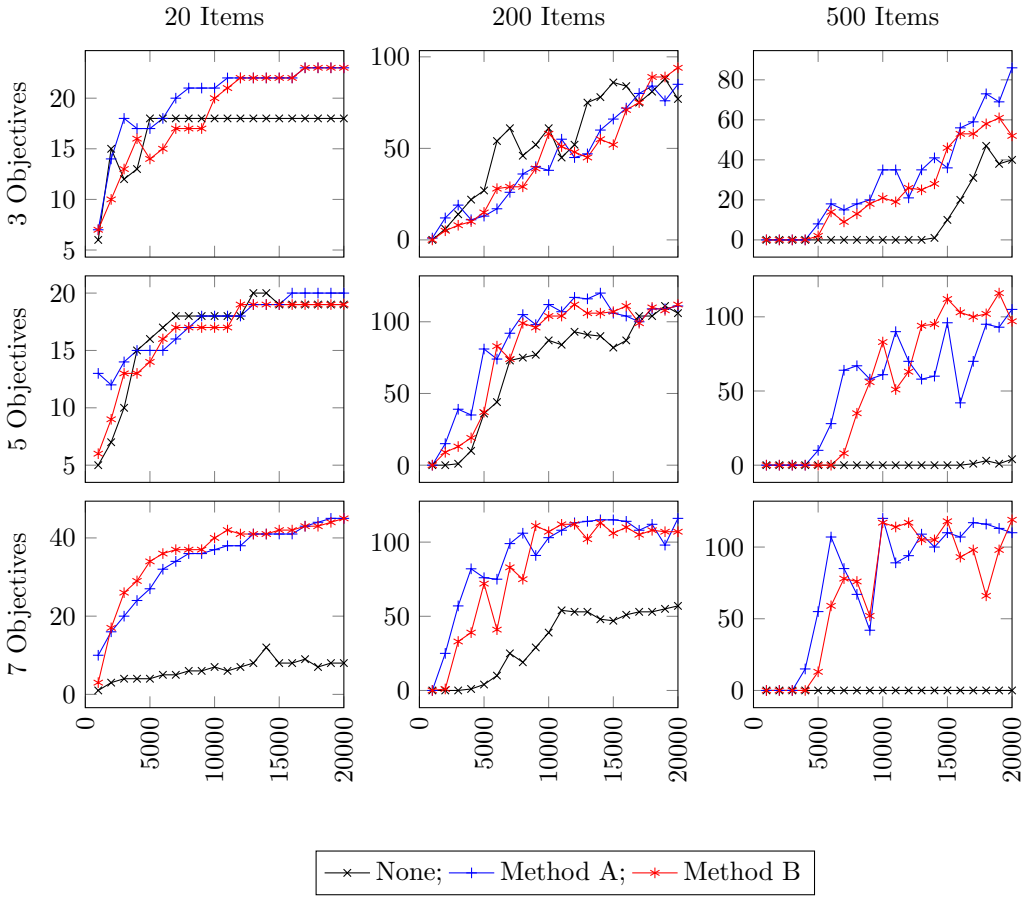


FIGURE 6.5: Development of of number of unique solutions within constraints as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$.

6.3.2 Experiments A.2

Two user constaints are set in these experiments. A minimum constraint is set on the the first objective with a value of $0.33 \cdot n$. It is the same constraint as in the series A.1. Likewise a minimum constraint is set on the second objective with the same value.

As now two constraints are in place the hyper volume value is expected to decrease in general as the reference point is moved through the additional constraint. The number of solutions within constraints will probably decrease compared to series A.1 as the preferred range is more restrictive.

From the development of hypervolume and number of solution curves within the first 2000 evaluations in Fig. 6.6 and Fig. 6.7 one can see that only for the simplest two problems with 20 knapsack items and three or five objectives the search returned solutions. As expected the value for the hypervolume is smaller. The curves also flatten a bit later (after more evaluations). The number of solutions measure is not significantly

lower compared to series A.1. Noticeable is that not considering the user constraints in experiment 1 results in the highest number of solutions. The non-dominated front did not reach the range defined by the constraints at the other seven experiments.

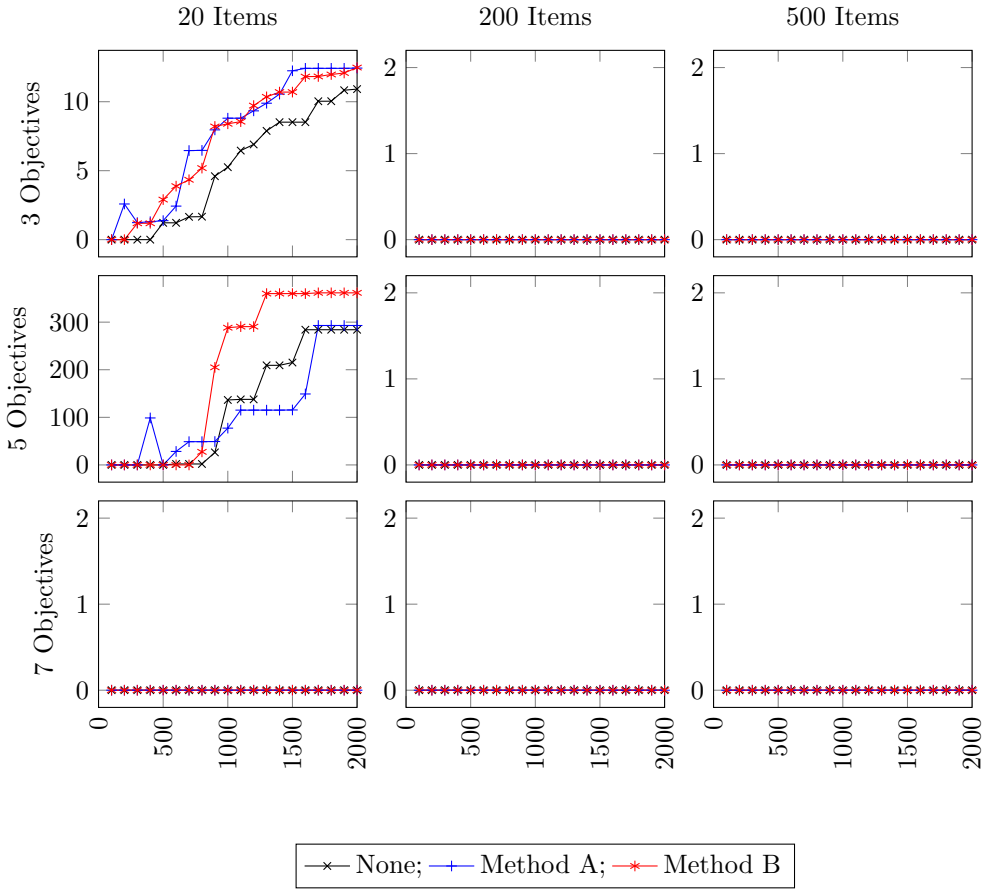


FIGURE 6.6: Development of constrained hypervolume as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Minimum constraint for objective 1 set to $0.33 \cdot n$.

Fig. 6.8 shows the development of the constrained hypervolume for up to 20000 evaluations. The plots 3, 5, 6, 8 and 9 show clearly better results for the constraint handling methods over the reference run where the constraints are not considered. Method A seems to perform a bit better as it finds solutions after less evaluations and reaches a higher metric after 20000 evaluations in four of the five mentioned plots. The curves for the 20 item problems flatten out after about 5000 evaluations. A comparison with the true Pareto-front could show if the found solutions are part thereof. The Pareto-front has not been calculated for the problems because of NP-hardness. For the problems with 20 items an exact calculation possibly could be performed with reasonable effort, however.

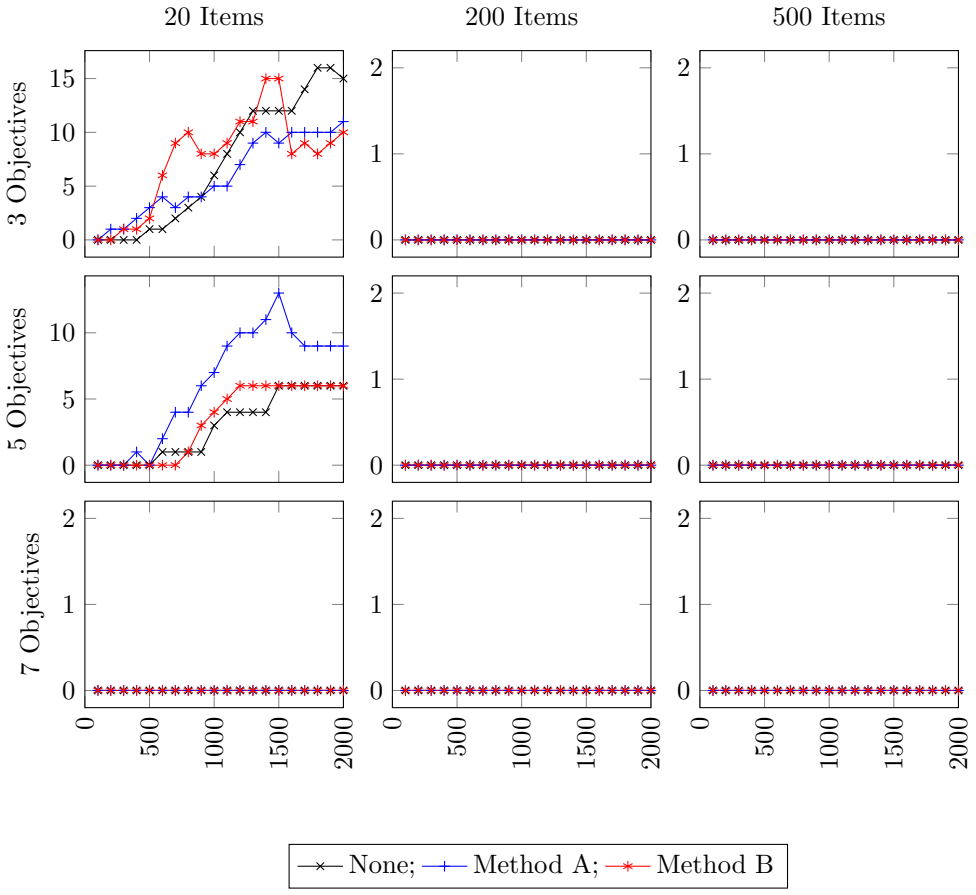


FIGURE 6.7: Development of number of unique solutions within constraints as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Minimum constraint for objective 1 set to $0.33 \cdot n$.

Fig. 6.9 shows the development of the number of solutions found within constraints for up to 20000 evaluations. Especially in the plots 5, 8 and 9 both constraints handling methods yield significantly more solutions compared to the run without constraint considerations. The curves again (as in the corresponding plot of series A.1) level at about 100 solutions which is supposedly because of the population size of the genetic algorithm. As expected, in general there are less solutions found after the same number of evaluations compared to series A.1. Noticeable is plot 2 for the problem with three objectives and 200 items. Here the constraint handling methods are not really favorable over the plain search algorithm with no constraint consideration. Also remarkable is plot 7 where maximally one solution was found. Comparing this plot with the development of the hypervolume one can conclude that it is always the same solution for the curve of method A. The solution found after 4000 evaluations with no constraint consideration (black curve) is a different one because the hypervolume measure also differs. It is not obvious why in the 20 items problems with three and five objectives considerably more

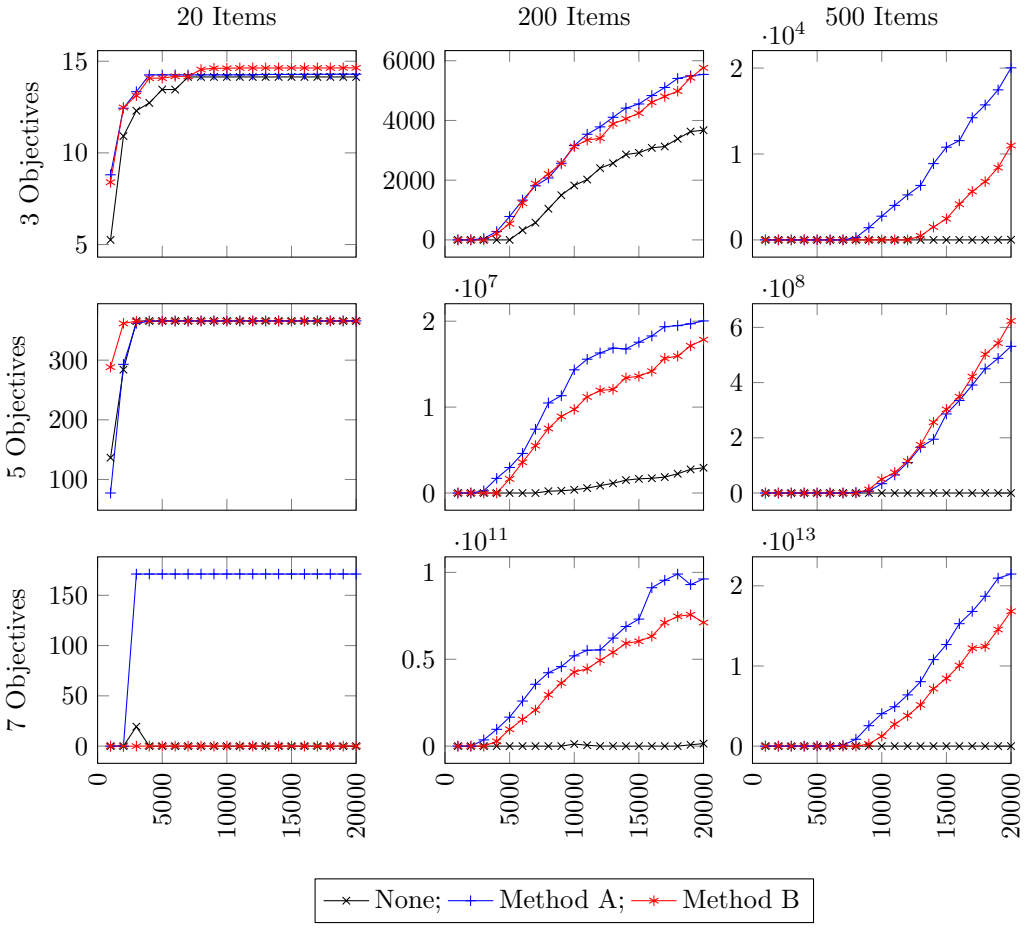


FIGURE 6.8: Development of constrained hypervolume as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Minimum constraint for objective 1 set to $0.33 \cdot n$.

solutions are found. The cause could be rooted in the random generation of the problem instance. Probably the values adding up to the two constrained objectives are lower for the seven objective problem by chance.

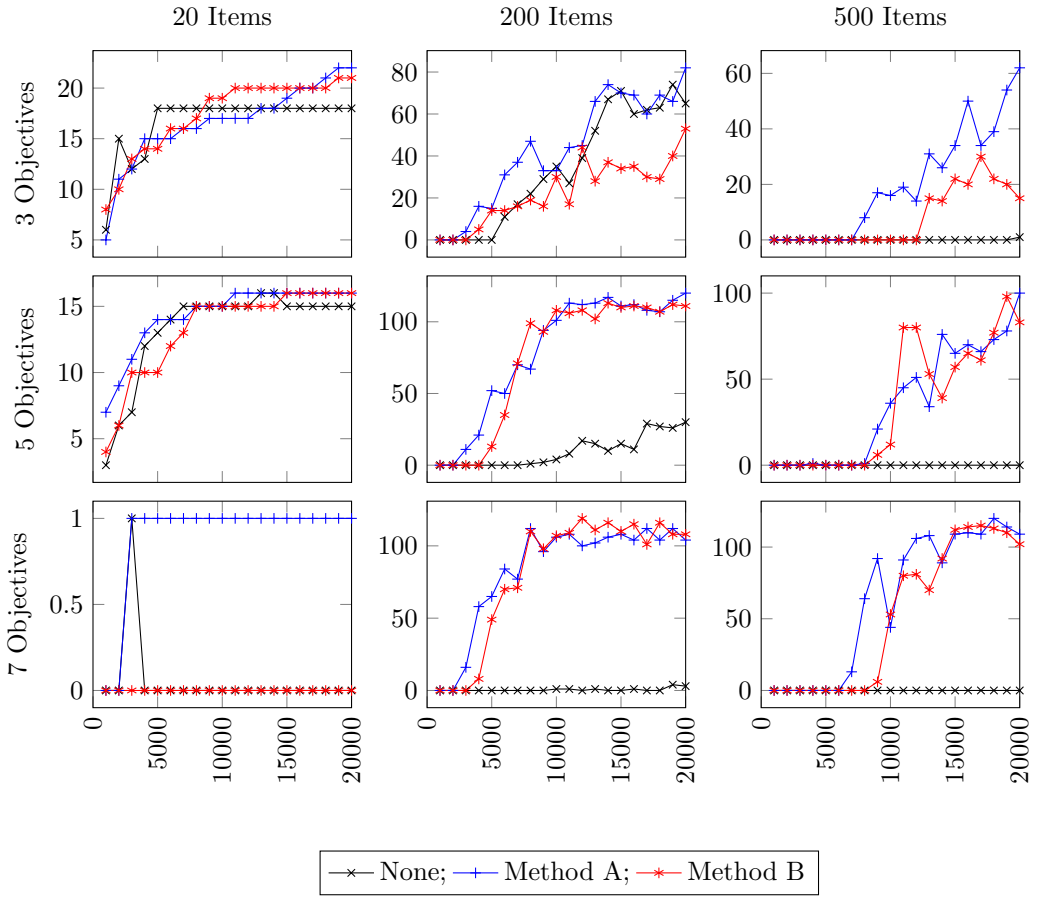


FIGURE 6.9: Development of of number of unique solutions within constraints as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Minimum constraint for objective 1 set to $0.33 \cdot n$.

6.3.3 Experiments A.3

The following experiments have two constraints in common. The first constraint is the same as in series A.1 and A.2 (minimum constraint on the first objective with value $0.33 \cdot n$). A second constraint is defined on objective two. It is a maximum constraint with value $0.30 \cdot n$.

Fig. 6.10 shows plots of the hypervolume for up to 2000 evaluations. The experiments with problems of 500 items again do not return solutions that comply with the constraints. However, compared to the experiments in A.2 with two minimum constraints for the experiments with 200 items as well as for the seven objective problem with 20 items the algorithm finds solutions. The curves of method A and B are exactly the same to the corresponding curves of series A.1. However, the curves of the experiments with no constraint consideration differ. The maximum constraint obviously does not have any impact on the search for these problems. Except for plots 2 and 7 the hypervolume

remains at 0 because no solutions are found within the constraints (according to Fig. 6.11). It is assumed that this is because the the search algorithm does not respect the maximum constraint and finds only solutions that already exceed this constraint. The hypervolume in the 20 items experiments is lower than in the respective experiments of series A.1. This is not surprising as the objective values of the second objective do not exceed the limit set by the maximum constraint.

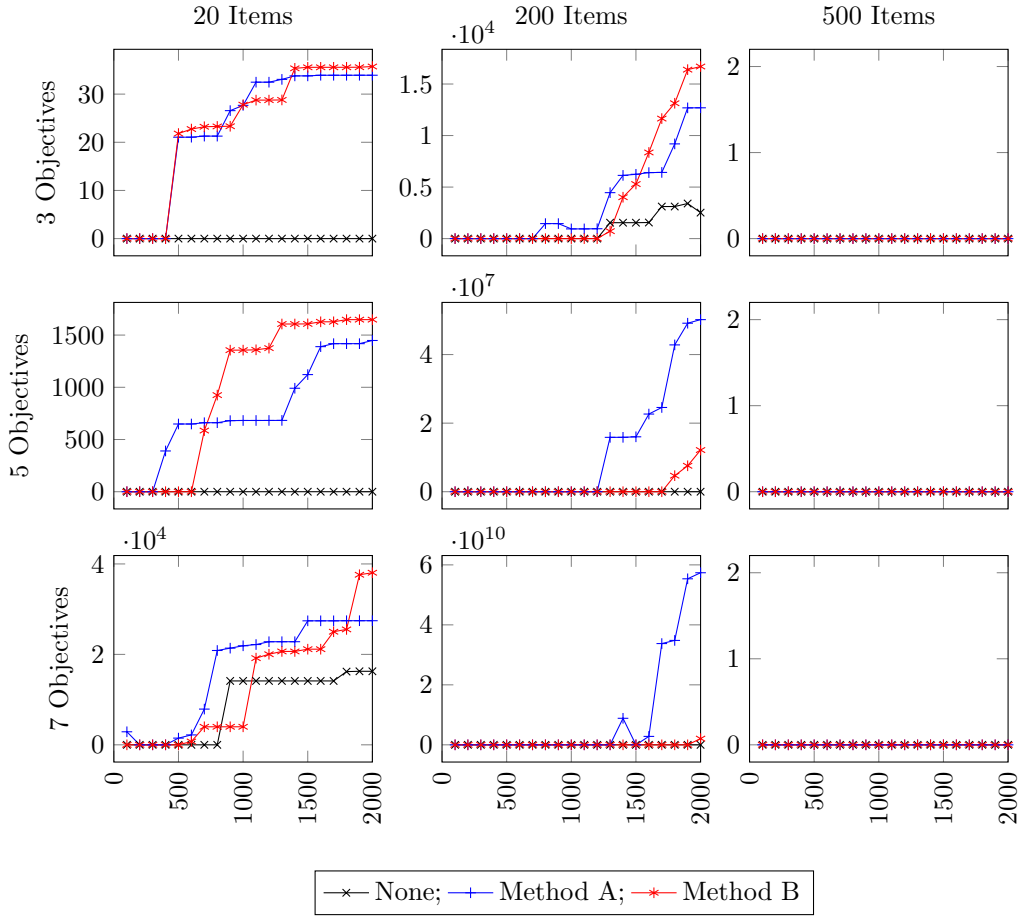


FIGURE 6.10: Development of constrained hypervolume as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Maximum constraint for objective 1 set to $0.3 \cdot n$.

The number of solutions in the constrained range are shown in Fig. 6.11 for up to 2000 evaluations. The number of solutions found in the 20 items experiments by for runs with constraint consideration are similar to the experiments with no maximum constraint of series A.1. Even some more solutions are found for the five objectives problem. The number of found solutions with the constrain handling method is exactly the same for 200 item problems as in series A.1, as mentioned before.

Fig. 6.12 and 6.13 show the development of hypervolume and number of solutions within constraints for up to 20000 solutions. The values for the runs without constraint

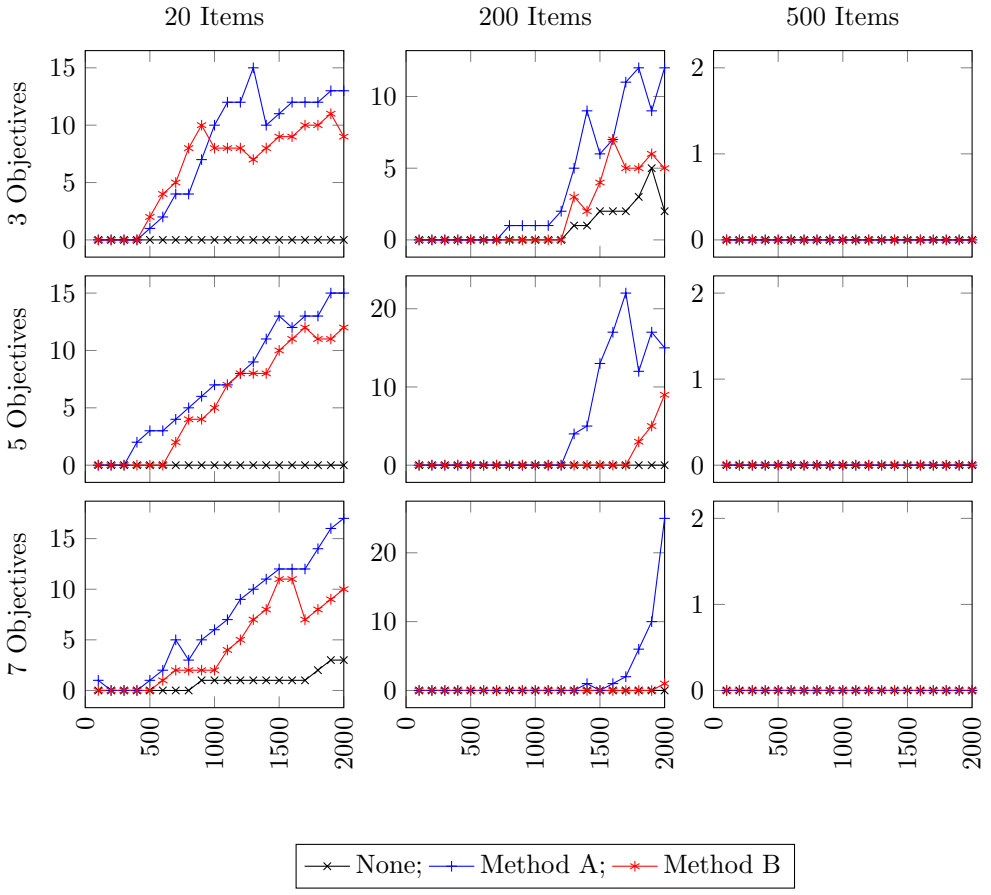


FIGURE 6.11: Development of number of unique solutions within constraints as function of number of evaluations in the range 0 to 2000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Maximum constraint for objective 1 set to $0.3 \cdot n$.

consideration are significantly lower than those of runs with methods A and B. Their behavior is not directly comparable to that of series A.1 and A.2 because the introduced maximum constraint has a special effect on the result. These curves show only the solutions that comply also with this maximum constraint. The search algorithm may however find solutions that exceed the limit set for this objective. Because maximization problems are considered the objective values tend to increase beyond the maximum constraint if it is not considered by the algorithm. The form of the first curve in plot 2 can be explained by the progress of the Pareto-front approximation. At first no solutions comply with the constraints because the objective values of the first objective are below the minimum constraint. As the algorithm proceeds by finding solutions with growing objective values the non-dominated front moves through a range where it consist of solutions that comply with both constraints (minimum constraint on first objective and maximum constraint on second objective). The algorithm then proceeds to find non-dominated fronts that consist of solutions with even higher objective values. After 5000

evaluations all the solutions comprising the non-dominated front have objective values for the maximum constrained objective that exceed the maximum constraint and are therefore not considered anymore.

In general the constraint handling methods are suitable to return more solutions that comply to the constraints also if a maximum constraint is used. The number of solutions again flattens at about 100 for certain experiments what could again be related to the population size parameter.

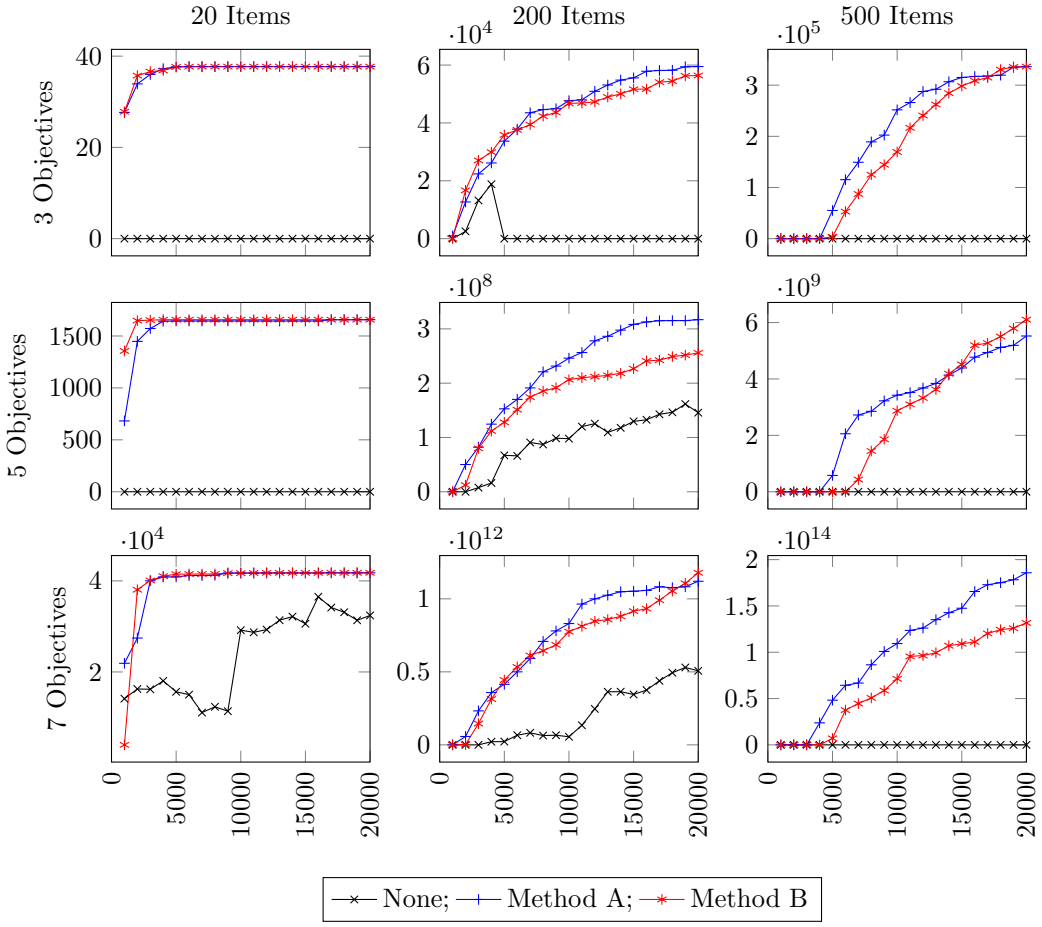


FIGURE 6.12: Development of constrained hypervolume as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Maximum constraint for objective 1 set to $0.3 \cdot n$.

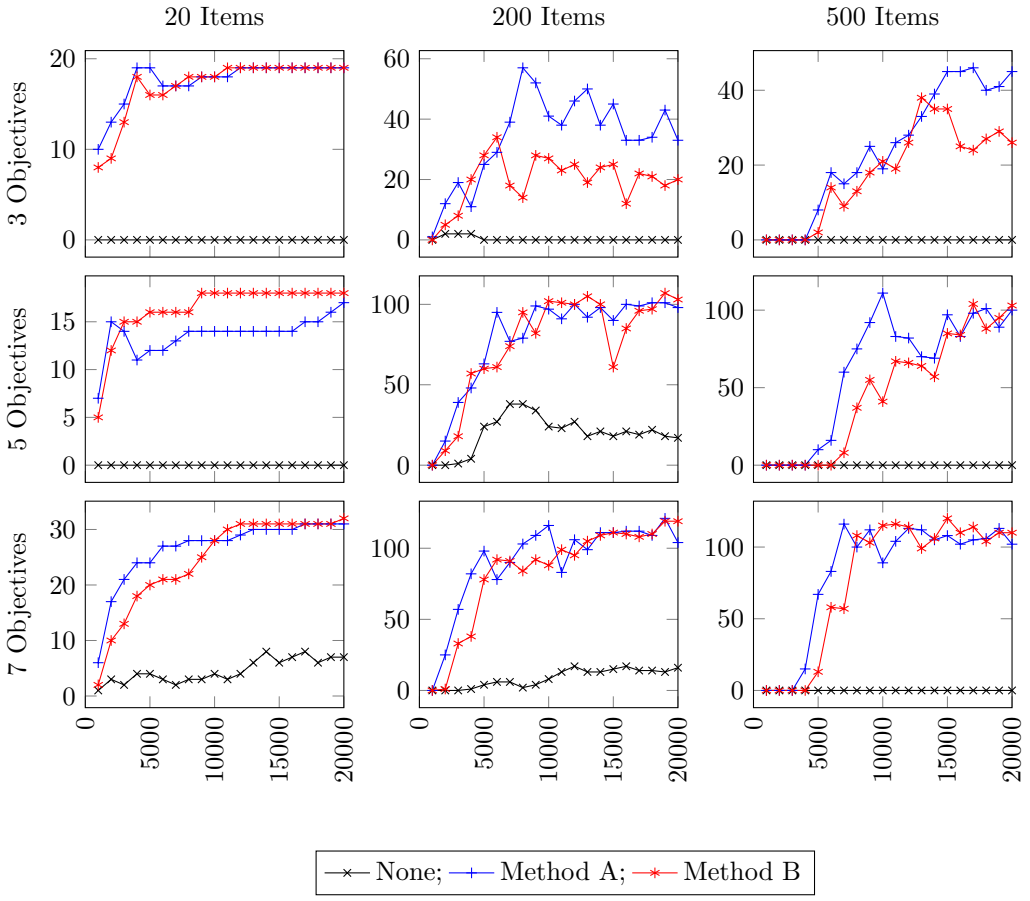


FIGURE 6.13: Development of of number of unique solutions within constraints as function of number of evaluations in the range 0 to 20000. Population size 100. Minimum constraint for objective 0 set to $0.33 \cdot n$. Maximum constraint for objective 1 set to $0.3 \cdot n$.

6.3.4 Comparison of Utopia and Nadir Vectors

As an example Fig. 6.14 shows the comparison of utopia and nadir vectors for the 200 items problem with five objectives (see appendix B for figures of other experiments). The three graphs show the resulting vectors for the different constraint settings of series A.1 to A.3. All the vectors of the returned solutions lie between the nadir and the utopia vector in the shaded area. The vectors resulting from the experiment run without constraint consideration (black) serve as a reference. Other than in the previous graphs where the measures have been applied to the solutions that comply with the user constraints the vectors have been derived from the plain solution sets without eliminating the solutions not adhering to the user constraints.

The plot in the middle shows that the minimum constraints effectively limit the values of the nadir vector. Also the maximum constraint in the third plot limits the utopia vector. This behavior is not surprising because the algorithm with employed user constraint

handling only returns solutions that are within the constraints. Comparing the third plot to the first one an increase of the maximum value of objective 3 can be observed. This could indicate that with setting a maximum constraint on an objective it in fact can be relaxed and enable the algorithm to find solutions with better values on other objectives. Although, at least for these experiments, the improvement is quite small and only for one objective. Also with other experiments no obvious indication that maximum constraints would enable better results for other objectives was found.

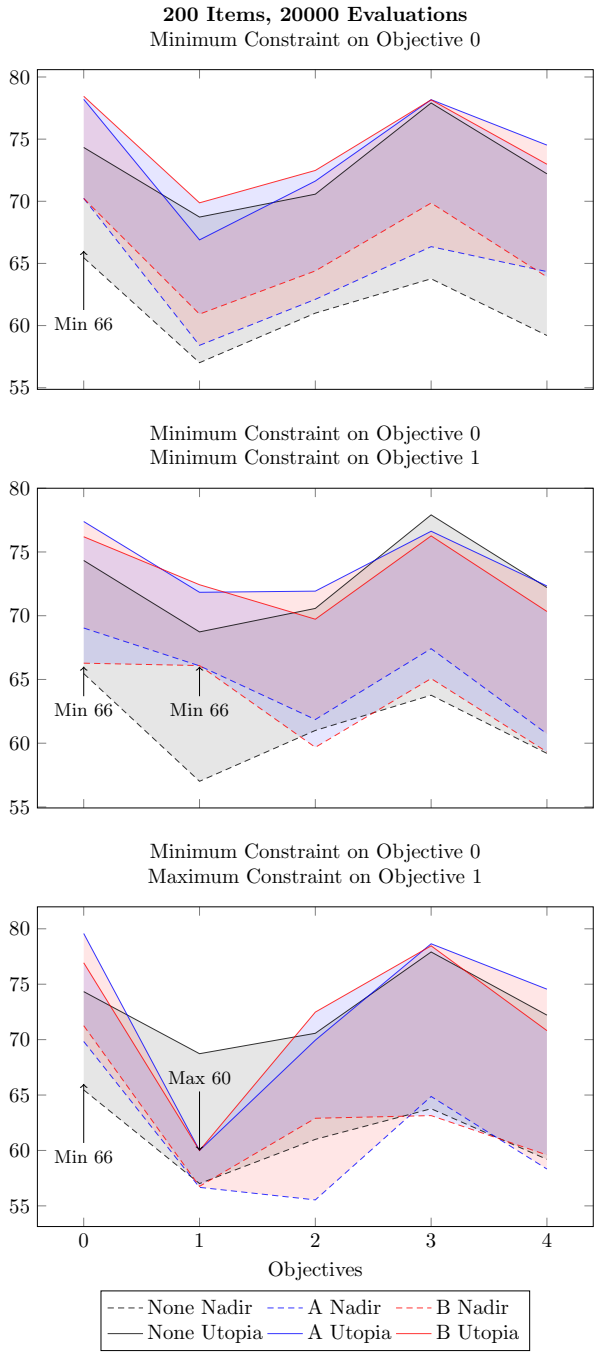


FIGURE 6.14: Utopia and nadir vectors after 20000 evaluations for experiments with five objectives and 200 items of series A.1 to A.3

6.4 Experiment Series B

This experiment series uses a population size of 500 individuals. In some experiments of series A the number of solutions within constraints flattened approximately at the population size. It is expected that for those experiments the metric would continue to increase because of the bigger population size.

Because for some experiments the hypervolume measure could not be computed in reasonable time (see section 5.5) only plots for the number of solutions are included here. Also no diagrams that detail the range of up to 2000 evaluations are included because only in the range of up to 20000 the data is meaningful.

When referring to the different plots of a figure with numbers the scheme according to Fig. 6.15 applies.

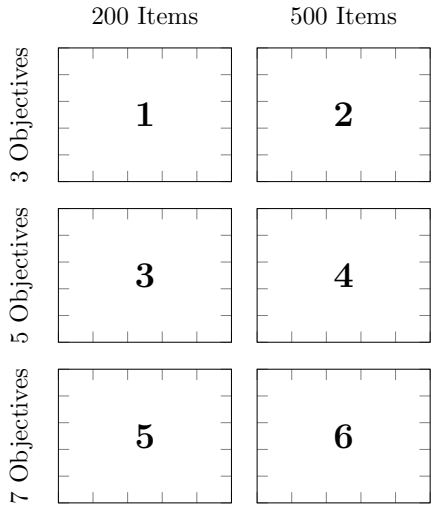


FIGURE 6.15: Numbering of individual plots

6.4.1 Experiments B.1

The following experiments have the same user constraints as the experiments of series A.2 (minimum constraints of $n \cdot 0.33$ for the first two objectives).

The plots 3 and 5 in Fig. 6.16 indeed show a higher maximum nuber of solutions that comply with the user constraints. Remarkable is that the curves for these experiments are steeper than the corresponding ones for population sizes of 100 individuals in series A.2. This means that more evaluations are necessary to find any valid solutions in the first place. Also for experiment 1 the after 20000 evaluations only about 30 solutions

are returned whereas with the population size of 100 valid solutions are found after only about 3000 evaluations and reach about 60 after 20000 evaluations (see Fig. 6.9). Experiments 2 and 4 do not produce practical output at all. In plot 6 a steep development is indicated but only after 19000 evaluations. The corresponding experiment with population size of 100 produces solutions much quicker.

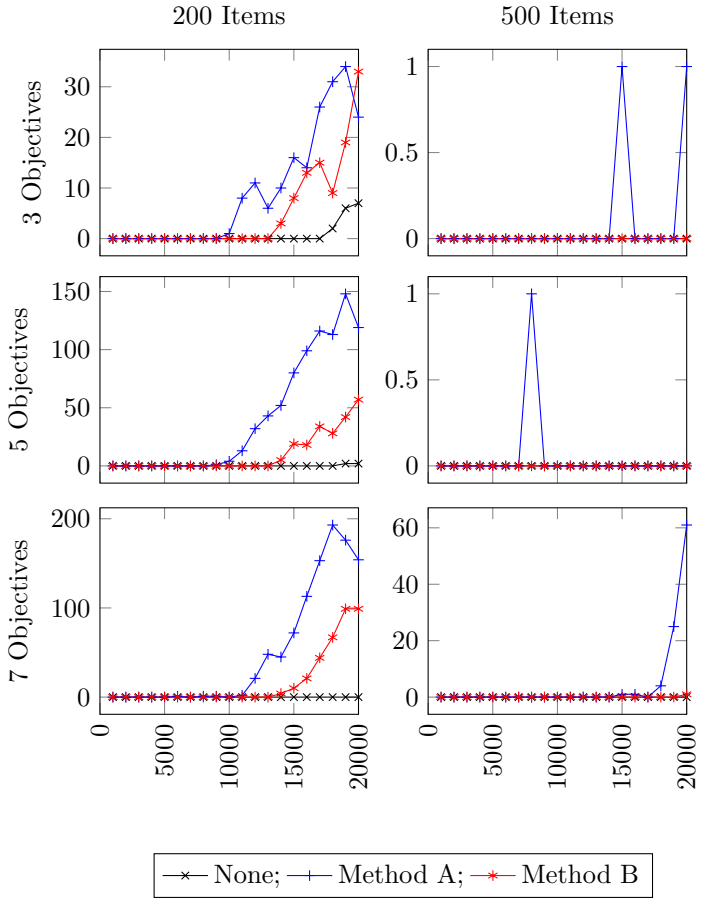


FIGURE 6.16: Development of of number of unique solutions within constraints as function of number of evaluations in the range 0 to 20000. Population size 500. Minimum constraint for objective 0 set to $0.33 \cdot n$. Minimum constraint for objective 1 set to $0.33 \cdot n$.

6.4.2 Experiments B.2

The following experiments have the same user constraints as the experiments of series A.3 (minimum constraints of $n \cdot 0.33$ for the first and maximum constraint of $n \cdot 0.30$ for the second objective).

Also in Fig. 6.17 it is obvious that the increased population size results in a later (in terms of necessary evaluations) discovery of solutions that lie within the user defined

constraints. Nevertheless again the measure achieves higher maximum values in the plots 3 and 5 compared with he equivalent experiments of series A.3.

The reason why the bigger population size results in solutions being returned later could be that the search algorithm calculates less generations. A bigger population size results in less generations calculated for the same amount of evaluations. Possibly the operations performed by the employed NSGAII in connection with the transition from one generation to the next (such as fitness assignment and selection) and their frequency are a reason for the observed results. Obviously the choice of this parameter is an important one because it has significant influence on the number of needed evaluations to find solutions of interest to the decision maker.

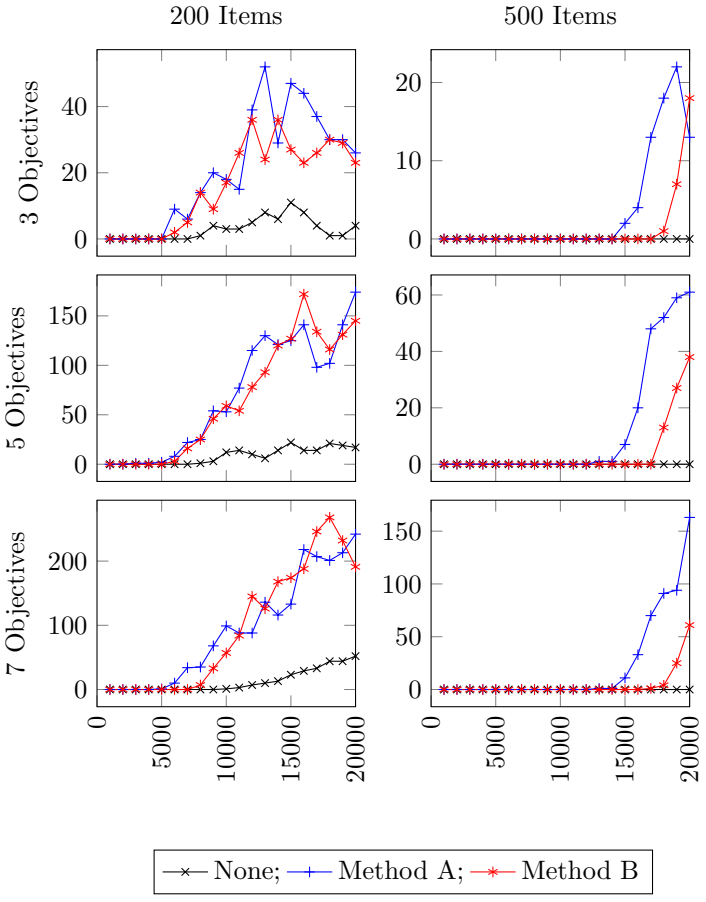


FIGURE 6.17: Development of of number of unique solutions within constraints as function of number of evaluations in the range 0 to 20000. Population size 500. Minimum constraint for objective 0 set to $0.33 \cdot n$. Maximum constraint for objective 1 set to $0.30 \cdot n$.

6.5 Summary

This chapter visualized the data of the validation experiments in the form of diagrams. The development of metrics of hypervolume and number of solutions within constraints for the different experiment series are discussed and compared in regards to the expected results.

In general the data complies with the expectations. Especially for experiments with problems of considerable complexity (five and more objectives with 200 and more knapsack items) the constraint handling methods promise a substantial advantage over the reference experiments with no constraint consideration. The population size has a considerable influence on the number of evaluations that are needed until the user is presented with solutions that adhere to the constraints.

Chapter 7

Conclusion

7.1 Introduction

This chapter presents the conclusions of the study by summarizing and discussing the experiment results. Also the research questions defined in section 1.5 are answered based on the evaluation of the appliaction prototype and results of experiments. Furthermore, limitations of the experiments and a real use case with its potential and risks are described. Finally subjects for further research are sugested.

7.2 Limitation of Experiments

The experiments used to validate the proposed methods only use static constraints. In a realistic use case a user would be able to dynamically alter the constraints by adding and removing limits and adjusting values. The effect of these dynamics maybe could be studied for example with use case scenarios solved by real users and appropriate statistical analysis.

Potentially the dynamic change of constraints will result in a quicker conversion to the preferred range as the decision maker can adapt constraints based on the current non-dominated set. Possibly this effect is strongest when the current non-dominated set contains a lot of solutions in regards to the population size.

A possible weakness of the dynamic change of constraints could be that the decision maker sets constraints too restrictively and thereby disadvantageously affects the search algorithm as no complying solutions can be found.

7.3 Discussion of Results

The analysis of the data produced through the experiments result in the following conclusions:

- In the vast majority of cases the constrained hypervolume and the number of unique solutions within user defined constraints achieve better values after an arbitrary number of evaluations when any of the two constraint handling methods is applied.
- The advantage of the constraint handling methods is especially significant for experiments with five and more objectives and for knapsack problems with 200 or more items.
- A very clear advantage of method A over method B or vice versa cannot be derived from the produced data. However, in a slim majority of cases metrics of experiments using method A exceed those of method B. This is a bit surprising as method B makes a graduated distinction of user and problem constraints that was expected to render better results.
- A relaxation effect of maximum constraints that enables the improvement of other objectives could not be identified. The maximum constraints, however, enable the decision maker to restrict the objective space what could be helpful to reduce the alternatives to decide on.
- The population size of the genetic algorithm has a significant impact on the number of evaluations needed to find solutions of interest.

7.4 Research Questions

7.4.1 Subquestions

a. How can a tool display a Pareto optimal set of solutions?

There are previous studies on the visualization of Pareto optimal solution sets (see for example Miettinen, 2014). One important characteristic of these visualization techniques is number of objectives that can reasonably be supported. In a software tool it is a matter of programming effort to represent the same data in various visual forms. In the developed application prototype a variant of the spider or radar chart has been chosen for this purpose. This chart type was chosen because

it is suitable for more than three objectives and it is convenient for the specification of user constraints.

b. How can a Pareto optimal set be visualized for intuitive perception?

Although there are some generally applicable principles what improves and degrades the perceptibleness of a visualization of data it is also a matter of personal preference. The data to be represented in the prototype tool is dynamic due to the nature of the search algorithm. It constantly produces new solutions to which the visualization must adapt. Usually all axes of a spider chart have the same scaling and and a center point at zero. The implemented spider chart supports an individual scale and origin for the axes. This enables the decision maker to visually distinguish solutions when the spread of objective values becomes smaller as the preferred range is approached. It also allows to display objectives of different dimension and variance. Important for intuitive perception is that the user is not distracted by crowded data points or visual elements that do not improve the comprehensibility of the data.

c. How can a decision maker visually specify preference information?

In the presented application the decision maker can specify the preference information directly in the diagram where also the solutions are displayed. The diagram serves as a visual control for displaying the solution data and for input of preference information. The preference information consists of maximum and minimum values that the user can define for objectives. Various approaches for the specification of user preference information using trade-offs, weights or classification are described for example by Miettinen et al. (2008). Often the required user input for these approaches is hard to determine. Also most of these methods are difficult to visualize or are unsuitable for graphical specification. The preference specification approach presented in this study is simple and the meaning of the preference information required by the user is easy to understand and visually perceptible. This makes this method suitable for non-expert decision makers.

d. How can preference information be used to guide a search algorithm?

It depends on the form of the preference information and the characteristics of the search algorithm how it can be directed. In the developed tool the user defines constraints on the objectives. Two constraint handling methods have been described, implemented and tested. These methods influence certain operations of the NSGAII algorithm to guide it.

e. How can this preference information be represented graphically?

The user defined constraints are displayed in the spider diagram as draggable handles on the axes that represent the objectives. These handles serve as both, input control as well as graphical representation of current values. An additional numeric table serves as reference for number and exact values of constraints.

7.4.2 Main Questions

1. **How should a tool be designed to support multi-criteria decision making?**

   The presented application prototype consists of three main components. The user interacts with a *graphical interface* that enables the visual specification of preference information. It also presents solutions in numerical and visual formats and contains elements for the control of an *optimization algorithm*. The algorithm is applied to the multi-objective problem and solutions are presented to the decision maker through the user interface. Of specific importance is the *preference handling system*. It defines the mechanism how the preference information is specified and how it controls the search algorithm.

2. **How can preference information identify a best compromise from a Pareto optimal set?**

   The study compared two methods to handle the preference information input by the user. This input is provided in the form of constraints that the user defines on objectives. For every objective the user can set a maximum and a minimum value and thereby instruct the search algorithm to find solutions that comply with these constraints. The main advantage of this kind of user input over other methods (such as trade-off or classification based methods) is that its meaning is easier to understand and simpler to determine by the decision maker and also suitable for visualization.

   Both of the studied constraint handling methods operate by manipulating the dominance relationship between solution candidates and indirectly influence the fitness of solutions. The experiments indicate that both methods are in general beneficial to find more and better solutions according to the user defined constraints than when the algorithm does not consider these constraints. That means that less iterations are needed to find non-dominated solutions that adhere to the decision maker's preferences.

7.5 Suggestions for Improvements and Further Research

7.5.1 Algorithm

The fitness of a solution is an important factor when it comes to the selection operations in the genetic algorithm. In the NSGAII algorithm this fitness is based on dominance and crowding distance. The definition of the dominance relationship has been used and adapted to account for the user defined constraints. There are namely two selection operations in the NSGAII algorithm. The first is the binary tournament for the selection of two parent solutions for crossover and the second is the selection of the new parent generation from the joint solution pool of the previous parent generation and the offspring generation. In the second case a ranking is performed that assigns each solution a domination rank. The non-dominated solutions get a rank of 0, rank 1 is assigned to the solutions that are only dominated by solutions of rank 0, rank 2 to solutions dominated only by solutions of rank 0 and 1 and so forth. Solutions of a common rank build a front. To form the new parent generation solutions with the lowest rank are selected and removed from the available pool. When there are more solutions of lowest rank available than needed to complete the new parent solution set the solutions of lowest rank with the best crowding distances are selected.

When the user defined constraints influence the domination relationship the ranking is influenced as well. As this ranking only takes place for the selection of the new parent generation the user constraints at the moment of the selection for the new parent generation have a high influence on the genetic pool. At the moment when the new parent generation is defined half of the evaluated solutions are discarded. Considering that the user is able to change the constraints dynamically this could mean that solutions are lost that could have been preserved with different user constraints. The fitness of a solution is dependent on the values of the constraints. It could be beneficial to study the influence of dynamics of the user constraints and design an elitism mechanism that considers these dynamics.

7.5.2 Application support for evolutionary trees

In the course of the study it became apparent that the application would benefit and gain usability from an undo or history function. With the possibility to rewind the algorithm execution to an earlier point in time with its then current generation, Pareto front and user constraints the user could explore what-if situations by changing the constraints and let the algorithm calculate a new evolutionary branch with different conditions (Fig.

7.1). One could also think of merging solutions from different branches to create manual elitist solution sets.

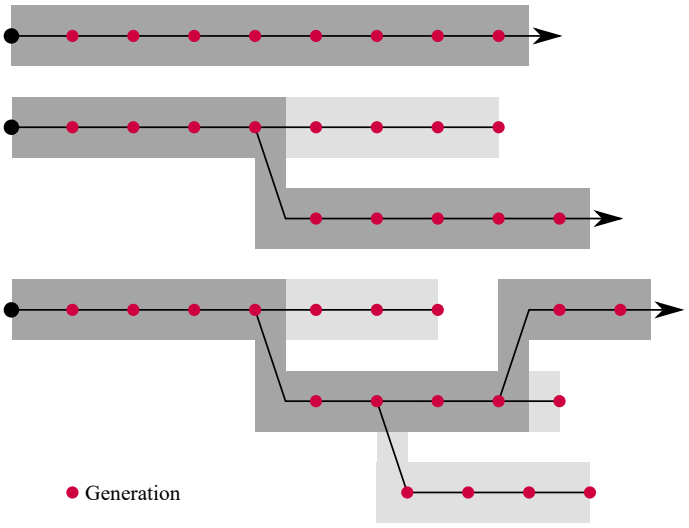


FIGURE 7.1: Tree of different evolutionary branches

7.6 Summary

The conclusion discusses the results derived from the validation experiments and provides answers to the research questions. The usage of constraints on objective values enables the design of a decision support tool that allows the visual specification of preference information in an interactive way. The meaning of the required user input in the form of constraints on objectives is not difficult to understand and makes the method suitable for decision makers with limited knowledge on multi-objective optimization. The employed validation experiment design does not allow a specific study on how dynamic changes to the preference information influences the search. However, according to the analysis of validation experiments both proposed constraint handling methods are suitable for directing a genetic algorithm.

Bibliography

Carlos Coello. A Short Tutorial on Evolutionary Multiobjective Optimization. *Evolutionary Multi-Criterion Optimization*, 1993(2508):21–40, 2001. doi: 10.1007/3-540-44719-9{_}2. URL `http://www.springerlink.com/index/2QDWCX5W3463LLQM.pdf`.

Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb. Evolutionary Algorithms for Multiobjective Optimization : Methods and Applications. (30):1–134, 1999.

Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. *Evolutionary Computation*, 535:3–37, 2004. ISSN 0075-8442. doi: 10.1007/978-3-642-17144-4{_}1. URL `http://www.springerlink.com/index/P6G209273G805278.pdf`.

Ghada Hassan. Multiobjective genetic programming for financial portfolio management in dynamic environments. *Solutions*, 2010. URL `http://eprints.ucl.ac.uk/20456/`.

J Dobeš, J Michal, and V Biolkova. Multiobjective Optimization for Electronic Circuit Design in Time and Frequency Domains. *Radioengineering*, 22(1):136–152, 2013.

Massimiliano Caramia and Paolo Dell'Olmo. *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms.* Springer Science & Business Media, 2008.

David C Major. *Multiobjective water resource planning*, volume 4. American Geophysical Union, 1977.

Andrzej Osyczka. Multicriteria optimization for engineering design. *Design optimization*, 1:193–227, 1985.

Hans L. Trinkaus and Thomas Hanne. KnowCube: A visual and interactive support for multicriteria decision making. *Computers and Operations Research*, 32(5):1289–1309, 2005. ISSN 03050548. doi: 10.1016/j.cor.2003.11.010.

Lidiane Sartini De Oliveira and Sezimária F. P. Saramago. Multiobjective optimization techniques applied to engineering problems. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 32(1):94–105, 2010. ISSN 1678-5878. doi: 10.1590/S1678-58782010000100012.

S S Rao. *Engineering Optimization: Theory and Practice*. Wiley, 2009. ISBN 9780470183526. URL `https://books.google.ch/books?id=YNt34dvnQLEC`.

M Schniederjans. *Goal Programming: Methodology and Applications: Methodology and Applications*. Springer US, 2012. ISBN 9781461522294. URL `https://books.google.ch/books?id=uODTBwAAQBAJ`.

Dylan F Jones and Mehrdad Tamiz. Goal programming in the period 1990–2000. In *Multiple Criteria Optimization: State of the art annotated bibliographic surveys*, pages 129–170. Springer US, 2002.

Dylan Jones and Mehrdad Tamiz. *Practical goal programming*, volume 141. Springer, 2010.

Melanie Mitchell. An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, ...*, page 162, 1996. ISSN 08981221. doi: 10.1016/S0898-1221(96)90227-8.

Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski. *Multiobjective optimization: interactive and evolutionary approaches*, volume 5252. 2008. ISBN 978-3-540-88907-6. doi: 10.1007/978-3-540-88908-3.

K Deb, S Pratab, S Agarwal, and T Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NGSA-II. *IEEE Transactions on Evolutionary Computing*, 6(2): 182–197, 2002.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2001. doi: 10.1.1.28.7571.

T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, 2:878–885, 2003. doi: 10.1109/CEC.2003.1299759.

David a Van Veldhuizen and Gary B Lamont. Evolutionary Computation and Convergence to a Pareto Front. *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, 1998. URL `http://www.lania.mx/{~{}}ccoello/EMOO/vanvel2.ps.gz`.

Piotr Czyzak and Andrzej Jaszkiewicz. Pareto simulated annealing. In *Multiple Criteria Decision Making*, pages 297–307. Springer, 1997.

Jason R Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, 1995.

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. ISSN 1089-778X. doi: 10.1109/4235.797969.

Jin Wu and Shapour Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 2001.

Kaisa Miettinen. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, 36(1):3–37, 2014. ISSN 01716468. doi: 10.1007/s00291-012-0297-0.

Kaisa Miettinen, Francisco Ruiz, and Andrzej P Wierzbicki. Introduction to multiobjective optimization: interactive approaches. In *Multiobjective Optimization*, pages 27–57. Springer, 2008.

Johannes Jahn. *Vector Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-17004-1. doi: 10.1007/978-3-642-17005-8. URL `http://www.springerlink.com/index/10.1007/978-3-642-17005-8`.

R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev. Linear Programming With Multiple Objective Functions: Step Method (Stem). In *Mathematical Programming*, volume 1, pages 366–375. North-Holland Publishing Company, 1971.

Kaisa Miettinen and Marko M. Mäkelä. Interactive MCDM Support System in the Internet. In theodor J. Stewart and Robin C. van den Honert, editors, *Trends in Multicriteria Decision Making*, pages 424–433. Springer, 1998.

Mark Saunders and Paul Tosey. The Layers of Research Design. pages 58–59, 2012.

V K Vaishnavi and W Kuechler. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition*. CRC Press, 2015. ISBN 9781498715263. URL `https://books.google.ch/books?id=OOE{_}CQAAQBAJ`.

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

Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — A comparative case study. (September):292–301, 1998. doi: 10.1007/BFb0056872.

Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. *Evolutionary Multi-Criterion Optimization*, 4403:862–876, 2007. ISSN 03029743. doi: 10.1007/978-3-540-70928-2{_}64. URL `http://dx.doi.org/10.1007/978-3-540-70928-2{_}64`.

Christian Feuersänger. Manual for Package pgfplots. *URL http://www. ctan. org/tex-archive/help/Catalogue/entries/pgfplots. html. Probablement install{é} dans votre syst{è}me sous le nom pgfplots. pdf*, 17, 2011.

Statement of Authenticity

I, Benjamin Keller, hereby confirm that this report was performed autonomously using only the sources, aids and assistance stated in the report, and that quotes are readily identifiable as such.

Signed: Benjamin Keller

Date: 31st of July 2016

Appendix A

Source Code

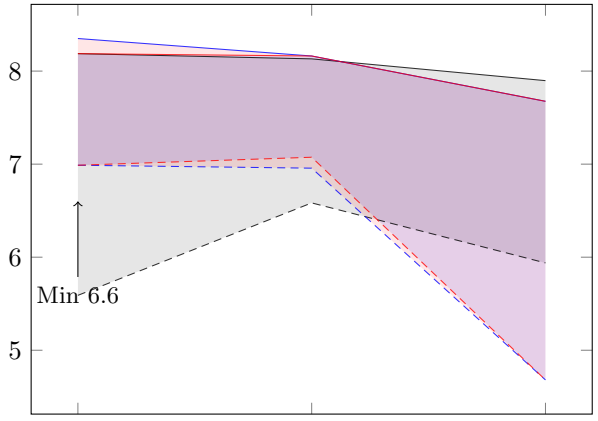
The source code written in the course of the thesis is part of this work. It is available as a Git repository at `https://github.com/eaglewings/GUIFMODS.git`.

Appendix B

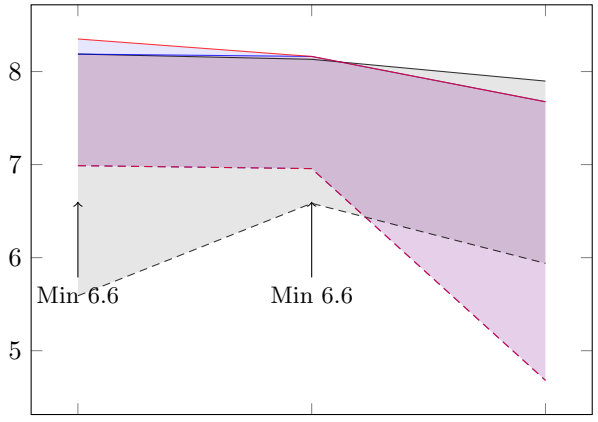
Additional Results

B.1 Visualization of Utopia and Nadir Vectors for Experiment Series A.1 to A.3

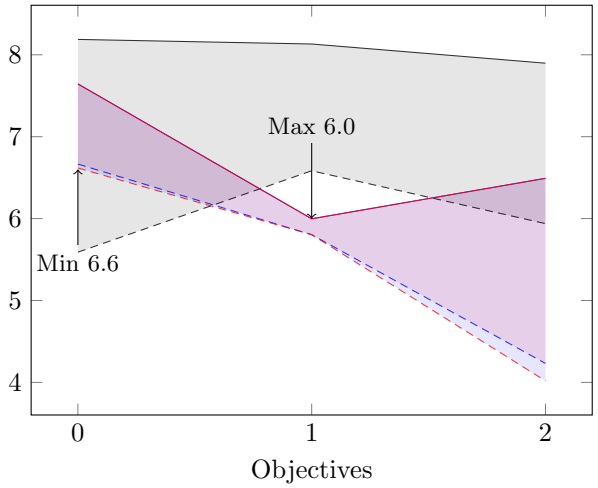
20 Items, 20000 Evaluations
Minimum Constraint on Objective 0

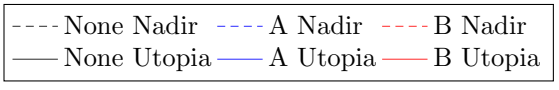


Minimum Constraint on Objective 0
Minimum Constraint on Objective 1



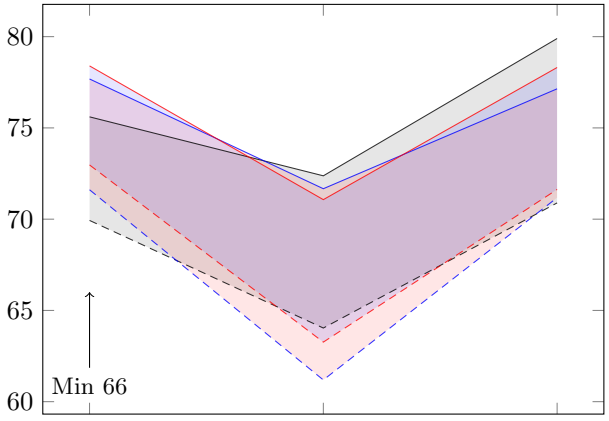
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1



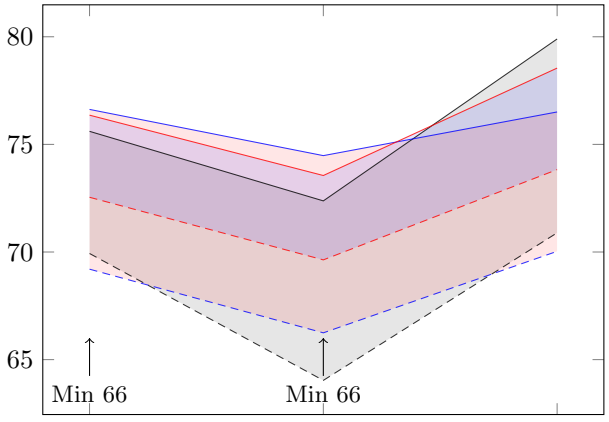


200 Items, 20000 Evaluations

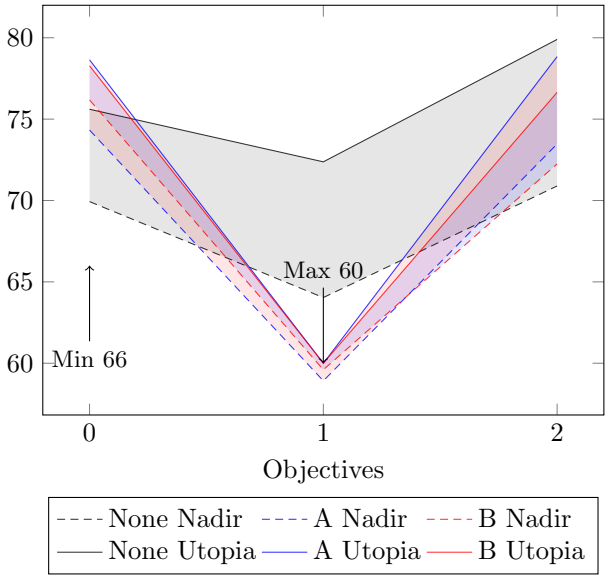
Minimum Constraint on Objective 0



Minimum Constraint on Objective 0
Minimum Constraint on Objective 1

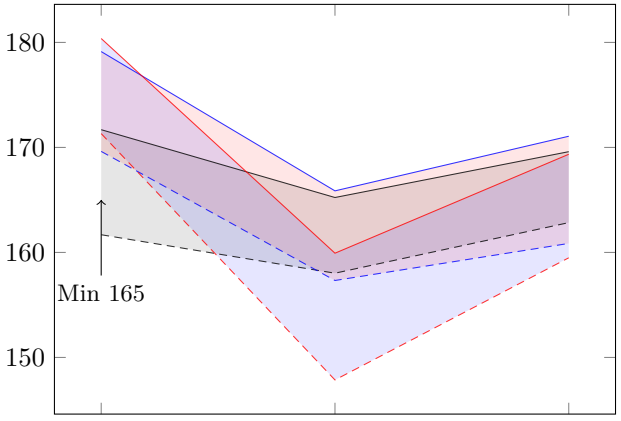


Minimum Constraint on Objective 0
Maximum Constraint on Objective 1

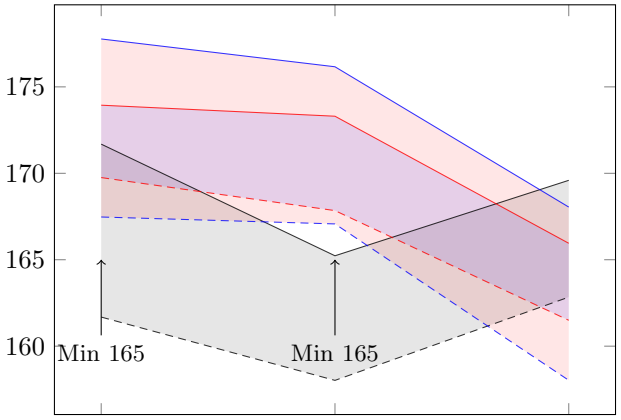


500 Items, 20000 Evaluations

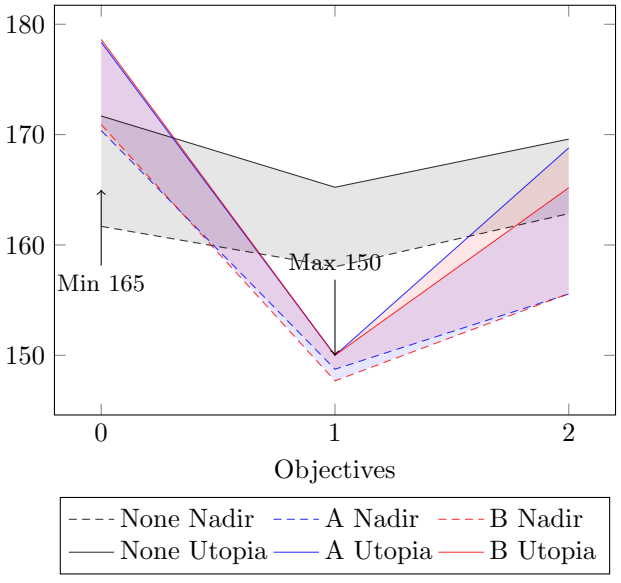
Minimum Constraint on Objective 0



Minimum Constraint on Objective 0
Minimum Constraint on Objective 1

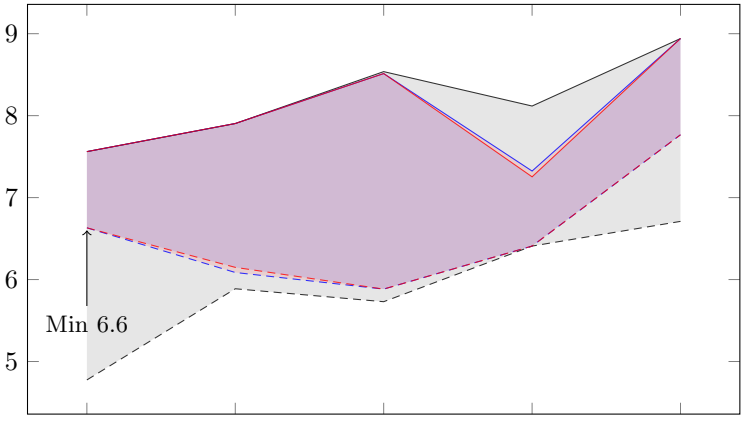


Minimum Constraint on Objective 0
Maximum Constraint on Objective 1

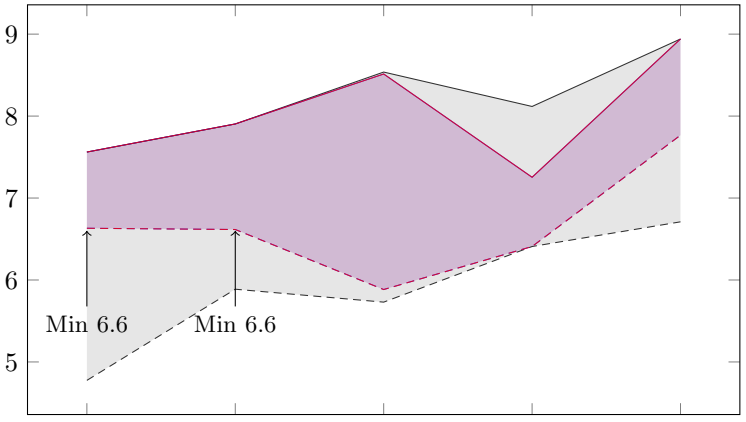


20 Items, 20000 Evaluations

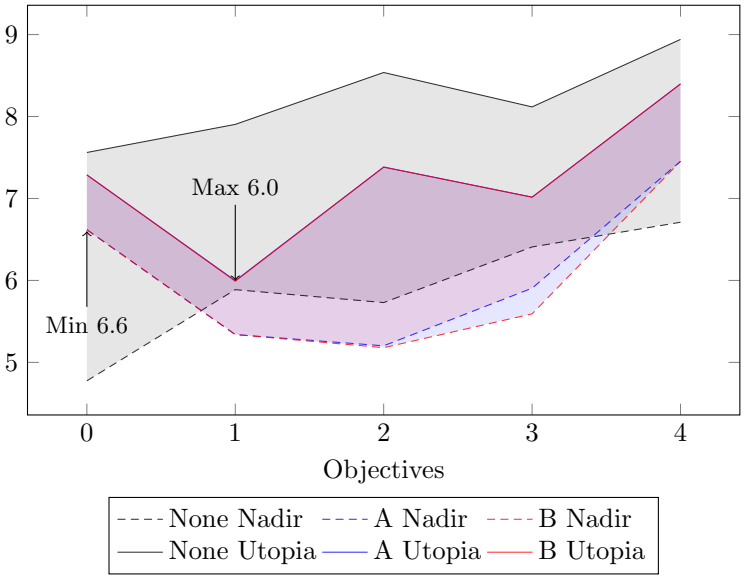
Minimum Constraint on Objective 0

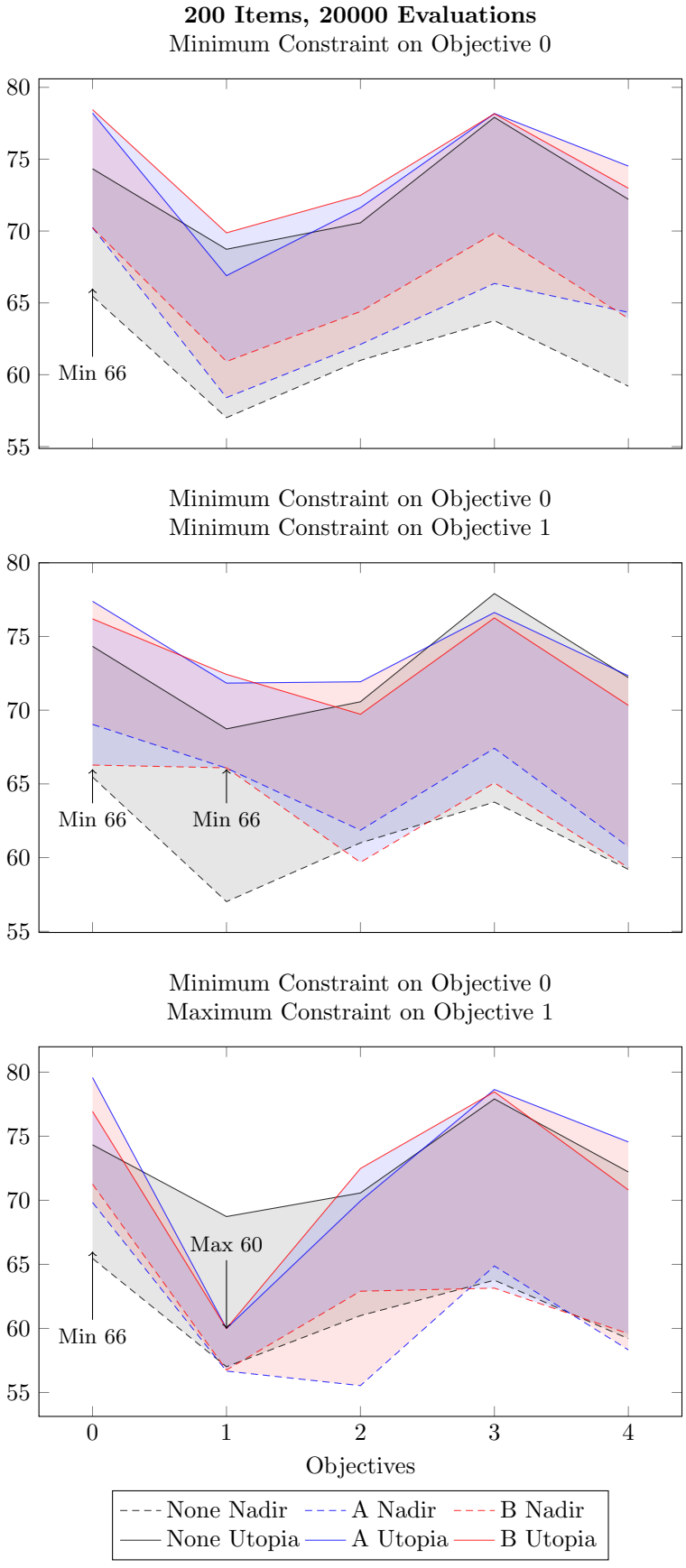


Minimum Constraint on Objective 0
Minimum Constraint on Objective 1



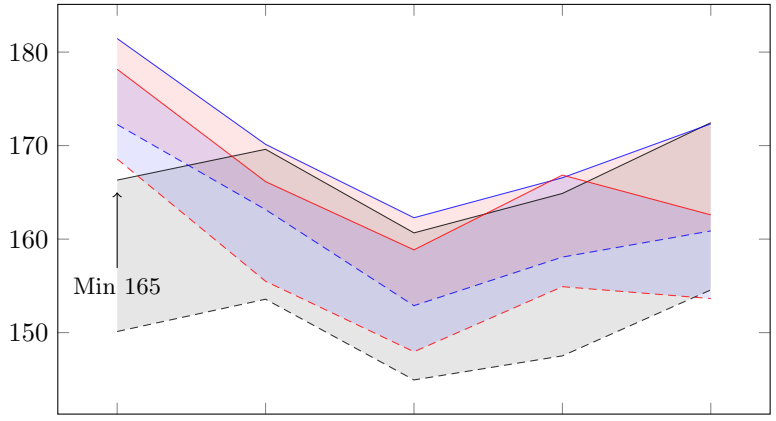
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1



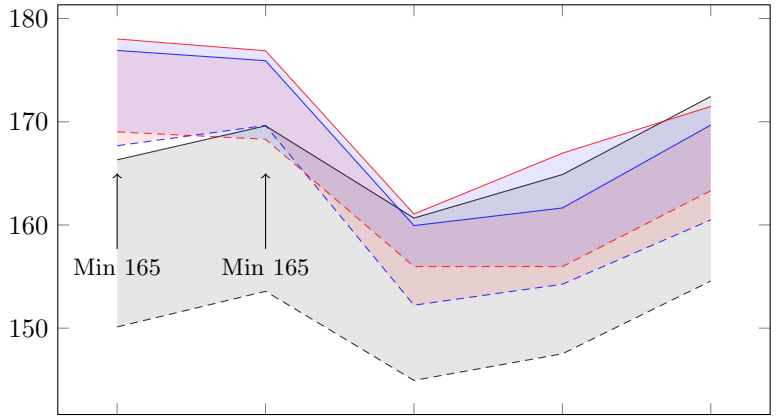

200 Items, 20000 Evaluations
Minimum Constraint on Objective 0
80
75
70
65
60
55
Min 66
Minimum Constraint on Objective 0
Minimum Constraint on Objective 1
Min 66
Min 66
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1
Max 60
Min 66
0
1
2
3
4
Objectives
None Nadir
A Nadir
B Nadir
None Utopia
A Utopia
B Utopia

500 Items, 20000 Evaluations

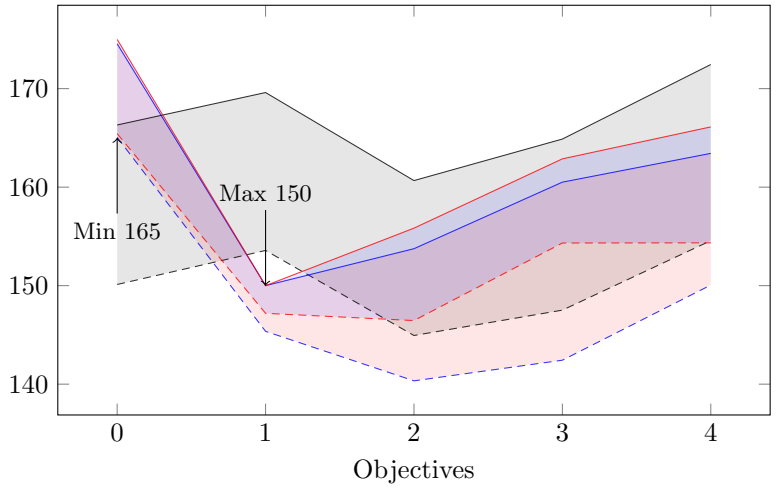
Minimum Constraint on Objective 0



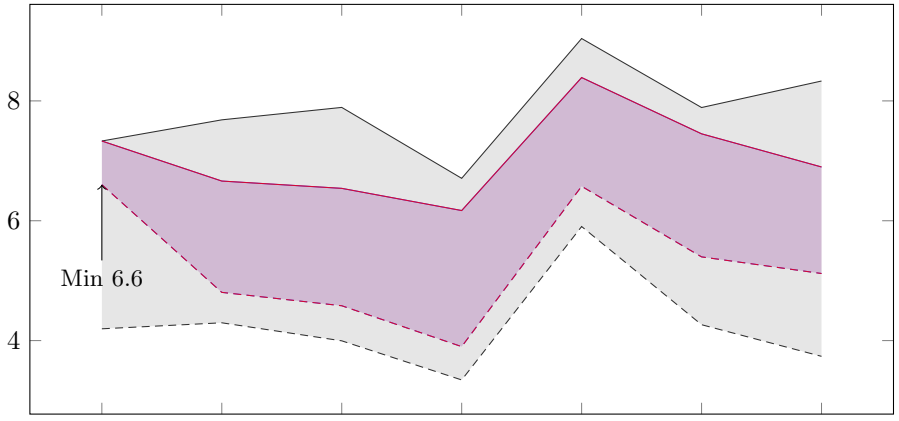
Minimum Constraint on Objective 0
Minimum Constraint on Objective 1



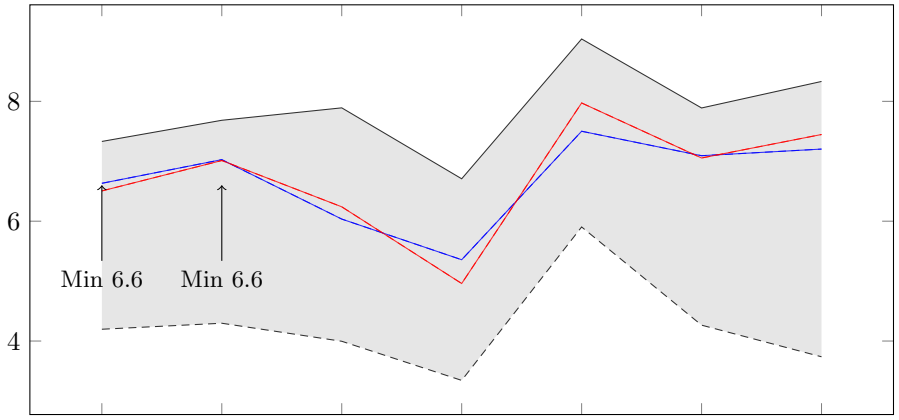
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1



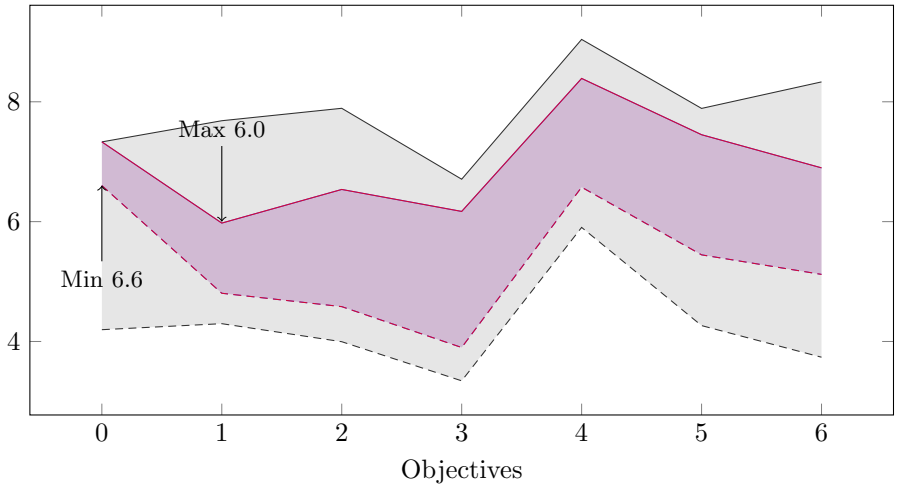
20 Items, 20000 Evaluations
Minimum Constraint on Objective 0

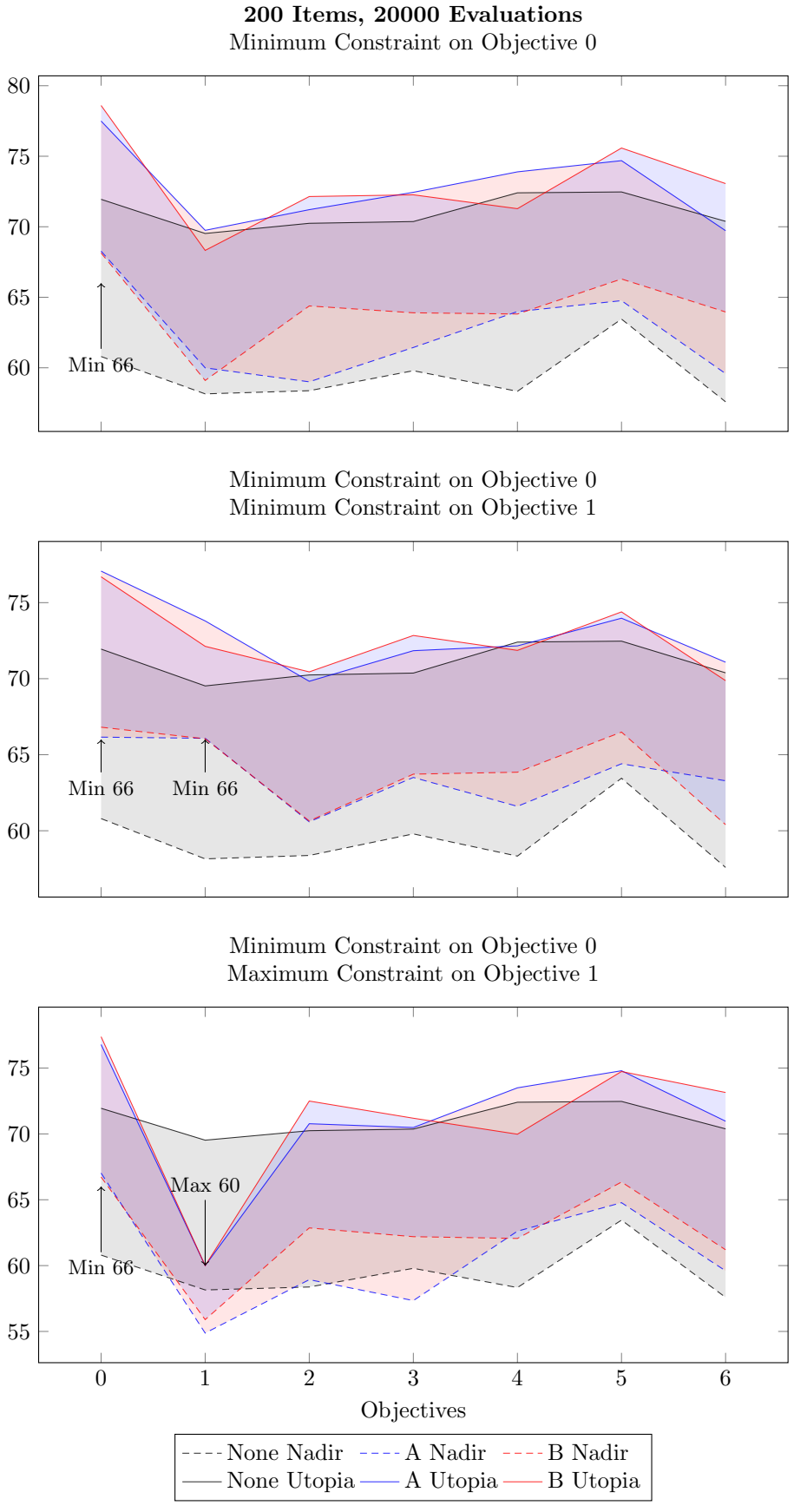


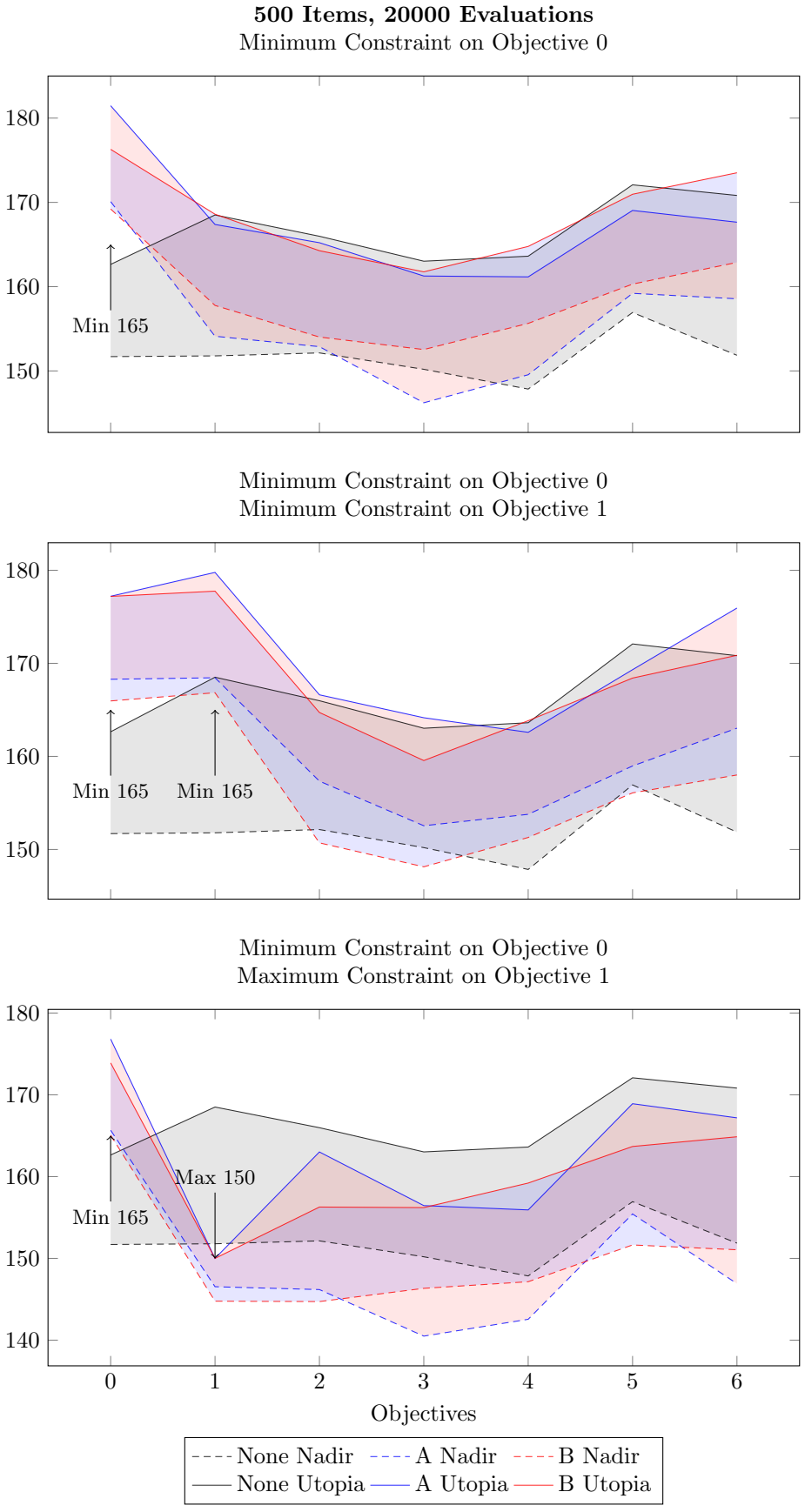
Minimum Constraint on Objective 0
Minimum Constraint on Objective 1



Minimum Constraint on Objective 0
Maximum Constraint on Objective 1




200 Items, 20000 Evaluations
Minimum Constraint on Objective 0
80
75
70
65
60
Min 66
Minimum Constraint on Objective 0
Minimum Constraint on Objective 1
75
70
65
60
Min 66
Min 66
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1
75
70
65
60
55
Max 60
Min 66
0
1
2
3
4
5
6
Objectives
None Nadir
A Nadir
B Nadir
None Utopia
A Utopia
B Utopia


500 Items, 20000 Evaluations
Minimum Constraint on Objective 0
180
170
160
150
Min 165
Minimum Constraint on Objective 0
Minimum Constraint on Objective 1
180
170
160
150
Min 165
Min 165
Minimum Constraint on Objective 0
Maximum Constraint on Objective 1
180
170
160
150
140
Max 150
Min 165
0
1
2
3
4
5
6
Objectives
None Nadir
A Nadir
B Nadir
None Utopia
A Utopia
B Utopia

B.2 Table of Approximated Utopia and Nadir Vectors of Example Problems

TABLE B.1: Nadir and utopia vectors of the example problems after 20000 evaluations with population sizes of 100 and 500 individuals. The percentage values relate the objective values to the number of items.

Population size		100												500							
Items	Objective	Nadir		Utopia		Nadir		Utopia		Nadir		Utopia		Nadir		Utopia		Nadir		Utopia	
20	O1	5,59	27,9%	8,19	40,9%	4,77	23,9%	7,56	37,8%	4,75	23,7%	7,33	36,7%	5,59	27,9%	8,35	41,8%	4,77	23,9%	7,56	37,8%
	O2	6,58	32,9%	8,13	40,7%	5,89	29,4%	7,90	39,5%	4,57	22,8%	7,68	38,4%	6,58	32,9%	8,18	40,9%	5,95	29,8%	7,90	39,5%
	O3	5,94	29,7%	7,90	39,5%	5,73	28,6%	8,54	42,7%	3,85	19,3%	7,85	39,3%	4,68	23,4%	7,90	39,5%	5,73	28,6%	8,54	42,7%
	O4	-	-	-	-	6,41	32,0%	8,12	40,6%	3,07	15,3%	6,81	34,0%	-	-	-	-	6,41	32,0%	8,12	40,6%
	O5	-	-	-	-	6,71	33,5%	8,94	44,7%	6,19	31,0%	9,04	45,2%	-	-	-	-	6,96	34,8%	8,94	44,7%
	O6	-	-	-	-	-	-	-	-	4,72	23,6%	7,89	39,4%	-	-	-	-	-	-	-	-
	O7	-	-	-	-	-	-	-	-	4,06	20,3%	8,63	43,1%	-	-	-	-	-	-	-	-
200	O1	69,93	35,0%	75,61	37,8%	65,46	32,7%	74,33	37,2%	60,80	30,4%	71,95	36,0%	64,29	32,1%	73,25	36,6%	63,10	31,5%	72,76	36,4%
	O2	64,04	32,0%	72,37	36,2%	57,01	28,5%	68,73	34,4%	58,15	29,1%	69,52	34,8%	56,72	28,4%	68,18	34,1%	54,72	27,4%	66,29	33,1%
	O3	70,89	35,4%	79,90	39,9%	61,00	30,5%	70,57	35,3%	58,37	29,2%	70,24	35,1%	66,72	33,4%	75,65	37,8%	58,66	29,3%	68,47	34,2%
	O4	-	-	-	-	63,76	31,9%	77,91	39,0%	59,79	29,9%	70,37	35,2%	-	-	-	-	66,07	33,0%	75,03	37,5%
	O5	-	-	-	-	59,20	29,6%	72,21	36,1%	58,33	29,2%	72,41	36,2%	-	-	-	-	59,39	29,7%	70,53	35,3%
	O6	-	-	-	-	-	-	-	-	63,45	31,7%	72,47	36,2%	-	-	-	-	-	-	-	-
	O7	-	-	-	-	-	-	-	-	57,59	28,8%	70,38	35,2%	-	-	-	-	-	-	-	-
500	O1	161,68	32,3%	171,69	34,3%	150,12	30,0%	166,31	33,3%	151,70	30,3%	162,64	32,5%	140,08	28,0%	156,77	31,4%	139,17	27,8%	157,10	31,4%
	O2	158,02	31,6%	165,23	33,0%	153,57	30,7%	169,60	33,9%	151,79	30,4%	168,51	33,7%	136,13	27,2%	154,36	30,9%	137,67	27,5%	157,92	31,6%
	O3	162,84	32,6%	169,59	33,9%	144,94	29,0%	160,67	32,1%	152,14	30,4%	165,98	33,2%	138,45	27,7%	159,24	31,8%	131,15	26,2%	153,84	30,8%
	O4	-	-	-	-	147,51	29,5%	164,88	33,0%	150,20	30,0%	163,02	32,6%	-	-	-	-	135,23	27,0%	152,98	30,6%
	O5	-	-	-	-	154,57	30,9%	172,45	34,5%	147,85	29,6%	163,62	32,7%	-	-	-	-	136,56	27,3%	162,26	32,5%
	O6	-	-	-	-	-	-	-	-	156,93	31,4%	172,08	34,4%	-	-	-	-	-	-	-	-
	O7	-	-	-	-	-	-	-	-	151,86	30,4%	170,82	34,2%	-	-	-	-	-	-	-	-
Minimum	∅ **27,4%**		27,9%		33,0%		23,9%		32,1%		15,3%		32,5%		23,4%		30,9%		23,9%		30,6%
Maximum	∅ **38,6%**		35,4%		40,9%		33,5%		44,7%		31,7%		45,2%		33,4%		41,8%		34,8%		44,7%