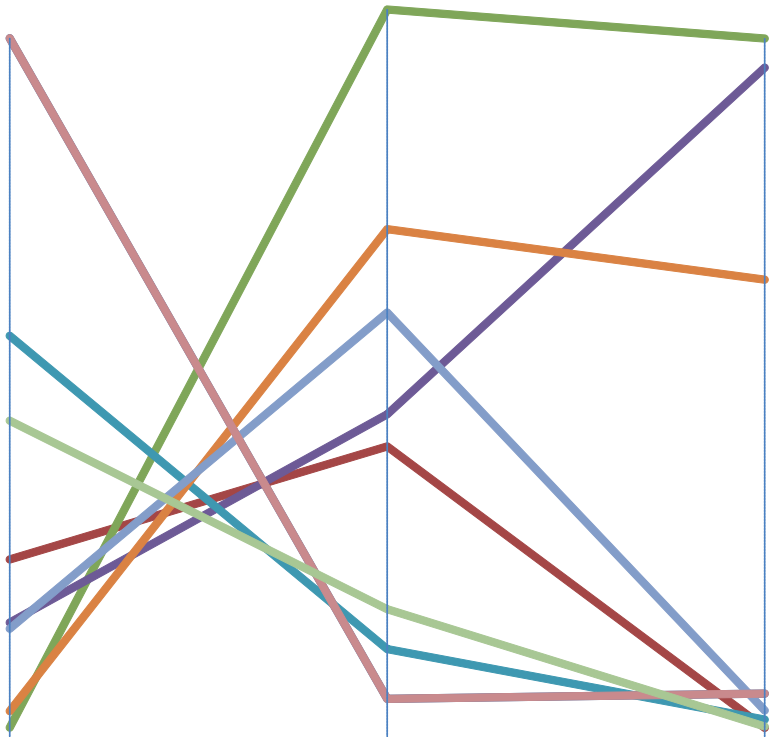
JYVÄSKYLÄ STUDIES IN COMPUTING
197

Vesa Ojalehto

On Solving Computationally Expensive Multiobjective Optimization Problems with Interactive Methods



JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 197

Vesa Ojalehto

On Solving Computationally Expensive Multiobjective Optimization Problems with Interactive Methods

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen auditoriossa 2
lokakuun 23. päivänä 2014 kello 14.30.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, auditorium 2, on October 23, 2014 at 14.30 o'clock.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2014

On Solving Computationally Expensive Multiobjective Optimization Problems with Interactive Methods

JYVÄSKYLÄ STUDIES IN COMPUTING 197

Vesa Ojalehto

On Solving Computationally Expensive Multiobjective Optimization Problems with Interactive Methods

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2014

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-5852-7
ISBN 978-951-39-5852-7 (PDF)

ISBN 978-951-39-5851-0 (nid.)
ISSN 1456-5390

Copyright © 2014, by University of Jyväskylä

Jyväskylä University Printing House, Jyväskylä 2014

ABSTRACT

Ojalehto, Vesa
On Solving Computationally Expensive Multiobjective Optimization Problems with Interactive Methods
Jyväskylä: University of Jyväskylä, 2014, 70 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 197)
ISBN 978-951-39-5851-0 (nid.)
ISBN 978-951-39-5852-7 (PDF)
Finnish summary
Diss.

In this thesis, we address the challenges encountered when solving multiobjective optimization problems with interactive methods. We concentrate on practical issues, that is, we discuss structures that can be utilized when implementing an interactive method and practicalities of connecting an interactive method with a multiobjective optimization problem formulated e.g. in different modelling environments. As most real-world multiobjective optimization problems are computationally expensive, we also study how to facilitate the use of interactive methods for such problems.

Even though interactive multiobjective optimization methods have been widely discussed in the literature, their implementations are rare. In this thesis, we define a core structure of interactive multiobjective methods, where we identify the main steps required for implementing an interactive method. As a concrete example of utilizing the core structure, we demonstrated the IND-NIMBUS software framework that includes implementations of several different interactive methods.

When dealing with a real-world optimization problem, the problem is often modeled with some simulator that must be connected to the optimization method implementation. In this thesis, we study how such a connection should be constructed, in order to enable changing the interactive method or the problem being solved with as little effort as possible. In addition, as the interactive method involves a decision maker, we propose an algorithm, where the decision maker can be involved in verifying the model of the multiobjective optimization problem in addition to solving it.

When solving a computationally expensive multiobjective optimization problem with an interactive method, the decision maker may experience unfeasible waiting times. We address this issue by introducing an agent assisted interactive algorithm, where we replace the computationally expensive multiobjective optimization problem with a computationally less-expensive surrogate problem. With the algorithm, the accuracy of the surrogate problem is maintained by identifying the areas of the Pareto frontier that the decision maker is interested in. The introduced algorithm is not specific to any particular method. As an example, we implement the algorithm for the interactive NIMBUS method and PAINT surrogate method.

Keywords: Computational cost, Pareto optimality, Interactive multiobjective optimization, NIMBUS method, software implementation

Author

Vesa Ojalehto
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisor

Professor Kaisa Miettinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Reviewers

Professor Dr. Karl-Heinz Küfer
Department of Optimization
Fraunhofer Institute for Industrial Mathematics ITWM
Germany

Professor Francisco Ruiz de la Rúa
Department of Applied Economics (Mathematics)
University of Malaga
Spain

Opponent

Professor Theodor Stewart
Department of Statistical Sciences
University of Cape Town
South Africa

Manchester Business School
The University of Manchester
United Kingdom

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Prof. Kaisa Miettinen without whose support and advise I would not have been able to finalize this thesis. I am grateful for the valuable feedback given by my thesis reviewers Prof. Francisco Ruiz and Prof. Karl-Heinz Küfer. I would also like to express my appreciation to the members of the industrial optimization group, both past and present, who have maintained an inspiring research environment and whose research topics have been the motivating force behind this thesis. This thesis could not be possible without ideas, and research questions arising from the different industrial projects that the group members have been involved in. Furthermore, I give thanks to Prof. Marko Mäkelä for giving me the first nudge to the path leading to this thesis.

Finally, I thank all members of my family, especially my partner Katja whose support and patience have been invaluable for finalizing this thesis as well as my daughter Helka, who has not yet failed to remind me of the life beyond research.

This study was financially supported by the Academy of Finland (grant number 128495), by the Ellen and Antti Nyyssönen Foundation, by the Jyväskylä Doctoral Program in Computing and Mathematical Sciences in Finland, by the Antti Wihuri Foundation and by the KAUTE foundation.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Timo Laukkanen · Tor-Martin Tveit · Vesa Ojalehto · Kaisa Miettinen · Carl-Johan Fogelholm. An Interactive Multi-objective Approach to Heat Exchanger Network Synthesis. *Computers & Chemical Engineering, 34(6)*, 2010.

PII Vesa Ojalehto · Kaisa Miettinen · Timo Laukkanen. Implementation Aspects of Interactive Multiobjective Optimization for Modeling Environments: The Case of GAMS-NIMBUS. *Computational Optimization and Applications, 58(3)*, 2014.

PIII Markus Hartikainen · Vesa Ojalehto · Kristian Sahlstedt. Applying Approximation Method PAINT and Interactive Method NIMBUS to Multiobjective Optimization of Operating a Wastewater Treatment Plant. *Engineering Optimization, to appear, DOI:10.1080/0305215X.2014.892593.*

PIV Karthik Sindhya · Vesa Ojalehto · Jouni Savolainen · Kaisa Miettinen · Hannu Niemistö. Coupling Dynamic Simulation and Interactive Multiobjective Optimization for Complex Problems: An APROS-NIMBUS Case Study. *Expert Systems with Applications, 41(5)*, 2014.

PV Vesa Ojalehto · Dmitry Podkopaev · Kaisa Miettinen. Agent-based Interactive Approach for Computationally Demanding Multiobjective Optimization Problems. *Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing, No. B 6/2014, University of Jyväskylä, Jyväskylä*, 2014.

1 INTRODUCTION

Many real-world optimization problems contain several conflicting objectives. Nevertheless, when optimizing, e.g., in the field of engineering, the problem is often modeled with a single objective. For example, objectives can be combined into a single objective as a weighted sum, or only one of the objectives may be selected to be optimized, while others are handled as constraints. Unfortunately, these straightforward conversions can lose information about interdependencies among the objectives that may affect the validity of the obtained results. Even if optimizing a single objective function results in an optimal solution, the solution might not be the best possible one for the original problem. A better solution could have been found if all trade-offs between conflicting objectives had been considered. Therefore, multiobjective optimization has been, and is, an active research area, which has produced a wide variety of different optimization methods (see e.g. [13, 20, 55] and references therein).

For an optimization problem with several conflicting objectives, it is usually not possible to find an unique optimal solution. Instead, a multiobjective optimization problem typically has several compromise solutions with different trade-offs. Therefore, some additional information is needed to select the best solution from the set of these so-called Pareto optimal solutions. This information is usually obtained from the preferences of a decision maker (DM) who is an expert in the problem domain. This means that the DM is in a very demanding position, as (s)he is responsible for providing the information that is used to decide which of the different solutions is the best compromise solution for the problem.

In other words, in comparison to a single objective optimization, the multiobjective optimization does not provide the DM with a single optimal solution. Instead, the DM is supported in exploring the different available alternative solutions in order to find the solution that best corresponds to his or her preferences. Especially when dealing with real-world applications, the overall aim of using a multiobjective optimization is usually to find a single, final solution for the optimization problem. It should be noted that the final solution is not necessarily directly provided by the optimization method, as the expert in the problem domain should always have the final say on the characteristics of the solution. The

DM or some other expert in the domain may choose to change some aspects of the solution generated with the optimization method before an actual, real-world product is produced.

When solving a multiobjective optimization problem, it may be difficult for the DM to provide preference information before getting to know the possibilities and limitations of the solutions attainable. On the other hand, generating and comparing many Pareto optimal solutions may be computationally expensive, i.e., time consuming, as well as exhaustive from the cognitive point of view. In the literature (see e.g. [55]), several distinct exist approaches for supporting the DM during the decision making process. In this thesis, we concentrate on interactive approaches, where the most preferred Pareto optimal solution is found by asking the DM to specify preference information in an iterative manner until the most preferred solution is found.

When using a method based on an interactive approach, i.e., *an interactive method*, the concerns related to cognitive load, and to computational expense are lessened as the solution process consists of consecutive iterations, each dealing with one or a small number of Pareto optimal solutions at a time. At each iteration, the DM provides preference information that is used to generate Pareto optimal solutions that best correspond to the DM's preferences. This means that the DM can concentrate on solutions that are interesting to him or her and that the cognitive load is distributed over several iterations. Furthermore, with the iterative process, the DM can learn about the interdependencies between the objectives while simultaneously learning about the feasibility of his or her preferences. In this way, the DM gains insight into the problem, which may even lead the DM to change his or her preferences. Based on the learning process originating from the interactive method, the DM can make informed decisions on which of the alternative solutions is the best compromise for the problem. For these reasons, in this thesis we concentrate on interactive methods.

Interactive methods have provided promising solutions in various fields including mechanical engineering [43, 58, 65], chemical engineering [6, 12, 60], health care [22, 68, 79, 80, 98], civil engineering [17, 69, 92], etc. Even though many interactive methods have been proposed in the literature during the years (see, e.g. [3, 35, 36, 47, 55, 67] and references therein), implementations of interactive methods are not commonly available. In [42] it is suggested that in order to increase the adaption of interactive methods, the technical aspects related to implementing an interactive method should be separated from the methodological issues. One aim of this thesis is to elaborate on this in order to enable the easier implementation of interactive methods by offering insights into the practical issues encountered when following this division.

In practice, when solving a multiobjective optimization problem, it is not enough to have access to an implementation of an optimization method. In addition, a realization of the optimization problem must exist, i.e., the problem model, and some kind of a connection must also exist between these two. The practicalities of the connection are often not considered at all, as the methods discussed in the literature are usually applied only for a single or, at most, for a few real-world

problems. As one of the aims of this thesis is to support the DM when solving multiobjective problems emerging from real-word applications, we consider the requirements of constructing such a connection. Furthermore, the problem model formulation is usually assumed to be correct when the optimization process is started, but based on our experiences, this is often not the case in practice. In light of this, we describe ways of utilizing the DM using an interactive method as an additional means to consider the correctness of the problem model.

Real-world multiobjective optimization problems are usually computationally expensive. The function evaluations may be complex, leading to computationally expensive computations or simulations (see e.g. [4, 36, 38, 93, 104]). The computational expense may even be so high that the use of an interactive method as described above may become infeasible due to the long waiting times needed to generate new solutions. The DM may be forced to restrict his or her search onto very few alternatives or, due to the time or other constraints, (s)he may stop the solution process prematurely. This may mean that the best compromise solution is not found or that the DM is not confident of the insight (s)he has gained during the process.

Historically, the amount of computational resources available for an individual engineer or a researcher has followed the so-called Moore's Law, which has predicted continuing increases to computing power as a result of doubling the amount of components in a circuit every 18 months. Over several decades, this has led to a significant increase in computational resources to be gained per monetary costing unit (see e.g. [48]). As computational resources have increased, it has been possible to solve existing optimization problems faster and faster. It is quite possible that an optimization problem which was considered to be computationally expensive when developed, requiring the use of optimization methods utilizing information regarding the problem characteristics when originally solved [65], can later be considered to be computationally inexpensive, and can be solved as a black-box problem, i.e., without using any additional information [58]. Naturally, this means that the issues related to interactive methods when dealing with a computationally expensive problem are overcome for those problems.

However, as the available computational resources have increased, it has become possible to solve problems with more details so that the real-world applications can be reflected more accurately. Therefore, the complexity, and in effect the computational expense of the problems, has also increased. This means that one cannot assume that increases in the available computational resources would remove the need for developing more efficient optimization methods. Instead, other approaches should be considered.

Naturally, when dealing with computationally expensive optimization problems, one should use methods that produce good solutions using as few function evaluations as possible. As mentioned earlier, one approach for solving computationally expensive problems is to utilize problem specific information, such as information on the derivatives of the objective functions. However, the real-world problems are often based on complex calculations, where it is hard, or even im-

possible to obtain any additional information besides the values of objective (and possibly constraint) functions and decision variables. Additional information can be obtained for these so-called black-box problems by using, e.g., numerical differentiation (see e.g. [21]) or automatic differentiation (see e.g. [31]). Nevertheless, these methods have their own trade-offs. For example, obtaining derivative information with numerical differentiation has issues in terms of the accuracy of the derivatives and increases the number of function evaluations. On the other hand, use of automatic differentiation requires access to the internal details of the problem model, which is usually not available when dealing with a black-box problem.

Another widely used approach is to execute function evaluations in a parallel fashion by distributing them to different processors or computers in order to decrease the total time needed for finding optimal solution(s) (see e.g. [5, 95]). Again, especially when dealing with black-box problems, the problem formulation may require resources that cannot be distributed. For example, the simulator needed for the function evaluations may only have a limited number of licenses available. It should be noted that even when the problem is developed "in house," i.e., when all implementation details are available and can be changed, the distribution of the function evaluations might prove to be burdensome if parallelism has not been considered at the beginning of the problem model formulation.

Finally, one commonly used methodology for handling computationally expensive black-box optimization problems is the use of a surrogate problem, where the computationally expensive problem is replaced with a computationally less expensive surrogate problem (see e.g. [2, 27, 74, 83, 106]). The construction of the surrogate problem may be computationally expensive, as it usually involves generating several Pareto optimal solutions. However, as the DM is involved only when exploring the Pareto frontier, the time-consuming calculations can be done without too large an effect on the solution process (see e.g. [98]). It should be noted that when using a surrogate problem with an interactive method, the Pareto optimal solutions shown to the DM are not solutions for the actual problem. As the DM gains insight into the characteristics of the problem based on the shown solutions, it is important that the solutions give accurate information. If the approximated solutions are not accurate, i.e., when they do not correspond to the Pareto optimal solutions for the original problem, the DM may develop a misleading view of the problem that can lead to wasted effort on the part of the DM. It is even possible that the final solution selected by the DM will not correspond to his or her preferences.

The motivation for this thesis originates from the experiences obtained from the author's participation in finding solutions for different optimization problems emerging from real-world applications. These have included a wide spread of different types of optimization problems, including the optimal design of the structure of a grapple loader [7], decision support for a housing mobility program [41], problems in chemical process design [32], the separation of glucose and fructose [33], the optimal control of a continuous casting of steel [57, 58], in-

tensity modulated radiotherapy treatment planning [79], brachytherapy for cancer treatment [81], the optimization of a heat exchanger network [PI, PII, 44], optimizing configurations of an oxyfuel power plant process [100], optimizing the concentration-dampening process in a tank [89], the optimization of an ultrasonic transducer [39], paper making [50], operating a wastewater treatment plant [PIII, 35, 36], the optimization of an internal combustion engine [4], optimal design and control of a paper mill [93], the design of a permanent magnet direct-driven wind generator [29], and the optimization of a two-stage separation process [PIV, PV], among others.

Finding solutions for these and other problems has involved aspects from the problem formulation to the technical details of supporting different simulator and modeling software. In this thesis, we restrict our examination to the issues related to implementing interactive optimization methods as well as to preparing a problem model for optimization. To provide the practical results of our research, we reflect on these issues by supporting several DMs in solving multiobjective optimization problems for heat-exchanger network synthesis [PI], operating a wastewater treatment plant [PIII], and two-stage separation process [PIV, PV]. As all three optimization problems, or their extended formulations, can be considered computationally expensive, we also discuss an approach for utilizing an interactive method for computationally expensive problems. All of our research results have been implemented in the IND-NIMBUS software framework [56] (http://ind-nimbus.it.jyu.fi/) for interactive multiobjective optimization. As an interactive method for solving these problems, we have used the interactive NIMBUS method [55, 62, 64]. The IND-NIMBUS framework contains implementations of all of the methods discussed in this thesis, as well as the tools for accessing the problems being solved.

This thesis consists of four articles and a manuscript. In article [PI], we describe an implementation of the interactive NIMBUS method within the GAMS modeling environment [14]. Based on the experiences gained, in [PII] we discuss how an interactive method can be regarded as consisting of distinct components. We describe these components as a core structure that can be used to facilitate the implementation of an interactive method. We demonstrate the core structure by re-implementing the interactive NIMBUS method within GAMS. These results are then used to support the DM in solving a multiobjective heat exchanger network synthesis problem [105] modeled with GAMS.

We introduce the IND-NIMBUS PAINT module in [PIII] for solving computationally expensive problems with interactive methods. The module can be used to construct a PAINT surrogate problem [37] that can be solved with the NIMBUS method. The IND-NIMBUS PAINT module is demonstrated by supporting the DM in solving the problem of operating a wastewater treatment plant.

In article [PIV], we describe a general interface for coupling an interactive method with simulator software. The interface is then used to connect the NIMBUS method to the dynamic process simulator APROS [88] and the connection is used to solve a two-stage separator process problem. In [PIV] we also discuss how the DM can assist in revisiting of a problem model by introducing an aug-

mented interactive multiobjective optimization (AIMO) algorithm.

As mentioned, using a surrogate problem may lead to a wasted effort on the part of the DM if the surrogate is not accurate enough. Therefore, in article [PV] we suggest maintaining the accuracy of a surrogate problem by updating it based on the preference information that the DM provides during the interactive solution process. To this end, we introduce an algorithm that utilizes independent and intelligent agents to observe the interactive solution process, and, based on the observations, to update the surrogate problem for the areas of interest. The agent assisted algorithm can also provide the DM with additional Pareto optimal solutions in the areas that the DM has shown interest in. We demonstrate the agent assisted algorithm by solving the two-stage separator problem introduced in [PIV].

The contents of the rest of this thesis are as follows. First, in Chapter 2 we provide the basic concepts related to the multiobjective optimization and briefly describe the methods that we have used in this research. In Chapter 3, we first define a computationally expensive problem in the context of this thesis and then continue with the author's contribution to this thesis.

In Chapter 3, we consider the experiences gained from solving different multiobjective optimization problems and suggest some practices for applying interactive methods in real-world applications. Chapter 4 is devoted to presenting the observations and hints on implementation issues of the interactive methods gathered during the research related to this thesis. In Chapter 5, we describe the new agent assisted algorithm developed for assisting the DM in solving computationally expensive multiobjective optimization problems with interactive methods. The author's contribution to the four articles and to the manuscript included in this thesis is described in Chapter 6. Finally, in Chapter 7, we provide some concluding remarks and discuss future research topics.

2 BACKGROUND MATERIAL

In this chapter, we discuss the background material used in this thesis. Besides providing the formulation and the main concepts of the multiobjective problem considered, we briefly give details on interactive methods in general, concentrating on the interactive NIMBUS method that has been the main method used during the preparation of this thesis. Finally, we give a short overview of multiagent systems that are utilized by the agent assisted algorithm introduced in this thesis.

In this thesis, we also refer to the PAINT method [37] and to the Pareto Navigator method [27] developed for computationally expensive problems. The Pareto Navigator method is suitable only for convex problems, and as the problem models discussed in this thesis are nonconvex, here we use the PAINT method. When using either of these methods, the original, computationally demanding optimization problem is replaced with a computationally less demanding *surrogate problem*. The surrogate problem is constructed in such a way that it approximates the original problem, while allowing new solution candidates to be produced faster than when using the original.

When using the PAINT method, the surrogate problem is constructed by interpolation of a pre-computed set of Pareto optimal solutions. Here this set is referred to as a *constructing set*. The constructing set can be generated with any multiobjective optimization method that generates many Pareto optimal solutions (see e.g. [85, 55, 13]). After constructing the surrogate problem, it can be used to replace the original problem. The Pareto optimal solutions obtained when using the surrogate problem are called *approximate Pareto optimal solutions*. More details of the PAINT method can be found in [37].

2.1 Some Concept of Multiobjective Optimization

In this thesis, we consider multiobjective optimization problems of the form

$$\begin{array}{ll} \text{minimize} & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{subject to} & \mathbf{x} \in S, \end{array} \tag{1}$$

where $f_i : S \to R$ are k (≥ 2) conflicting objective functions, and $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the *decision variable* vector bounded by constraints that form the feasible set $S \subset \mathbb{R}^n$. Objective vectors $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$ consist of *objective function* values calculated at $\mathbf{x}$. If some objective function is to be maximized, the objective function values can be multiplied by -1 to get a function to be minimized.

A decision vector $\hat{\mathbf{x}}$ and the corresponding objective vector $\mathbf{f}(\hat{\mathbf{x}})$ are called a Pareto optimal if there does not exist any other feasible $\mathbf{x}$ so that $f_j(\mathbf{x}) \leq f_i(\hat{\mathbf{x}})$ for all $i = 1, ..., k$ and $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$ for least one $j = 1, ..., k$. The objective vector $\mathbf{f}(\hat{\mathbf{x}})$ is called *Pareto optimal solution* to problem (1), and a set of Pareto optimal solutions is called a Pareto optimal set [55] or *Pareto frontier*.

For solving multiobjective objective optimization problems there exist several different *optimization methods*. A *solution process* is the overall process to find a solution for the problem (1). Depending on the method used during the solution process, the solution to a multiobjective optimization problem can be a single Pareto optimal solution or a set of Pareto optimal solutions. In practice, only a single solution is desired to be implemented and in this thesis such a solution is referred to as the *final solution*.

During a solution process, it is often useful to obtain information on the possible ranges of the objective function among the Pareto optimal solutions. An *ideal (objective) vector* $\mathbf{z}^\star \in \mathbf{R}^k$ is the vector representing the best objective function values, and a *nadir (objective) vector* $\mathbf{z}^{\text{nad}} \in \mathbf{R}^k$ represents the worst values. Due to issues related to method implementation, an *utopian (objective) vector* $\mathbf{z}^{\star\star}$ is sometimes needed. The utopian vector is a vector whose components are strictly better than those of $\mathbf{z}^\star$.

These frequently used concepts of multiobjective optimization are illustrated in Figure 1. In the figure, there are two objective functions, f_1 and f_2, and the feasible set S. The ideal objective vector is shown as the point $\mathbf{z}^\star$ and nadir objective vector as the point $\mathbf{z}^{\text{nad}}$. The Pareto frontier is marked with a bold line.

The components of the ideal objective vector can be determined by optimizing each objective function f_i individually subject to the feasible set S. For multiobjective optimization problems with more than two objectives, obtaining a nadir objective vector is a more complex task. One approach of obtaining an estimate of the nadir objective vector values is to use a pay-off table [10, 55] which can be formed after determining the ideal objective vector. Obtaining more accurate values for the nadir objective vector is still an ongoing topic, see e.g. [25].

In order to an optimization problem to be solved, there must exist some mathematical description of it. That is, some model that represents the different

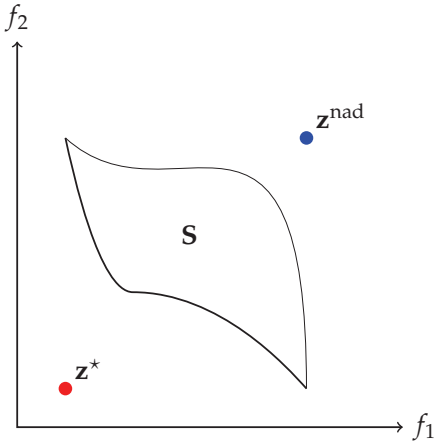


FIGURE 1 Illustration of Pareto frontier, ideal and nadir objective vectors with $k = 2$

mathematical relations which describe functions of some physical phenomenon or system. To solve this mathematical model with a computer, it must be implemented as a computer program. In this thesis, a *problem model* refers to such a program and the purpose of a problem model is to study a phenomenon or system without need for physical example of it. The problem models considered in this thesis are implemented either with a simulator software or with a modeling language. Here, a *simulator* is a computer program that can be used to imitate physical phenomena or systems. A *modeling language* is a definition of how mathematical expressions describing a physical phenomena or system so that it can be interpreted with a *modeling environment*. In addition to interpreting problem models, the modeling environment contains an optimization method to solve the models. For further details on simulation and modeling environments see e.g. [28, 73, 91].

As there exist multiple Pareto optimal solutions for any problem (1) with conflicting objectives, some additional information is needed in order to select a single or a set of most preferred Pareto optimal solutions. This is usually done with the help of a *decision maker* (DM), who can provide information on his or her preferences in relation to the problem being solved. In addition to the DM, the solution process may involve an *analyst*. The analyst is a person who is familiar with the method used during the solution process and helps the DM during its use. In addition, the analyst may formulate the problem model, or the model may by modeled by a *modeler*.

The role of the DM is so important that multiobjective optimization methods are often classified according to the role of the DM (see, e.g., [55]) to four distinct classes. When a multiobjective optimization problem is solved by a *no preference* method, the DM's preferences are not taken into account. If preference information is provided by the DM before the solution process, the method is referred to as an *a priori* method. When using an *a posteriori* method, the DM is provided with a representative set of Pareto optimal solutions, from which (s)he selects the

solution (s)he prefers the most. Lastly, with an *interactive* method the DM participates in an iterative process, where (s)he is shown Pareto optimal solution(s) and asked to provide preference information in relation to those solutions. Then the DM is provided with new Pareto optimal solution(s) which should correspond to his or her preferences.

With the interactive solution process, the DM can study the relationships between the objective functions by obtaining an understanding on how changes in some objective functions affect the others. By specifying the preference information, the DM can guide the search process to the areas of interest, allowing the DM to concentrate on a small set of Pareto optimal solutions. By learning the behavior of the problem, the DM can adjust his or her preferences to reflect the gained understanding.

One of the commonly used ways of solving a multiobjective optimization problem is to formulate a single objective optimization problem that uses preference information specified by the DM to generate a Pareto optimal solution. Such a single objective optimization problem is referred here as a *scalarized subproblem* and the objective function of the subproblem is called a *scalarizing function*. In the next section we will give short description of scalarized subproblems.

2.2 Scalarization of the Multiobjective Optimization Problem

Interactive methods considered in this thesis are based on the scalarization of the multiobjective optimization problem. As defined in the previous section, the scalarization means that the problem is converted to a single objective optimization problem. This is achieved by constructing a single objective scalarizing function and, in some cases, by adding new constraint functions to the subproblem. This subproblem can then be solved by an appropriate single objective optimization method.

The scalarizing function should be constructed in such a way that it can be used to generate any Pareto optimal solution and all solutions generated are ensured to be Pareto optimal [55]. In addition, the scalarized subproblem typically includes some preference information in order to generate satisfactory solutions for the DM. As there exist several different ways of constructing the scalarizing function, it is possible to generate different Pareto optimal solutions using the same preference information (see e.g. [16, 63]).

In addition to using different scalarizing functions, the scalarized subproblems can differ on what kind of preference information is used and how the preference information is specified by the DM. For example, the DM may be asked to specify his or her preferences by providing aspiration levels, i.e., values of objective functions that (s)he would like to obtain, or (s)he could be asked to do pairwise comparisons between Pareto optimal solutions in order to indicate which solutions (s)he prefers.

2.3 NIMBUS Method

As mentioned, in this thesis we utilize the interactive NIMBUS method [55, 64] as the interactive method for solving multiobjective optimization problems. The implementations of the NIMBUS method used here are realized in the IND-NIMBUS software framework [58], along with the other methods researched in this thesis. In addition IND-NIMBUS, there exist a WWW-NIMBUS optimization system, that has been available for free on the Internet for academic teaching and research purposes since 1995 (wwwnimbus.it.jyu.fi) [62]. The WWW-NIMBUS system is not considered in this thesis.

The NIMBUS method has been used for solving a wide array of multiobjective optimization problems, including chemical process design [32], optimal shape design of ultrasonic transducers [39], separation of glucose and fructose [33], continuous casting of steel [57, 58, 65] intensity modulated radiotherapy treatment planning [79], brachytherapy for cancer treatment [81], optimization of heat exchanger network [44], optimizing configurations of an oxyfuel power plant process [100], wastewater treatment plant simulation and optimization [35, 36], dynamic optimization of a two-stage separation process [PIV] and for concentration control problem [89].

Next we describe the algorithm of the interactive NIMBUS method and the scalarized subproblems used in the NIMBUS method to generate Pareto optimal solutions. Originally (see [55, 61]), the NIMBUS method utilized only a single scalarized subproblem, but in the so-called synchronous NIMBUS method [64] three additional subproblems were introduced. Therefore, when using the NIMBUS the DM is provided up to four different Pareto optimal solution for each preference information (s)he specifies.

2.3.1 NIMBUS Algorithm

The central idea of the NIMBUS method is that the DM is provided with the objective function values of the current Pareto optimal decision vector. The DM is then asked to specify his or her preferences by classifying the objective functions into up to five classes. This means that the DM is asked to indicate how the objective function values should be changed, in order to obtain a more satisfactory solution. The classes are functions f_i whose values

- should be improved ($i \in I^{<}$),
- should be improved to some aspiration level $\hat{z}_i < f_i(\mathbf{x^c})$ ($i \in I^{\leq}$),
- are satisfactory at the moment ($i \in I^{=}$),
- are allowed to impair up till some bound $\varepsilon_i > f_i(\mathbf{x^c})$ ($i \in I^{\geq}$),
- are allowed to change freely ($i \in I^{\diamond}$).

As solutions presented to the DM are Pareto optimal, there does not exist any Pareto optimal solutions whose all objective function values could be better than those of the solution shown to the DM. Therefore, the classification must be done so that at least one objective function value should be improved, and at least one is allowed to impair.

Based on the classification information specified by the DM, up to four different scalarized subproblems are constructed. As solutions to these subproblems we obtain new Pareto optimal solutions to be presented to the DM. It should be noted that it is possible that different subproblems do not all produce different Pareto optimal solutions for the same classification information [63].

The NIMBUS method is described in Algorithm 1.

Algorithm 1 NIMBUS method algorithm

1: Calculate ranges of the objective functions and generate an initial Pareto optimal solution and set it as the current solution.
2: Ask the DM to specify the classification information at the current solution.
3: Ask the decision maker to select the number (one to four) of new Pareto optimal solutions to be generated.
4: Generate new Pareto optimal solutions.
5: Present the new Pareto optimal solutions to the DM.
6: Ask the DM to choose the most preferred solution among the set of previously generated the Pareto optimal solutions or the current solution.
7: If the DM wants to continue, go to step 2. Otherwise, stop.

In addition to steps presented in Algorithm 1, the DM can decide to generate intermediate solutions between any two Pareto optimal solutions at any point of the algorithm. This is done by generating as many evenly spaced objective vectors as the DM desires between the two selected Pareto optimal solutions. The resulting objective vectors are projected to the Pareto frontier and then presented to the DM.

The initial Pareto optimal solution can either be given by the DM to be projected to the Pareto frontier, or it can be some neutral compromise solution between the nadir and ideal vectors [64]. The solution process is continued iteratively until the DM does not want to improve any objective function value, is not willing to let any objective value impair or does not wish to generate additional intermediate solutions.

By following the algorithm, the DM can move along the Pareto frontier, in effect searching for the most satisfactory solutions available for the problem. During this search, the DM can learn about the problem and adjust one's preferences and finally to find a Pareto optimal solution that (s)he can be confident of being the most satisfactory solution for the problem. For more details on the NIMBUS method, see [64].

2.3.2 Scalarized subproblems of the NIMBUS method

As mentioned, there are up to four different single objective scalarized subproblems used in the NIMBUS method. In the original NIMBUS method [62], there was a single subproblem, a so-called standard subproblem. The standard NIMBUS subproblem is of the form

$$\begin{array}{ll} \text{minimize} & \max\limits_{\substack{i\in I^{<}\\ j\in I^{\leq}}} \left[\dfrac{f_i(\mathbf{x}) - z_i^{\star}}{z_i^{\text{nad}} - z_i^{\star\star}}, \dfrac{f_j(\mathbf{x}) - \hat{z}_j}{z_j^{\text{nad}} - z_j^{\star\star}} \right] + \rho \sum\limits_{i=1}^{k} \dfrac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} & f_i(\mathbf{x}) \leq f_i(\mathbf{x}^c) \ \text{ for all } \ i \in I^{<} \cup I^{\leq} \cup I^{=}, \\ & f_i(\mathbf{x}) \leq \varepsilon_i \ \text{ forall } \ i \in I^{\geq}, \\ & \mathbf{x} \in S. \end{array} \tag{2}$$

where $\mathbf{x}^c$ represents the decision variable vector of the current solution. The latter part of the objective function is an *augmentation term* that guarantees that the generated solutions are Pareto optimal (see [55, 63]) and $\rho > 0$ is a so-called augmentation coefficient. The aspiration levels and bounds are given as $\hat{z}_i$ and ε_i, respectively. Here (and in the following formulations) we assume that all object functions are to be minimized.

As different subproblem formulations typically generate different Pareto optimal solutions [63], the current, so-called synchronous NIMBUS method [64] has three additional, reference point based subproblems. These subproblems assume that instead of a classification information, the DM has specified his or her preferences as a reference point, marked here as $\bar{z}_i \in \mathbf{R}^k$. If we know the ranges of the objective functions in the Pareto frontier, the classification information can be easily convert to a reference point. To do the conversion, it is possible to set $\bar{z}_i = z_i^{\star}$ for $i \in I^{<}$, $\bar{z}_i = \hat{z}_i$ for $i \in I^{\leq}$, $\bar{z}_i = f_i(\mathbf{x}^c)$ for $i \in I^{=}$, $\bar{z}_i = \varepsilon_i$ for $i \in I^{\geq}$ and $\bar{z}_i = z_i^{\text{nad}}$ for $i \in I^{\diamond}$.

Next we will present the three reference point based subproblems used in the NIMBUS method. Of these, the first subproblem uses the achievement scalarization function [102]

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,\ldots,k} \left[\dfrac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\text{nad}} - z_i^{\star\star}} \right] + \rho \sum\limits_{i=1}^{k} \dfrac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} & \mathbf{x} \in S. \end{array} \tag{3}$$

It should be noted that an objective vector can be projected to the Pareto frontier by setting it as a reference point to problem (3). For example, in the NIMBUS method, problem (3) is used to generate the intial Pareto optimal solution and intermediate solutions.

The scalarization function of the satisficing trade-off method (STOM) [70] is used in the second reference point based subproblem

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,\ldots,k} \left[\dfrac{f_i(\mathbf{x}) - z_i^{\star\star}}{\bar{z}_i - z_i^{\star\star}} \right] + \rho \sum\limits_{i=1}^{k} \dfrac{f_i(\mathbf{x})}{\bar{z}_i - z_i^{\star\star}} \\ \text{subject to} & \mathbf{x} \in S. \end{array} \tag{4}$$

And finally, the scalarization function of the last subproblem is based on the GUESS method [15]

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,\ldots,k} \left[\dfrac{f_i(\mathbf{x}) - z_i^{\text{nad}}}{z_i^{\text{nad}} - \bar{z}_i} \right] + \rho \sum\limits_{i=1}^{k} \dfrac{f_i(\mathbf{x})}{z_i^{\text{nad}} - \bar{z}_i} \\ \text{subject to} & \mathbf{x} \in S. \end{array} \tag{5}$$

The subproblems presented above are nondifferentiable as they involve single objective min-max functions. The multiobjective optimization problems considered in articles [PI, PII] are differentiable, and therefore the original nondifferentiable subproblems were modified to their differentiable equivalents. This was done by adding a new decision variable α to the problem, and treating the min-max functions as constraints. As an example, the differentiable version of subproblem (2) is

$$\begin{array}{ll} \text{minimize} & \alpha + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} & \dfrac{f_i(\mathbf{x}) - z_i^{\star}}{z_i^{\text{nad}} - z_i^{\star\star}} \leq \alpha \text{ for all } \; i \in I^{<}, \\ & \dfrac{f_j(\mathbf{x}) - \hat{z}_j}{z_j^{\text{nad}} - z_j^{\star\star}} \leq \alpha \text{ for all } \; j \in I^{\leq}, \\ & f_i(\mathbf{x}) \leq f_i(\mathbf{x}^c) \;\text{ for all } \; i \in I^{<} \cup I^{\leq} \cup I^{=}, \\ & f_i(\mathbf{x}) \leq \varepsilon_i \;\text{ for all } \; i \in I^{\geq}, \\ & \mathbf{x} \in S. \end{array} \tag{6}$$

The differentiable formulations of the three other subproblems used in the NIMBUS method are straightforward, and therefore they are not presented here.

2.4 Multiagent Systems

Next, we will give a short overview of the agents and multiagent systems used in this thesis. For the past few decades, agent-based computational intelligence approaches have been widely applied in several areas of computational sciences [71]. There does not exist any clear, universally agreed definition of an agent, but to put briefly, an *agent* is an autonomous computational unit that can observe its environment and take actions to fulfill its purpose in reaction to the observations. Agents are usually combined to form a *multiagent system*, where individual agents can communicate between themselves in order to share information on the environment and results of their actions.

Previously, agent-based technologies have been used for single (see e.g. [8, 46, 84, 86] and multiobjective (see e.g. [9, 23, 26, 45, 87, 90] optimization. In the previous research on agent-based multiobjective optimization, the focus has been on enhancing existing a posteriori methods or on studying a new agent-based a priori methods. To our knowledge, research conducted in [PV] is the first agent-

based approach to a combine multiagent system with an interactive method to support the DM in solving multiobjective optimization problems.

For the needs of our research, we have defined an agent to have the following six properties:

Emergent : agents solve problems with a set of simple rules.
Autonomous : agents have control of their inner state and they can take actions without human intervention.
Reactive : agents take actions based on their environment.
Goal-oriented : agents aim at achieving some goal with their actions.
Communal : agents are able to communicate with other agents, be they human or artificial.
Fault tolerant : agents can attempt to recover from a failure, e.g., a failure in reaching their goal.

With relatively simple agents fulfilling these properties, it is possible to construct complex multiagent systems that can be utilized for solving complex problems. Even tough the efficiency of utilizing multiple agents has been widely discussed in the literature (see e.g. [24, 51]), the research has concentrated on empirical studies, where the performance of agent-based optimization technologies is compared with more traditional methods. Nevertheless, it has been noted that by combining different methods, for example as a multiagent system, it is possible to ensemble structures where weaknesses on some parts are compensated by strengths in others [96]. Furthermore, in [99] it has been shown that the use of agent-based optimization should not decrease the efficiency of the underlying optimization method.

As multiagent-based systems have been widely used since 1990's, there exist several software frameworks for implementing multiagent systems and handling the communication issues between the agents. For this research, we have used the SPADE framework [30], following the FIPA communication model [71]. For further information on agents and multiagent systems, see e.g. [82, 103].

3 FINDINGS ON SOLVING MULTIOBJECTIVE OPTIMIZATION PROBLEMS WITH INTERACTIVE METHODS

As mentioned earlier, the main research questions of this thesis have originated from the direct need to solve multiobjective optimization problems emerging from several different fields. These problems have usually been complex, resulting in computationally expensive problem models. In addition, the complexity of the problems has often raised issues concerning the problem model's validity. In this chapter, we first give a definition of a computationally expensive problem model in the context of this thesis, i.e., when solving a problem with an interactive method. We then start by describing the author's contribution in this thesis by giving some insight into issues that can be encountered when solving such problems. We continue with the author's contribution by discussing how the involvement of the DM can be taken into account for revising the problem model when using interactive methods. Finally, as a problem model must be connected to an implementation of a method before it can be solved, we discuss how such a connection can be achieved.

3.1 On Computationally Expensive Problems

As mentioned earlier, the multiobjective optimization problems that have motivated this thesis have commonly been what we consider as computationally expensive. In the context of this thesis, a multiobjective optimization problem is considered to be computational expensive, if providing a solution for the problem is time consuming. For the problems that we have dealt with, the demands on time have been due to the time needed for performing complex calculations to evaluate the objective (and constraint) function values (see e.g. [36, 93]).

In our experience, when solving a computationally expensive problem with an interactive method, the time consumed may prove to be so high that the DM

is unable or unwilling to finish the solution process. It is also possible that the analyst involved in the solution process has only limited time available. Therefore, the process can be stopped even if some of the participants feel that there may be areas of the problem that have not yet been thoroughly explored. If so, the prematurely selected final solution may not be the most preferred solution, or the DM may not be confident that his or her preferences have been taken fully into account.

In this thesis, we approach solving a computationally expensive problem with an interactive method by replacing the problem with a computationally less expensive surrogate problem. When the DM has found the set of solutions for the surrogate problem that best corresponds to his or her preferences, these solutions can be projected to the Pareto frontier of the original problem and shown to the DM. In [PIII] we replace the computationally expensive wastewater treatment problem with a surrogate problem constructed with the PAINT method [37]. However, as discussed in [PV], such a surrogate problem may be inaccurate in the areas that the DM is interested in, and the solution process may lead to misleading results. Furthermore, it is possible that the surrogate problem may be accurate for some areas of the problem, but not for others. Therefore, the accuracy all of the solutions to the surrogate problem must be checked. Overall, when using a surrogate problem, the DM should be given an indication of how accurate the found solutions are. If the accuracy information is not provided by the method used for constructing the surrogate problem, the DM can be shown the differences between the solutions for the surrogate problem and the solutions obtained by projecting them onto the Pareto frontier of the original problem in relation to the ideal and nadir vector component values of the problem so that (s)he can gain some idea on as to how accurate the solutions to the surrogate problem might be.

The total time consumed to obtain the Pareto optimal solutions with an interactive method depends ultimately on the problem being solved. Nevertheless, the DM is the final arbitrator on deciding if the solution process is computationally expensive, i.e., if it is taking a "long" time, or not. Therefore, it is difficult to define a general distinction between a computationally expensive and an inexpensive problem, and the DM should be given the choice regarding "if and when" to use a surrogate problem. The DM should be given information on the accuracy of the surrogate problem, as if the surrogate problem is unacceptably inaccurate, then the whole solution process may prove to be a waste of effort on the part of the DM.

In general, if the inaccuracies cannot be adequately addressed, we suggest that the use of the surrogate problem should not be the default approach when solving computationally expensive problems with interactive methods. However, if the solution process cannot be made more efficient, i.e., faster, with other approaches such as parallelization, the use of a surrogate problem may be the only feasible option. In Chapter 5, we propose dealing with the accuracy of the surrogate problem by updating the accuracy in the areas that the DM is interested in.

In any case, the method implementation should be constructed so that the DM has an option to pause the solution process at any point (s)he so wishes. This means that the implementation should take note of the preference information provided by the DM and the Pareto optimal solutions obtained, and should allow the DM to study them, even after some period of time.

3.2 Augmented Interactive Multiobjective Optimization Algorithm

Even though in this thesis we do not consider how an optimization problem is modeled, the problem model is an integral part of our research. When exploring the problem model with an interactive method, the DM usually notices if any discrepancies exist in the trade-offs between conflicting objectives. In this section, we will consider how this aspect of interactive methods can be utilized during the solution process.

Based on the experiences of the author, and as discussed in [PIV], it is surprisingly common that the model for the optimization problem to be solved is not complete when an optimization method is initially used for finding the optimal solution(s). Even though the model validation is an important part of the process, it is usually not discussed in the literature. Instead, it is assumed that the model representing the optimization problem is formulated correctly without further comments. On the other hand, the possible inconsistent and erratic behaviors arising from the incomplete model can usually be easily observed when a problem is solved with an interactive method. When exploring the Pareto frontier, the DM can easily see if the interdependencies between the objective functions or their values do not correspond to his or her expectations. Similarly, the analyst can notice issues with the sensitivity of the problem model during the solution process.

Ideally, the correctness of the problem model should be verified before starting the solution process. In practice, this is often not the case, due to the complexity of the model, the time constraints imposed on the problem modeler, etc. In [PIV] we have presented AIMO algorithm, where we take advantage of the involvement of the DM in the interactive solution process by giving him or her an opportunity to revise the problem model. In Figure 2, we present a slightly simplified version of the algorithm.

A more detailed description of the AIMO algorithm is described in Algorithm 2. In Algorithm 2, new solutions are generated with an interactive method as described in Section 2, but the DM is given an option to provide information on the correctness of the problem model. If (s)he feels that the problem model is not correct, the modeler should revise the problem model, taking into account the comments made by the DM. Naturally, after each formulation, the modeler should also verify the correctness of the model.

In practice, the AIMO algorithm has been utilized when solving several different multiobjective optimization problems, including the problems in [PIII, PIV,

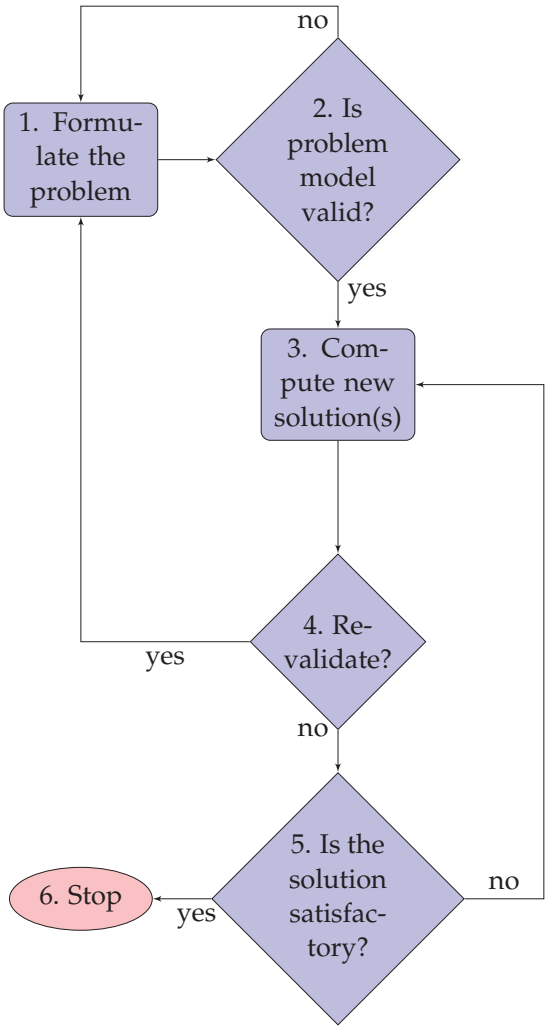


FIGURE 2 Flowchart of the augmented interactive multiobjective optimization algorithm

Algorithm 2 Augmented interactive multiobjective optimization algorithm

1: Formulate the problem model.
2: Verify the correctness of the problem model. If verification fails, go to step 1, otherwise continue.
3: Compute the initial solution to be shown to the DM.
4: Ask the DM to specify the preference information.
5: Compute new solution(s) to be shown to the DM.
6: If the DM wishes to revise the problem model, go to step 1, otherwise continue.
7: If the DM finds his or her preferred solution, continue, otherwise go to step 4.
8: The solution process is stopped.

PV], although it was described for the first time in [PIV]. For example, as reported in [PIV], an example of successful validation of the problem model by applying the AIMO algorithm can be found in [89], but the AIMO algorithm is not mentioned there. In [89], mistakes in the problem model formulation were detected immediately when the AIMO algorithm was applied, yet regardless of that, the problem model was previously solved with several different a posteriori methods. When using the posteriori method, the set of Pareto optimal solutions was shown to the DM, but he was not provided with any support in choosing the most preferred solution from the set. When using the interactive method, the DM was forced to consider the trade-offs when providing his preferences in step 4 of the AIMO algorithm and when the obtained Pareto optimal solutions did not follow his expectations, the question of problem model validity was brought up. The author has had similar experiences with other studies, but they have not been reported beyond [PIV, 89] and this thesis.

To summarize, we propose to use the AIMO algorithm to take advantage of the DM's expertise regarding the problem being solved. As the DM explores the trade-offs between objective functions, (s)he can detect any discrepancies and suggest that the problem model might be incorrectly formulated.

3.3 On Connecting the Interactive Method with the Problem Model

In this thesis we assume that a problem model exists that can be utilized in the solution process. Naturally, when solving an optimization problem it is not enough to have a valid problem model as the model must somehow be connected to an implementation of an optimization method. Next, we will discuss some approaches for building such a connection between a multiobjective optimization problem model and an interactive method.

In [PII], we give a short overview of how an existing model of an optimization problem can be prepared to be solved with an interactive method in a modeling environment (such as GAMS [14], AMPL [28] and AIMMS [11]). In [PIV], we consider how to build a more general interface to enable communication between an implementation of an interactive method and the software utilized (referred as to simulator software in what follows) for modeling the optimization problem. To this end, we consider two different alternatives: (*i*) including the interactive method as a part of the simulator software, or (*ii*) defining the optimization method and the simulator software as two independent components connected with a communications layer.

For the first alternative, the most simplistic approach is to connect the implementation of an optimization method very tightly to the simulator software to form a single software package. With such a tight coupling, any changes either in the problem model or in the implementation of the interactive method require the reconstruction of the whole software package. This usually cannot be done without full access to both. The connection can be loosened, for example by utilizing

so-called call-back functions, where instead of directly connecting the implementation of the optimization method to the simulator software, a part of the other component is passed to the other component. The relevant part can then be used whenever needed by the receiving component. For example, the implementation of the optimization method can be provided with information on the location of the machine code used for calculating the objective and constraint function values in the computer's memory. By providing this machine code with information on the decision variable values, the function values can be calculated whenever needed by the implementation of the method. By clearly defining the format of the passed information, the implementation of the optimization method and the simulator software can be easily replaced with some other implementation or software. However, even with this approach, the interactive method and the simulator software are combined to form a single software package and replacing either part of the package can prove to be difficult.

Furthermore, when using the call-back function approach, the implementation of the optimization method and the simulator software share the process and memory space with each other. This makes it difficult to distribute different parts of the software package between different processors or computers. As distributing the calculations is a commonly used technique for dealing with computationally expensive problems, in [PIV] we suggest loosening the connection between these components with the second proposed alternative, i.e. with a communications layer. In this way, we can separate the implementation of the optimization method from the simulator software into clearly distinct components. This layer can be as simple as a predefined format for input and output files that are written and read by the different components as needed. Using such files still restricts the available options when operating in a distributed environment. Therefore, in [PIV] we have developed a socket based communications layer, referred to in this thesis as a *coupling interface* that can be used for communication between the implementation of the optimization method and the simulator software in varied environments.

The sequence diagram of the coupling interface can be seen in Figure 3. Here, the problem model side of the interface offers two requests, i.e., *getProblem()* and *evaluate()*. When called, the getProblem request returns a problemDescription object that contains the description of the optimization problem, including problem dimensions (such as the number of objective functions and decision variables) and information on the problem characteristics (such as if decision variables are continuous or discrete). If the problem model is not defined, or incomplete, the getProblem request can cause an invalidProblem message to be sent. This message must then be processed by the interactive method implementation, for example by notifying the DM of the error and stopping the solution process.

After the interactive method implementation has obtained problem description, it can formulate the scalarized subproblem(s) and start the single objective optimization. The objective and constraint function values for the optimization are obtained by the *evaluate()* request of the problem model. By passing the deci-

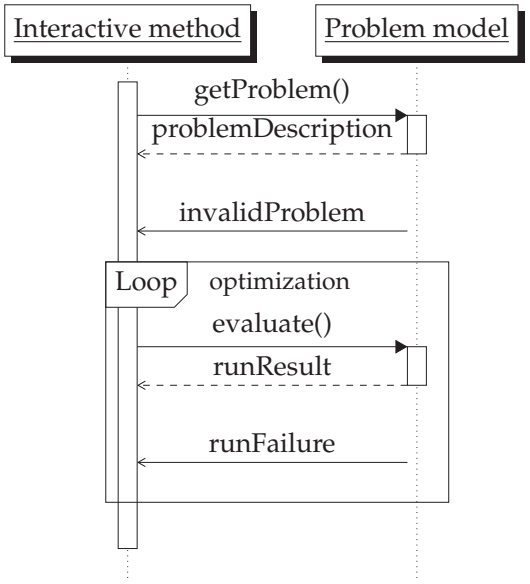


FIGURE 3 Sequence diagram of the coupling interface

sion variable values with the evaluate request to the problem model, the implementation of the interactive method can obtain the function values as a runResult object. The evaluate requests are continued in an optimization loop until the optimization finishes. It is possible for the evaluation to fail during the optimization, for example, if some of the functions could not be evaluated for the given decision variable values. If so, the problem model sends a runFailure message. Naturally, the implementation of the method must then consider if the optimization loop must be interrupted or if the evaluate requests should be repeated with other decision variable values.

When utilizing the coupling interface as suggested in [PIV], the problem model is clearly separated from the implementation of the optimization method. This means that as the access to the optimization problem model is available only through the coupling interface, the problem model can be changed without any modifications to the implementation of the method. Similarly, if the tools used for formulating the problem model use the coupling interface, using an implementation of another interactive method for solving the problem model should be a straightforward task.

In this chapter we have given an insight into how to utilize interactive methods for computationally expensive problems. We have noted that the DM involved with the interactive solution process can provide information on the validity of the problem model. We have described how this information could be utilized by proposing the AIMO algorithm. Finally, we have introduced the coupling interface that can be used for building a flexible connection between an optimization method and a problem model. In the next chapter, we continue with a further discussion on the issues related to implementing an interactive method.

4 CONTRIBUTION WITH RESPECT TO IMPLEMENTATION

Several interactive multiobjective optimization methods have been discussed in the literature (see, e.g. [47, 55, 67] and references therein), but t not that many implementations of them exist (see e.g. [75, 101]). This can be considered to be somewhat surprising, as these interactive methods have been successfully used to solve multiobjective optimization problems for various topics, including reservoir management [3], operating a wastewater treatment plant [35, 36], the construction of bridges [69] and analyzing air pollution [92], among others.

In this chapter, we present observations and hints gathered during this research for enabling the easy implementation of interactive methods. In our research, we have identified a set of core steps that can be used as a framework when building practical implementation of an interactive multiobjective optimization method. We consider this an important topic, as without a practical implementations of methods it is not possible to apply new methods for solving multiobjective optimization problems.

In what follows, we first describe a core structure that has been formed by using the identified steps. Then we briefly provide details on how, by following the core structure, we have constructed the IND-NIMBUS software framework that includes several interactive method implementations as well as tools for connecting problem models to the implementations. One can see IND-NIMBUS as a concrete example of how the core structure can be utilized in implementing interactive methods. We continue by providing details of the GAMS-NIMBUS Tool and the IND-NIMBUS PAINT module developed in this thesis. We finish this chapter with a short overview of the user interface the different DMs have used while solving the problems considered during the research of this thesis.

4.1 Core Structure of Interactive Methods

In this section we describe the characteristics common to scalarization-based interactive multiobjective optimization methods. The given description is intended as a general guideline on building an implementation of an interactive multiobjective optimization method. We do not consider one particular method; the characteristics apply for all scalarization-based interactive methods.

As described in Chapter 2, the solution process of an interactive multiobjective optimization method is iterative. The DM is first shown the objective function values of a Pareto optimal solution and is then asked to specify his or her preferences in relation to that solution. This preference information is used to generate new, more satisfactory Pareto optimal solution(s). The process is continued with the DM specifying preferences for the new Pareto optimal solution (or for a previous one, if the DM did not find the new solution(s) satisfactory). By following this process, the DM can direct the exploration of the Pareto frontier in the directions that (s)he is interested in. The aim of this solution process is to support the DM to find a single Pareto optimal solution that (s)he considers the most preferred of the solutions found in the Pareto frontier (see, e.g. [67]). Thus, the iterative process is continued until the DM feels that such a Pareto optimal solution has been found.

Even though the solution process described above seems to be quite straightforward, previous experiences have shown that such a literal description does not necessarily offer enough insight into different components and their roles for researchers who are not familiar with interactive methods to enable them to construct an implementation of an interactive method successfully. Based on the results presented in [PI], it is evident that a need exists for a more formal presentation of the structure of interactive multiobjective optimization methods . By following the general interactive approaches described in [47, 78], in [PII] we present a *core structure* of an interactive multiobjective optimization method. The core structure describes the steps needed for implementing an interactive method, as well as the information flow between different components of the structure. Naturally, each interactive method includes different details, which implies there are various needs for each implementation. Nevertheless, the core structure presented here offers an insight into how an interactive method can be implemented, simplifying the process, especially when constructing a software tool including different interactive methods.

In general, an interactive multiobjective optimization method can be considered to be constructed from the following six steps:

1. Initialize, e.g., calculate the ideal and nadir objective vectors.
2. Solve a method-specific subproblem to generate an initial Pareto optimal solution to be used as a current solution.
3. Ask the DM to provide preference information related to the current solution.

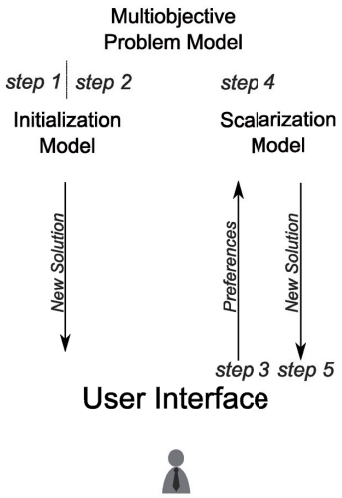


FIGURE 4 Core structure of an interactive method

4. Generate new solution(s) based on the preference information by solving appropriate subproblem(s).

5. Ask the DM to select the best solution of the previously generated solutions and denote it as the current solution.

6. If the selected current solution is satisfactory, stop. Otherwise continue from step 3.

As noted in [PII], two distinct parts can be identified from the core structure: the *user interface* part (steps 3 and 5) and the *algorithm* part (steps 1, 2 and 4). The information flow between different components of the core structure can be seen in Figure 4. As can be seen, steps 1 and 2 of the core structure do not need information from the DM and can be combined into a single *initialization model*. The initialization model produces the initial Pareto optimal solution (in addition to other possible initialization routines specified by the interactive method in question) that is conveyed to step 3. The DM is then asked to specify his or her preferences, which are provided to step 4 of the core structure.

In step 4, the preference information is used to formulate a scalarized single objective subproblem. As mentioned earlier, the scalarized subproblem, i.e., *the scalarization model*, is formulated in such a way that the generated solution is Pareto optimal and corresponds to the preferences specified by the DM. Different interactive methods ask the DM to specify the preference information in different ways, and naturally different methods use different scalarizations when formulating the subproblem (see, e.g. [15, 64, 67, 70, 78, 102]).

The Pareto optimal solution obtained with the scalarized subproblem is then conveyed to step 5 of the core structure. The DM then selects a new current solution and either continues to step 3 in order to specify new preference information or selects the current solution as the final solution for the problem (as per step 6).

The core structure has been used as a guideline by the author for implementing several different interactive methods. By following the core structure and the coupling interface described in Chapter 3, we were able to construct the IND-NIMBUS software framework that we have used as the foundation for the research conducted in this thesis.

4.2 The IND-NIMBUS Software Framework

One of the main aims of this thesis has been finding ways of increasing the use of interactive methods for solving multiobjective optimization problems. To this end, the applicability of the results presented in this thesis has been demonstrated by software implementations and by utilizing the implementations for solving a wide array of multiobjective optimization problems. The main apparatus for this has been the IND-NIMBUS software framework.

The IND-NIMBUS software framework is a multi-platform framework providing tools for implementing different interactive multiobjective optimization methods. The design of the IND-NIMBUS framework follows the core structure of the interactive methods described in Section 4.1; that is, the framework has been clearly separated into the user interface, algorithm and problem model parts. Therefore, the IND-NIMBUS software framework can be utilized for implementing several different interactive multiobjective optimization methods in the same environment. As an example of such methods, the IND-NIMBUS framework contains implementations of the NIMBUS [64] and the Pareto Navigator [27] methods as well as the PAINT surrogate method [37]. These implementations reuse the user interface and algorithm components in different methods. This has led to an environment where new methods can be implemented conveniently and in a timely manner.

The IND-NIMBUS framework does not support formulating multiobjective optimization problems. Instead, problems modeled with different simulators or programming languages can be connected to the framework with different integration tools. Examples of the use of such integration tools include the MATLAB® environment used for radiotherapy treatment planning [79, 80], the Balas® process simulation software for chemical process design [32, 34], the Numerrin environment used in optimal shape design [39], the GAMS environment used for optimization of a heat exchanger network [PI, 44] and an oxyfuel power plant process [100], the GPS-X™ simulator used for the optimization of operating a wastewater treatment plant [PIII, 35, 36], the APROS dynamic process simulator used for optimization of a two-stage separation process [PIV] and for plant automation [89] and almost any programming language, such as C, C++, and Fortran [40, 49, 58, 65].

The IND-NIMBUS software framework has been invaluable for this thesis. As the framework follows the core structure, it has offered a convenient basis for different interactive methods. By utilizing the coupling interface described

in Section 3.3 it has been possible to use different interactive methods for solving the same multiobjective optimization problem without the need to adapt the problem. In addition, the clear definition of the coupling interface has enabled a convenient way for connecting the interactive methods implemented within the IND-NIMBUS framework with different simulation and modeling tools.

The results presented in this thesis have been obtained with methods and tools implemented in the IND-NIMBUS framework. Without the possibility of being able to utilize this framework, the research undertaken in this thesis would have taken a greater effort, and in part, could have been impossible to carry out. In the next section, we provide brief description of the implementations of the different methods used in this research.

4.3 Method Implementations

As mentioned earlier, the IND-NIMBUS software framework has been implemented following the core structure described in [PII], where the algorithm part of the interactive multiobjective optimization method is clearly separated from the user interface part as well as from the underlying multiobjective optimization problem. In this section, we will briefly describe the different methods implemented within the IND-NIMBUS software framework during the research of this thesis. These include an implementation of the NIMBUS method [64] for the GAMS modeling environment as well as an implementation of the PAINT surrogate method [37].

In addition to the implementations described here, the IND-NIMBUS framework includes another implementation of the NIMBUS method, called the NIMBUS Kernel, and an implementation of the Pareto Navigator method [27, 97]. As mentioned earlier, the Pareto Navigator method is suitable only for convex problems and is not discussed in this thesis. The NIMBUS Kernel is an implementation of the NIMBUS method that traces it roots to the WWW-NIMBUS optimization system operating on the Internet [62]. Originally, the NIMBUS Kernel did not follow the core structure described in [PII], but later on it was refactored (on refactoring, see e.g. [54] and references therein) to reflect the core structure. The NIMBUS Kernel is utilized for the research carried out in article [PV] included in this thesis, but here we concentrate on the more recent implementation of the GAMS-NIMBUS Tool. Nevertheless, it should be noted that in addition to the scalarized subproblems of the NIMBUS method described in Section 2.3, the NIMBUS Kernel contains several single objective optimization methods that can be used for solving the subproblems. These include global optimization methods such as differential evolution [94], a controlled random search [77] and genetic algorithm [66], as well as local optimization methods such as COBYLA [76] and a proximal bundle method if the subgradient information is available [72]. As they are not considered in this thesis, they are not discussed further.

Next, we will describe the GAMS-NIMBUS Tool that has been implemented

following the steps given in the core structure. Then we will describe the IND-NIMBUS PAINT module that can be used for solving computationally expensive problems with a surrogate problem constructed by the PAINT method using the NIMBUS method.

4.3.1 GAMS-NIMBUS Tool

The GAMS modeling environment [14] is a widely used modeling system for mathematical optimization. It is aimed at solving single objective optimization problems, but GAMS has been applied to problems with several, conflicting objectives (e.g. [1, 53]). To our knowledge, no implementations existed for interactive multiobjective optimization methods for the GAMS modeling environment prior to the implementation of the interactive NIMBUS method proposed in [PI, PII].

In [PI], we consider the multiobjective heat exchanger network synthesis (synheat) problem that was originally solved using the weighted sum method [105]. The problem model was reformulated for [PI] by adding the GAMS formulations of scalarized subproblems needed by the NIMBUS method and by removing the formulations related to the originally used weighted sum method. This resulted in a Synheat-NIMBUS GAMS model, which is a combination of the synheat problem model and of the NIMBUS method. The Synheat-NIMBUS model is constructed manually, which is a laborious and error-prone process, and the NIMBUS method was only partially implemented. For example, it contained only the standard scalarized subproblem of the NIMBUS method and generated only a single solution per iteration instead of the (up to) four generated by the NIMBUS method. Furthermore, the formulations contained minor errors and some of the obtained solutions were not Pareto optimal.

In addition, the Synheat-NIMBUS model is specific to the heat exchanger network synthesis problem and cannot be directly utilized for solving different multiobjective optimization problems. Due to this, and other issues identified in [PI], we decided to give a detailed description of how to solve the dilemma of implementing an interactive method in [PII]. This resulted in the core structure described in Section 4.1. Based on the core structure, in [PII] we provide insights into how to further develop the Synheat-NIMBUS model in order construct a tool that can be used to solve almost any multiobjective optimization problem modeled in a GAMS modeling environment with the interactive NIMBUS method. This research resulted in the GAMS-NIMBUS Tool.

As per the core structure described in Section 4.1, in the GAMS-NIMBUS Tool the GAMS models describing the subproblems of the NIMBUS method are separated from the GAMS problem model into distinct scalarization models. Due to this separation, it is possible to modify the problem model without modifying the method-specific subproblem models, and vice versa. This means that even though the GAMS-NIMBUS Tool is motivated by solving the heat exchanger network synthesis problem described in [PI], it can be easily used for solving different problem models. In [PII], we demonstrate this by using the GAMS-NIMBUS

Tool to solve a power generation problem [52] modeled with GAMS in addition to the heat exchanger network synthesis problem.

With the GAMS-NIMBUS Tool, we can provide the GAMS community with a well-studied interactive method, as well as with mature user interface for using the method. In [PII], we have demonstrated how to apply the steps of the core structure when implementing the GAMS-NIMBUS Tool, and by following similar steps it should be possible to implement other interactive methods either for GAMS or for any other environment such as with other modeling (i.e. AMPL [28] or AIMS [11]) or programming languages. Such implementation would allow for utilizing the advantages of the interactive multiobjective optimization methods for solving problems modeled in the environment. On the other hand, the implementation would also allow the scalarization-based interactive methods to gain the benefit of being able to solve scalarized subproblems with the efficient single objective optimization methods included in the environment.

4.3.2 IND-NIMBUS PAINT Module

As mentioned in Section 3.1, when solving computationally expensive multiobjective optimization problems with an interactive method, a fruitful approach may be to replace the original problem with a computationally less expensive surrogate problem. To accomplish the use of surrogate problems in the IND-NIMBUS software framework, in [PIII] we implement the PAINT method [37] for the framework. With this so-called IND-NIMBUS PAINT module, we can construct a surrogate problem that can be solved with the interactive NIMBUS method.

As described in [PIII], when a problem is solved with the IND-NIMBUS PAINT module, the module constructs a surrogate problem based on a set of Pareto optimal solutions generated with an a posteriori method. Then the preferences specified by the DM are used to generate approximate Pareto optimal solutions, i.e., the Pareto optimal solutions for the surrogate problem. At any point during the solution process, the DM can decide to project an approximate solution onto the Pareto frontier of the original problem using the achievement scalarization function [102], as described in e.g. [PII].

As the IND-NIMBUS PAINT module follows the coupling interface described in Section 3.3 and the core structure described in Section 4.1, it is not attached to any distinct problem model and a surrogate problem can be constructed for any problem model on the IND-NIMBUS framework. Furthermore, the surrogate problem can be used with any interactive method included in the framework. Similarly, the PAINT method of the IND-NIMBUS PAINT module can be replaced with any other method that uses a set of Pareto optimal solutions to construct the surrogate problem.

In [PIII], the IND-NIMBUS PAINT module is utilized for constructing a surrogate problem for the computationally expensive problem of operating a wastewater treatment plant. The surrogate problem was explored by the DM using the interactive NIMBUS method, producing solutions for the surrogate problem. Af-

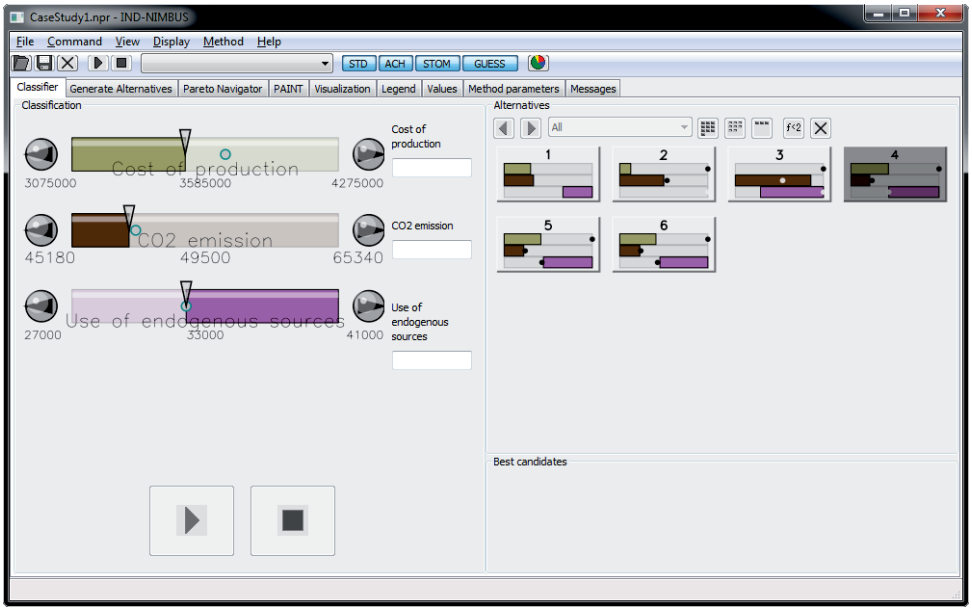


FIGURE 5 The IND-NIMBUS software framework

ter projecting these solutions onto the Pareto frontier of the original problem, the DM deemed them to be satisfactory for the problem. Nevertheless, for some of the projected solutions, the errors between the solution for the surrogate problem and projected solutions seemed to be very high in relation to the ideal and nadir values. Therefore, the IND-NIMBUS PAINT module was further studied in [PV], where the accuracy of the PAINT surrogate problem is increased in the areas that the DM is interested in. Unfortunately, due to licensing issues, the problem of operating wastewater treatment plant could not be considered in [PV]. It should be noted that the problem was previously solved in [35, 36] with the NIMBUS method, and as the NIMBUS implementation utilizes the same coupling interface as the IND-NIMBUS PAINT module, the problem could be used for this research without any additional effort.

4.4 User Interface

In addition to the implementations of different interactive methods and coupling interfaces for connecting problem models to the framework, the IND-NIMBUS software framework contains several graphical user interfaces. Currently, the IND-NIMBUS framework contains two different graphical user interfaces for specifying the classification information for the NIMBUS method as well as two different graphical user interfaces for navigating the Pareto frontier with the Pareto Navigator method. As mentioned earlier, the Pareto Navigator is only suitable for convex problems, and therefore we do not consider it in this thesis. As the

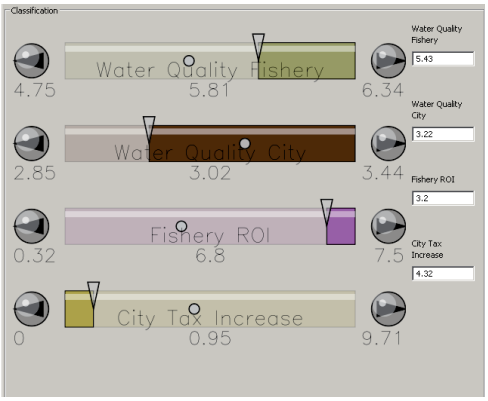


FIGURE 6 NIMBUS classification view

functionality of the two NIMBUS method user interfaces is more or less identical, we give a brief description of only the most commonly used classification view.

The initial view of the IND-NIMBUS software framework can be seen on Figure 5. In the figure, the DM has opened a problem model with three objectives and then used the NIMBUS method to produce six Pareto optimal solutions for the problem. The user interface of the NIMBUS method is located on the left side of the figure, where the objective function values of the current solution can be seen. The six generated Pareto optimal solutions can be seen in the right side of the figure. Here each bar-chart corresponds to a single Pareto optimal solution.

When using the interactive method, the DM first specifies his or her preferences on the right side. The new Pareto optimal solution(s) can then be requested by pressing the "play" button shown at the bottom of the figure. At any point, the DM can decide to stop the current calculations by pressing the "stop" button. When the new Pareto optimal solution is generated, it is set as the current solution and shown on the left side. In addition, it is added to the set of Pareto optimal solutions shown on the right. The DM can then continue with the current solution, or select another solution as the current solution from the right side. On the IND-NIMBUS toolbar (located at the top of the figure), the DM can decide which subproblems of the method (s)he would like to utilize and by which single objective optimization method the subproblem(s) is to be solved.

As mentioned earlier, the IND-NIMBUS framework includes several different interactive methods. In Figure 5, this is reflected by the tab leaflets below the toolbar. In the tab leaflets, names of the NIMBUS, PAINT and Pareto Navigator methods can be seen. When a problem model has been opened in the framework, the DM can change the used method simply by clicking the tab leaflet for the method (s)he would like to use.

In Figure 6, we give an example of how the NIMBUS classification is specified by the DM when using the NIMBUS method of the IND-NIMBUS framework. Here, the DM is shown the current Pareto optimal solution for which (s)he is asked to provide his or her preferences. The problem being solved has four objective functions, each of which is represented by one of the four horizontal bars

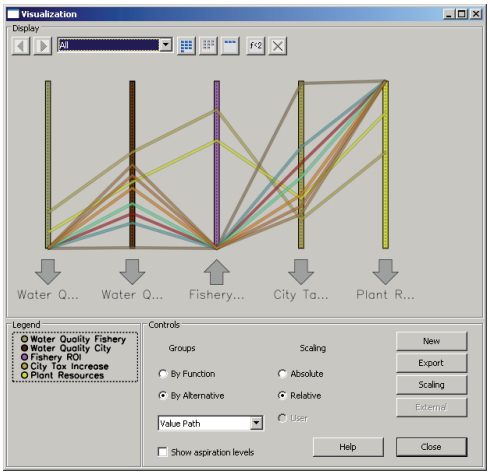


FIGURE 7 Visualization view

on the figure. The first three of the objective functions are to be maximized, as indicated by the placement of the bar on the right side. The fourth objective function is to be minimized. The ranges and current values of the objective functions are given as numbers below each objective function bar.

The classification view differs slightly from the presentation of the NIMBUS method given in Section 2.3. The IND-NIMBUS classification view does not refer to the five different NIMBUS classes. Instead, the DM is simply asked to click the different parts of the objective function bar to indicate how (s)he would like to change the shown Pareto optimal solution. If the DM wishes to decrease the objective function value, (s)he clicks the arrow pointing to the left. If instead (s)he wishes for an increase, the DM clicks the arrow pointing to the right. If the DM wishes to obtain some particular value, (s)he click the bar itself, and the corresponding value is then shown in the box next to the objective function bar, where the value can be edited. If the DM deems the current objective function value suitable, (s)he clicks the arrow pointing downwards. For more details on the classification view, see [PII]. When using the NIMBUS method with the IND-NIMBUS PAINT module described in [PIII], the DM uses the same classification view as shown in Figure 6.

In addition to the graphical user interfaces of the interactive methods, the IND-NIMBUS framework contains tools for examining and comparing the Pareto optimal solutions. The framework contains tools for the graphical visualization of the Pareto optimal solutions, including different bar charts, a spider web chart, value path, whisker plot, petal diagram, and a multi-way dot. An example of the value path visualization can be seen in Figure 7 where the DM is examining eight Pareto optimal solutions for the problem shown in Figure 6. For more details on visualizations available in the IND-NIMBUS framework, see [59]). In addition to the included graphics, the Pareto optimal solutions can be visualized by exporting the objective function and decision variable values to an external software, such as Microsoft Excel.

To organize the Pareto optimal solutions, the DM can filter the solutions based on the values of the objective and constraint functions as well as decision variables. For example, the DM may decide to have a look at only those solutions where the first objective function has values above a certain limit. There is also the possibility of being able to review the solutions by iterations, i.e., to only show the solutions generated from the same preference information. It is also possible to step back in the solution process, i.e., to view the previous iterations. At any point, the DM can store the best solution candidates in a specific list that can be considered separately. Any undesirable solution can be deleted from the set of solutions. In addition, the DM can view the objective function and decision variable values in a table, and (s)he can change the parameters of the single objective optimization methods if so desired.

In the following chapter we describe how the DM can be supported when solving a computationally expensive problem with an interactive method. We utilize the coupling interface described in Chapter 3 and the IND-NIMBUS framework with the IND-NIMBUS PAINT module along the user interface of the NIMBUS method described in this chapter to construct an agent assisted algorithm that can overcome some of the obstacles encountered when solving a computationally expensive problem with an interactive method.

5 A NEW AGENT ASSISTED INTERACTIVE MULTIOBJECTIVE OPTIMIZATION ALGORITHM

As mentioned earlier, when solving a computationally expensive multiobjective optimization problem with an interactive method, the waiting times imposed on the DM may affect the solution process negatively. This can be alleviated by replacing the original problem with a computationally less expensive surrogate problem. However, when using an interactive method with a surrogate problem, the DM is not provided with Pareto optimal solutions. Instead, (s)he is provided with approximate Pareto optimal solutions, i.e., Pareto optimal solutions for the surrogate problem, and they must be projected to obtain Pareto optimal solutions for the original problem (for more details, see [PIII]). Naturally, it is possible that the surrogate problem is not accurate enough to provide the DM with approximate Pareto optimal solutions that are close to the actual Pareto frontier. In the worst case, if the surrogate problem is too inaccurate for the specific areas that the DM is interested in, the approximated Pareto optimal solutions may give the DM misleading information of the actual Pareto frontier. In this case, the effort spent to find the approximate Pareto optimal solutions may be wasted.

In order to minimize the effect of an inaccurate surrogate problem, in [PV] we propose a new approach for utilizing a surrogate problem when solving computationally expensive multiobjective optimization problems with an interactive method. In the new approach, we use intelligent agents to update the surrogate problem based on the preference information specified by the DM during the solution process. In this manner we are able to acknowledge the information provided by the DM even if the approximated Pareto optimal solutions have not been accurate enough (for the DM) and increase the accuracy of the surrogate for the areas that are of interest to the DM.

In what follows, we briefly describe the main details of the algorithm. We first presented the agent assisted algorithm as a general set of steps that can be used to extend an interactive method when it is used alongside with a surrogate problem. We then continue with details of the different agents used in the agent assisted algorithm. We finish this chapter by giving a concrete example, where we combine the existing implementation of the NIMBUS method described in Chap-

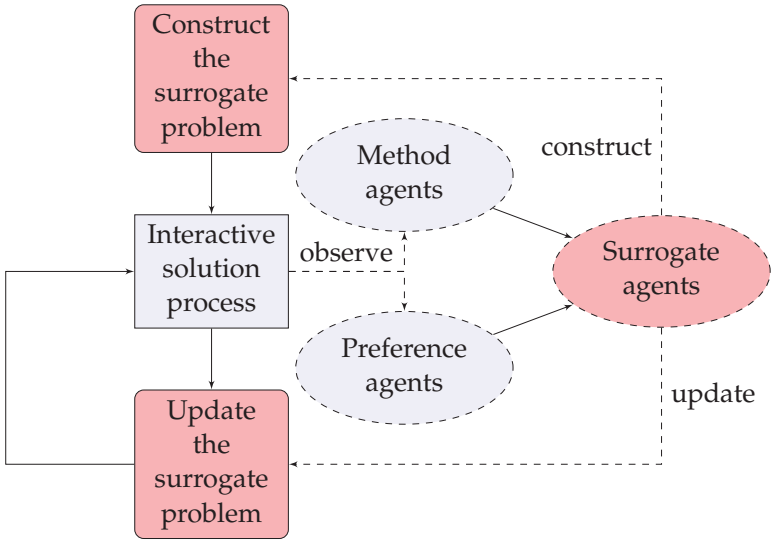


FIGURE 8 Flowchart of the agent assisted algorithm

ter 2 with the IND-NIMBUS PAINT module introduced in [PIII] to construct an agent assisted algorithm for solving computationally demanding multiobjective optimization problem..

5.1 Description of the Agent Assisted Algorithm

The main idea of the agent assisted algorithm is to extend an interactive method so that the waiting times experienced by the DM are minimized. This is achieved by utilizing a surrogate problem during the interactive solution process and updating the surrogate problem based on the DM's preferences, but without involving (him or her in the update process. In addition, the agent assisted algorithm strives to decrease the cognitive load of the DM by decreasing the amount of preference that is information asked for.

The overall progress of the agent assisted algorithm can be seen in Figure 8. As can be seen, the algorithm consists of three main phases. First, a surrogate problem is constructed and then an interactive method is used by the DM to find the most preferred solution for the surrogate problem. Finally, the surrogate problem is updated to increase its accuracy for the areas of the Pareto frontier that the DM is interested in. As the DM is not involved in the first and the third phases, they are called *offline phases* (o.p.) and the second phase is called a *decision phase* (d.p). In Figure 8, the components of the offline phase have darker (red) backgrounds, while the decision phase components have lighter (blue) backgrounds.

As can be seen in the flowchart presented in Figure 8, the agent assisted algorithm utilizes three different types of agents (pictured as ellipses). In the decision phase, the agent assisted algorithm involves *the preference agents* that observe

the preference information expressed by the DM in order to build a preference model. At the same time, *the method agents* collect information on how the interactive method is used to generate new Pareto optimal solutions. The third type of agents, i.e. *the surrogate agents*, operate during the offline phase and are responsible for generating new Pareto optimal solutions used to construct the surrogate problem.

To provide more detail, the agent assisted algorithm can be further divided into the following six steps:

1. (o.p.) The surrogate method constructs the surrogate problem using all previously generated set of Pareto optimal solutions
2. (d.p.) The DM starts using the interactive method and specifies preference information as per the used interactive method.
   - Set decision phase iteration $t = 1$.
   - Preference agents collect the preference information to build a model of the DM's preferences.
3. (d.p.) The interactive method generates approximate Pareto optimal solution(s) to the surrogate problem to be shown to the DM.
   - The method agents collect information on how approximate Pareto optimal solutions are generated.
4. (d.p.) The DM selects one approximate Pareto optimal solution either
   (a) as the new starting solution for the next iteration, set $t = t + 1$ and continues with step 2, or
   (b) as the most preferred solution for the surrogate problem and continues to step 5.
5. (o.p.) The surrogate agents generate new Pareto optimal solutions for the original problem based on the information collected by the method agents and the preference agents.
6. The preference agents select Pareto optimal solutions that are shown to the DM with information on solution accuracy, if available. The DM either
   (a) continues with step 1, or
   (b) selects one as the most preferred solution for the original problem and stops.

By ensuring that the operations done in the decision phase are computationally inexpensive we can provide the DM with approximate Pareto optimal solutions in a timely fashion. At the same time, the computationally expensive operations, such as constructing the surrogate problem and generating Pareto optimal solutions for the original, computationally expensive problem are handled in the offline phases without requiring the involvement of the DM, and are then shown to the DM when (s)he so wishes.

5.2 Agents Used in the Algorithm

Next we provide details on the three types of agents utilized in the agent assisted algorithm. It should be noted that each individual agent can be implemented using different approaches, and here we outline the main features of the agents implemented in [PV]. In addition, here we present only the general features of the agents without method-specific details, even though details of different agent implementations depend on the specific interactive method and surrogate problem used, i.e., the NIMBUS method and the PAINT surrogate problem in [PV].

It should be noted that the fourth type of agents, i.e., *the optimization agents* presented in [PV], are specific to the problem being solved. In this thesis we deal with interactive multiobjective optimization methods, and while the methods that we are concerned with do produce new Pareto optimal solutions by solving single objective optimization problems, the single objective optimization methods and their parameters are usually selected based on the characteristics of the problem. The optimization agents used in [PV] were suitable for that particular problem, but we did not study their applicability in general. For other optimization problems, the optimization agents can simply be replaced by a suitable single objective optimization method.

5.2.1 Preference Agents

One of the central parts of the agent assisted algorithm is the model of the DM's preferences, i.e., the model that is used to identify the areas of the Pareto frontier that are interesting to the DM. In what follows, the model of the DM's preferences is called a *preference model*. The preference model is constructed by observing the preferences specified by the DM during the interactive solution process. This learning process is called *preference learning*, for short. It should be noted that we cannot assume that the DM is able to undertake hundreds of iterations of the solution process, and therefore the selected preference modeling technique cannot depend on large amounts of data.

The objective of a preference agent is to use the preference model to generate new preference information that should correspond to the preferences of the DM without requiring additional input from the DM. That is, after the preference model has been constructed, an interactive method can be utilized for solving a multiobjective problem without the involvement of the DM by using the preference agent in place of the DM. Naturally, the accuracy of the preference model determines how well the preference information generated by the preference agent corresponds to the DM's preferences and ultimately how satisfactory the solutions that are generated are to the DM.

As mentioned earlier, when using an interactive method, the DM will continue until (s)he finds the solution (s)he prefers the most. During the solution process, the DM can change his or her preferences, based on the Pareto optimal solutions shown to him or her. Ideally, the DM continues until (s)he is confident

that the final solution is the best solution. Therefore, no clearly defined point exits at which the preference learning has been completed. In the agent assisted algorithm as described in [PV], the preference agents continue the preference learning until the DM decides to stop the decision phase. Nevertheless, the preference agents can be used for generating new preference information at any point during the algorithm. For example, during the decision phase the DM could be provided with Pareto optimal solutions that are generated based on the preference information provided by the preference agents. If the preference model is accurate, these solutions could be of interest to the DM, even if they are concerned with areas that (s)he has not yet considered.

In [PV] we identified two different preference learning approaches. First, when utilizing a *computer learning* approach, we assume that the DM has a more or less unchanging preference model for the duration of the interactive solution process. When utilizing *human learning*, we assume that the DM learns and adapts his or her preferences during the interactive solution process. In [PV], we have implemented agents following both of these approaches, as it cannot be known in advance whether the DM is following a computer or a human learning approach or some mixture of them.

As the interactive method consists of selecting the most suitable Pareto optimal solution and then specifying classification information in relation to that solution, in [PV] we further divided the preference agent into two subtypes, i.e., into *selecting* and *classification* preference agents.

In the agent assisted algorithm, the preference agents are utilized for generating new Pareto optimal solutions in the second offline phase by replacing the DM in the interactive solution process for solving the original problem instead of the surrogate problem. That is, the approximate Pareto optimal solutions generated during the decision phase are projected onto the Pareto frontier. Then the selecting preference agents are used to select the most preferred of them and the classification preference agents are used to provide preference information for those solutions. The interactive process can be continued with the agents until they do not generate new preference information or until a preset time limit has been reached.

5.2.2 Method Agents

The method agents are used to find Pareto optimal solutions that correspond to the preference information specified either by the DM during the decision phase or by the preference agents. These solutions are solutions to the original multiobjective optimization problem that is assumed to be computationally expensive. As one of the main aims of the agent assisted algorithm is to reduce waiting times imposed on the DM when using an interactive method, the method agents should be used only during the offline phase of the agent assisted algorithm and the new Pareto optimal solutions should be presented to the DM only after the offline phase has been finished.

In practice, when using the preference information specified by the DM, the

method agent solves the same scalarized subproblem during the offline phase for the original problem that was solved for the surrogate problem during the decision phase. In effect, if the surrogate problem is accurate enough, the method agent finds the solution that the DM would have found if (s)he had not used the surrogate problem. In addition, the method agents can be used for projecting the approximate Pareto optimal solutions onto the Pareto frontier of the original problem.

Therefore, the method agents may produce two different Pareto optimal solutions for the original problem per each approximate Pareto optimal solution. When the offline phase has finished, both of these solutions can be shown to the DM, but if the DM has iterated for a high number of decision phases, this may lead to a very high number of solutions. In such a case, the selecting preference agent can be used to choose the solutions to be shown to the DM.

5.2.3 Surrogate Agents

One of the aims of the agent assisted algorithm is to update the surrogate problem intelligently for those areas that are of interest to the DM. How this is achieved naturally depends on how the surrogate problem is constructed. If, as in [PV], the surrogate problem is constructed from a set of Pareto optimal solutions, the accuracy of the surrogate can be increased by generating new Pareto optimal solutions that are located in the areas that the DM is interested in and then updating the surrogate problem using all of the obtained solutions.

The initial set of Pareto optimal solutions, i.e., the construction set that is used to construct the surrogate problem in step 1 of the agent assisted algorithm can be generated with any a posteriori-type of multiobjective optimization method that generates many Pareto optimal solutions (see e.g. [13, 55, 85]). After the first decision phase, the surrogate agents update the construction set based on the gathered information. For this purpose, we have identified two types of surrogate agents. The *projecting* surrogate agents utilize method agents to generate new Pareto optimal solutions of the original problem as described in Section 5.2.2. On the other hand, the *decision* surrogate agents repeat the decision phase of the agent assisted algorithm using the selecting and classification preference agents as the DM. That is, for each iteration of the decision phase, a decision surrogate agent uses a selecting agent to choose a Pareto optimal solution from the set of Pareto optimal solutions generated by the method agents and then specifies preference information in relation to the chosen solution with a classification preference agent.

Depending on the technique used for preference agents, it is possible that the new preference information, and therefore the new Pareto optimal solutions, will not correspond to the DM's preferences. As the new solutions are Pareto optimal, the accuracy of the surrogate should not be affected even if the new solutions are generated for the wrong area of the Pareto frontier. On the other hand, an excessive number of Pareto optimal solutions can increase the computational cost of constructing or updating the surrogate problem, but as this is done during

the offline phases, it should not affect the decision phase. Therefore, the increase in the computation cost during the surrogate construction, and in effect the time taken, is acceptable, as the aim of the agent assisted algorithm is to reduce time demands on the part of the DM.

5.3 Example Implementation of the Agent Assisted Algorithm

Now that we have described the agent assisted algorithm as a general structure that does not depend on any specific methods, we give details of an example implementation of the agent assisted algorithm used in [PV] for solving a computationally expensive two-stage separation process problem. In [PV], we implement the agent assisted algorithm using the NIMBUS implementation available from the IND-NIMBUS software framework [PII] as the interactive method and the PAINT method [37] for constructing the surrogate problem. To implement the multiagent system and learning features of the agents, we use libraries commonly available for the Python programming language. As these libraries are not relevant to this thesis, they are not further described here.

In what follows, we give detailed examples of three different preference agents utilized during the solution process described in [PV]. We first describe how two preference agents utilizing computer learning can be used to select the most preferred solution from a set of solutions and then to specify preference information in relation to the selected solution. We continue by showing how a preference agent utilizing the human learning approach generates new preference information based on the preferences specified by the DM. We also provide numerical examples for both of these approaches.

The solution process of the NIMBUS method for solving the two-stage separation problem in [PIV] is shown in Table 1. Here, in each iteration, the DM is first shown the current Pareto optimal solution and is then asked to provide a NIMBUS classification in relation to that solution. The DM is then shown two to three new Pareto optimal solutions and is asked to select the most preferred solution from among them. The most preferred solution is then used as the current Pareto optimal solution for the next iteration. The final solution is shown on the last row of the table. Next we will give an example of how this solution process can be used to create selecting and classification preference agents, which can be used for a new solution process without involving a human DM. Naturally, how well the selection follows the DM's preferences depends on the amount of training data and on the features of the used learning technique. In practice, agents cannot be used to replace the human DM, as this would require the DM to iterate for an impractical number of iterations.

As described in Section 4.1, the first step taken by the DM in the interactive solution process is to provide preference information for the current Pareto optimal solution. With the agent assisted algorithm, this is achieved by the classification preference agent. As an example, we next give details on the classifi-

TABLE 1 Solution process of the two-stage separation problem in [PIV]

Iter	Issue	Max Permeate (kg)	Min Impurity	Min Energy (kJ)
	z^1	2222	11.02	16842
1	Classif^1	$I^{\geq}$=2000	$I^{\leq}$=2.30	$I^{\leq}$=9500
	z^2	1732	3.92	12402
	z^3	1483	2.02	14632
	z^4	2096	3.39	16155
	$\boldsymbol{z^3}$	**1483**	**2.02**	**14632**
2	Classif^1	$I^{\leq}$=1900	$I^{\geq}$=2.35	$I^{\leq}$=9600.000
	z^5	950	6.84	11606
	z^6	1240	2.06	14939
	$\boldsymbol{z^6}$	**1240**	**2.06**	**14939**
3	Classif^1	$I^{\leq}$=1500	$I^{\geq}$=2.40	$I^{\leq}$=12000
	z^7	1348	2.14	9329
	z^8	1236	2.07	9339
	z^9	1234	1.85	9857
Pref.	z^7	**1348**	**2.14**	**9329**

TABLE 2 Training data for generating the NIMBUS classification

Input	Output
[{2222, 11.02, 16842}	{2000, 2.30, 9500}]
[{1483, 2.02, 14632}	{1900, 2.35, 9600}]
[{1240, 2.06, 14939}	{1500, 2.40, 12000}]

cation preference agent using the computer learning approach utilizing epsilon-support vector regression (see e.g. [19]) for building the DM's preference model. The preference model is built by converting the solution process shown in Table 1 to the training data for the support vector machine used by the agent. In the demonstrated approach, the solution shown to the DM in each iteration is used as the input data, and the preference information specified by the DM is used as the output data. Before the training, the NIMBUS classification information is converted to reference points (as reasoned earlier), i.e. $\bar{z}^1 = (2000, 2.3, 9500)$, $\bar{z}^2 = (1900, 2.35, 9600)$ and $\bar{z}^3 = (1500, 2.4, 12000)$. The training data for an agent corresponding to the solution process given in Table 1 is shown in Table 2.

Next, the interactive method generates new Pareto optimal solution(s) and the DM selects the most preferred from among them. In with the agent assisted algorithm the selection is done by a selecting preference agent. Here we demon-

TABLE 3 Training data for selecting the most preferred solution

Input	Output	Selection
{1732, 3.92, 12402} {1483, 2.02, 14632} {2096, 3.39, 16155}	{1483, 2.02, 14632}	
{950, 6.84, 11606} {1240, 2.06, 14939}	{1240, 2.06, 14939}	⇒ {1236, 2.07, 9339}
{1348, 2.14, 9329} {1236, 2.07, 9339} {1234, 1.85, 9857}	{1348, 2.14, 9329}	

strate an approach used in the example implementation where the selecting preference agent is implemented with an epsilon-support vector regression. In this approach, the agent is given a set of Pareto optimal solutions presented to the DM at each NIMBUS iteration as the input data, and the solution selected by the DM is used as the output data. If the DM does not select any of the solutions in an iteration, the iteration is ignored. The input and output data pairs corresponding to the solution process shown in Table 1 are given in Table 3. As the solution process involves three NIMBUS iterations, the training data consists of three input and output data pairs.

After the preference agents are trained, they can be used to replace a DM during a solution process. Naturally, if the agents are used for replicating the solution process used for training the agents, the actions are the same and therefore trained agents should be used for a modified solution process. For example, the selecting preference agent can be used to select the most preferred solution from among all solutions $z^l, l \in [1,9]l$ from Table 1. In Table 3, the selected solution for the given training data is shown on the third column. As can be seen, the agent selects solution z^8. On the other hand, if the agent would be used to select the most preferred solution among solutions of the iteration 3, i.e. $\{z^7, z^8, z^9\}$, it would select the solution z^7 selected also by the DM. Naturally, if the selecting preference agent were to include training data where the DM has selected a solution from among all of the solutions, then the agent would then select the same solution as the DM. After a solution is selected, the trained classification preference agent can be used to give new preference information. For solution z^7, a classification preference agent trained with the data given in Table 2 would specify a NIMBUS classification where $I{\leq}= 1860$, $I{\geq}= 2.34$ and $I{\geq}= 9960$.

In the example implementation used in [PV], the preference agents described here were used during the offline phase of the agent assisted algorithm as described in Section 5.2.1. That is, the approximate Pareto optimal solutions were projected onto the Pareto frontier of the original problem, and then the selecting and classification preference agents were used as a DM in an interactive solution process to find new solutions for the original problems. These solutions were

TABLE 4 Training data for predicting the reference point $\bar{z}^4$

	Input	Output	Prediction
$\bar{z}_1^4$	$\{2000, 2.3, 9500, 1900, 2.35, 9600, 2.4, 12000\}$	1500	$\Rightarrow 1435.34$
	$\{2000, 2.3, 9500, 2.35, 9600\}$	1900	
$\bar{z}_2^4$	$\{2000, 2.3, 9500, 1900, 2.35, 9600, 1500, 12000\}$	2.4	$\Rightarrow 2.20$
	$\{2000, 2.3, 9500, 1900, 9600\}$	2.35	
$\bar{z}_3^4$	$\{2000, 2.3, 9500, 2.35, 1900, 2.35, 2.4, 1500\}$	12000.0	$\Rightarrow 12299$
	$\{2000, 2.3, 9500, 2.35, 2.35, 2.4\}$	9600	

then used as the construction set for updating the surrogate problem and some of them were selected to be shown to the DM.

Both preference agents described above are considered to be using the computer learning approach for building the preference model, as they consider the solution process iteration by iteration. Next, we demonstrate how new preference information can be obtained when using the human learning approach. In the demonstrated approach, we first convert the preference information to reference points (as reasoned earlier), and then consider the preferences specified by the DM as a multivariate time series. We then generate new preference information, i.e., a reference point, by predicting what would be the next value in the series (see e.g. [18]). In the implementation of the agent assisted algorithm, the components of the reference points are joined to form $t-1$ time series for k objectives. Here t is the number of iterations. The last value of each of the time series is considered as the output data, and the other values are used as the input data. The time series are ordered in the following way:

$$\left.\begin{array}{c|c} \text{Input} & \text{Output} \\ \left[\begin{array}{ccc} \{\bar{z}_n^{[1,t-1]} & \bigcup & \bar{z}_m^t\} \\ \{\bar{z}_n^{[1,t-2]} & \bigcup & \bar{z}_m^{t-1}\} \\ & \vdots & \\ \{\bar{z}_n^1 & \bigcup & \bar{z}_m^2\} \end{array}\right. & \left.\begin{array}{c} \bar{z}_j^t \\ \bar{z}_j^{t-1} \\ \vdots \\ \bar{z}_j^2 \end{array}\right] \end{array}\right\} \Rightarrow \bar{z}_j^{t+1}, \text{ for all } n \in \{k\}, m \in \{k\} \setminus j \tag{7}$$

As a numerical example, let us consider the preference information specified by the DM in [PIV] for the two-stage separation problem. The solution process for solving the two-stage separation problem with the NIMBUS method is shown in Table 1.

The time series used for predicting a new reference point is shown in the "Input" and "Output" columns of Table 4. Here the classification information is converted to reference points, as in Table 2. The reference points are then used to form three time series following the ordering given in (7). The formed training data is then used to train a feed-forward multi-layer neural network. After training, the neural network can predict what the next reference point in

the series should be. The prediction is shown in the last column of Table 4, i.e. $\bar{z}_4 = (1435.34, 2.2, 12299)$. In Table 1, there are only three NIMBUS iterations, and therefore there are only two input and output training data pairs. With more iterations, the number of training data pairs would naturally increase.

In addition to the preference agent implementations described here, the example implementation of the agent assisted algorithm consists of preference agents using different underlying machine learning tools, method agents for the NIMBUS method, and a surrogate agent for constructing the PAINT surrogate problem. As their implementations only add technical details to the descriptions given earlier, they are not described here in detail. In addition, as commented on [PV], the example implementation of the agent assisted algorithm contains so-called optimization agents, but as they are utilized for solving the single objective subproblems of the NIMBUS method, they are beyond the scope of this thesis.

It should be noted that the problem has been previously solved in [PIV], without the use of a surrogate problem. In [PIV], the DM was able to obtain only two sets of new Pareto optimal solutions per day, and the DM was involved in the process for more than a week to obtain ten Pareto optimal solutions. When using the agent assisted algorithm, the DM obtained approximated Pareto optimal solutions immediately after specifying his preferences, and the decision phase of the solution process took less than 30 minutes. This allowed the DM to explore the problem with 19 approximated Pareto optimal solutions. The agent assisted algorithm then generated Pareto optimal solutions based on the preference model constructed during the decision phase and it selected four of the solutions to be shown to the DM. From the set of these solutions, the DM was able to find a final solution for the two-stage separation process that was more satisfactory than the final solution found in [PIV].

When using the agent assisted algorithm, the offline phases of the algorithm took more than two weeks to generate the Pareto optimal solutions. Therefore, with the agent assisted algorithm the whole solution process took significantly more time than the solution process of the initial research, but, on the other hand, the main aim of the agent assisted algorithm is to lower the time required from the DM. When compared to [PIV], where the DM was involved in the solution process for five days, the 30 minutes needed when using the agent assisted algorithm was a significant improvement. Furthermore, the agent assisted algorithm was capable of finding Pareto optimal solutions with minimal guidance from the analyst, and, in effect, the amount of effort required from the analyst was also smaller.

6 AUTHOR'S CONTRIBUTION

In article [PI], the author developed the Synheat-NIMBUS model for solving the heat exchanger network synthesis problem. The model was developed specifically to the network synthesis problem and could not be utilized generally. The author's main contribution in [PI] was in formulating the NIMBUS method's integration with the GAMS modeling system and supporting the DM in solving the network synthesis problem. This work was continued in [PII], where the author presented an approach for implementing interactive methods. In this approach, referred to as the *core structure of an interactive method*, the interactive method is divided into three distinct parts. By using the division, implementing an interactive method is a straightforward process, while simultaneously ensuring that the implementation can be applied when solving a set of multiobjective optimization problems instead of some specific problem. In addition, in this article the author gives an insight into the details related onto the implementation issues encountered while implementing an interactive multiobjective optimization method. As an example of utilizing the core structure, the author implemented and introduced the new GAMS-NIMBUS Tool integrating the NIMBUS method with the GAMS modeling system. In [PI] the author also briefly described how a multiobjective optimization problem could be formulated with the GAMS modeling language.

The multiobjective optimization problems discussed in this thesis can be either considered as computationally expensive or they can be expanded to contain more features, resulting in a computationally expensive problem. In this thesis, the solution process for computationally expensive multiobjective optimization problems was first studied in paper [PIII]. There, the author implemented the IND-NIMBUS PAINT module that combines the PAINT method for constructing a surrogate problem with the interactive NIMBUS method. In addition, the author tested the IND-NIMBUS PAINT module and participated in the solution process of the wastewater treatment problem. The author further developed the IND-NIMBUS PAINT module for the research conducted in [PV].

In article [PIV], the author tackled the issue of connecting an interactive method to an external simulator. For this purpose the author gathered the re-

quirements needed for building the connection and implemented the connection as a *coupling interface*. This coupling interface was then demonstrated by utilizing it while solving the multiobjective optimization problem of a two-stage separation process implemented in the APROS dynamic simulator software with the interactive NIMBUS method. The author then acted as the analyst during the solution process of the two-stage separation process problem.

In the author's experience, problem model validation is an integral part of the solution process. Therefore, in [PIV], the author introduced an AIMO algorithm, in order to take into account how the DM's expertise regarding the problem could be utilized when validating a model for a multiobjective optimization problem. The author proposed that as the DM explores the Pareto frontier when using an interactive method, (s)he can observe if the obtained objective function values do not correspond to his or her expectations. This could then lead to a reformulation of the problem model

In article [PV], the author developed a new approach for solving computationally expensive multiobjective optimization problems with interactive methods. To this end, the author introduced an agent-based interactive algorithm to approach the issue of surrogate problem accuracy and of how to increase that accuracy specifically in the areas that the DM is interested in. The author achieved this by utilizing several different kinds of independent agents that observe the actions taken by the DM during the interactive solution process, and based on these observations, the agents construct and update a surrogate problem for the areas that the DM is interested in. The author implemented an example of an agent-based algorithm using the previously implemented NIMBUS method and the IND-NIMBUS PAINT module to solve the two-stage separation problem presented in [PIV], providing significantly improved solutions for the problem.

7 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

The research ideas for this thesis have arisen during the author's involvement in the implementation of different optimization methods for solving several multiobjective optimization problems. The methods involved have been mainly interactive multiobjective optimization methods, and in general, the problems being solved have been computationally expensive. Therefore, this research has had two main goals. Firstly, this research describes the procedures that should be used when implementing interactive multiobjective optimization methods. Secondly, this research introduces new approach for solving computationally expensive multiobjective optimization problems with interactive methods in practice.

Even though interactive multiobjective optimization methods are evidently suitable for solving multiobjective optimization problems, only a few of their implementations are available. In this thesis, we have described a core structure for implementing scalarization-based interactive methods. With this structure we have identified the different components that are necessary to construct an implementation of an interactive method. This should facilitate easy implementation. The structure has been utilized for designing the IND-NIMBUS software framework, enabling the framework to be used for implementing several different interactive methods.

We demonstrated the use of the core structure by constructing the GAMS-NIMBUS Tool and by utilizing the tool for solving power generation and heat exchanger network synthesis problems. With the GAMS-NIMBUS Tool it is possible to solve multiobjective optimization problems modeled with the GAMS modeling language using the interactive NIMBUS method. To our knowledge, interactive methods have not previously been used with modeling languages, but by following the instructions given in [PII], implementing a scalarization based interactive method for a modeling environment should be a straightforward task.

We have also studied how the connection between a model of a multiobjective optimization problem and an implementation of an interactive method can be handled. Based on the results, we have described a general approach for building a coupling interface between the two. When using the coupling inter-

face, the problem model can be solved with different interactive methods without any changes to the model itself. We have demonstrated the coupling interface by connecting the NIMBUS method with the APROS dynamic process and utilizing the interface for solving a two-stage separation process problem in [PIV] and [PV]. The coupling interface also allows for changing the optimization method used for solving the problem model. In [PIII] we further demonstrated this by utilizing the coupling model when solving the problem of an operating wastewater treatment plant, previously solved with the NIMBUS method, with the new IND-NIMBUS PAINT module.

In our experience, it is quite common that the problem model is not complete when the interactive solution process is started, even though this is usually at least implicitly assumed. We noted that when using an interactive method, the DM is able to discover that the obtained results do not follow his or her expectations. If so, it should not be blithely assumed that this is always a part of the learning process that the DM experiences when exploring the problem. While this is possible, especially in the beginning of the interactive solution process, the option for revisiting the problem model should not be disregarded. To emphasize this, in [PIV] we have introduced an AIMO algorithm where the DM can conveniently revise the problem model. In our experience, the importance of well formulated problem models cannot be overstated, as we have encountered several problems that have been previously solved with non-interactive methods but the fact that the problem was not modeled correctly has only been brought up when an interactive method is involved.

When solving a computationally expensive problem with an interactive method, the involvement of the DM may lead the solution process to a final solution that the DM cannot be certain of as being the best compromise solution to the problem. Therefore, the computational cost of a problem model must be handled somehow. If all other approaches have proven to be unfruitful, we propose that a computationally expensive problem should be replaced with a less expensive surrogate problem. To this end, we introduced a combination of the PAINT and NIMBUS methods as the so-called IND-NIMBUS PAINT module. With the module it is possible to replace a computationally expensive problem with a less expensive surrogate problem, and thus decrease the waiting times experienced by the DM. As the IND-NIMBUS PAINT module follows the core structure, it can easily take advantage of the implementation of the NIMBUS method developed in earlier work. More importantly, due to the combination of the core structure and the coupling interface, the IND-NIMBUS PAINT module can be utilized for solving problem models without any additional work required to connect the problem with the module. We demonstrated the IND-NIBMUS PAINT module by solving the problem of operating wastewater treatment plant in [PIII].

In this thesis we note that the inaccuracy of the surrogate problem may lead the solution process to an undesired final solution. To address this issue, we utilized the NIMBUS method and the IND-NIMBUS PAINT module as components of the agent assisted algorithm to construct a new optimization tool for solving computationally expensive problems with an interactive multiobjective

optimization method. As the IND-NIMBUS PAINT module and the NIMBUS method implementations follow the core structure described in Section 4.1 as well as the coupling interface described in Section 3.3 for utilizing the problem model, the implementation of the new optimization tool was a straightforward task. By using the agents, it is possible to update the PAINT surrogate problem for the areas that the DM has shown interest in. Furthermore, the agents can provide the DM with Pareto optimal solutions for areas that the DM has not yet explored but which still could be interesting to him or her, without the DM providing additional preference information. The agent assisted algorithm was demonstrated in [PV] by solving the two-stage separation process problem.

With the research results presented in this thesis we have been able to produce clear instructions regarding how an interactive multiobjective optimization method can be implemented. We have discussed the importance of careful consideration regarding how a problem model is connected to the model, as well as the importance of model validation and how an interactive method can be utilized for the validation. We have also combined our research to introduce a new agent assisted algorithm that can be used to handle some of the issues encountered when solving computationally expensive problems with an interactive method.

During the research conducted for this thesis, we encountered several interesting topics not considered here. As mentioned in Section 3.3, the IND-NIMBUS software framework has been applied for solving several different multiobjective objective optimization problems implemented with different tools, such as with different simulator software or programming languages. One interesting approach for extending this work would be to study how different modeling tools could be combined together in order to form an aggregate problem model. For example, by studying the requirements for the coupling interface when combining objective functions for the same problem implemented with different tools, e.g. with MATLAB and the GAMS modeling language.

In Section 3.2, we suggested that during an interactive solution process the DM is able to suggest that the problem model should be revisited. In the proposed AIMO algorithm we assume that the DM provides the modeler with information on the problem aspects where (s)he feels the formulation is incorrect, but, in practice, the revision process can be successful only if the modeler receives accurate information concerning how the problem model should be changed. Therefore, step 6 of the AIMO algorithm should require the DM to provide the information in a structured way, e.g., to determine whether any of the objective functions are too sensitive, or if some of the problem constraints should be reformulated.

In addition, it might be useful to include some additional validation possibilities to the interactive solution process. For example, the AIMO algorithm could be extended to contain sensitivity analysis of the Pareto optimal solutions before showing them to the DM. This, and previous research related to the augmented algorithm could be combined with the agent assisted algorithm, so that during the initial offline phase the validation could be conducted at least partially

without feedback from the analyst.

As far as the coupling interface discussed in Section 3.3 is concerned, the DM could be provided with additional information for the problem model, i.e., information that is not directly related to the optimization process. For example, the coupling interface could offer the DM the possibility of studying the problem model with visualizations that are specific to the simulator software used.

While the agent assisted algorithm described in Chapter 5 did produce promising results for the case under consideration, it was applied only to a single multiobjective optimization problem with a single DM. Therefore, it should be applied to different multiobjective optimization problems and with different DM. Especially the preference agents used in this thesis need further study, as they were built with general machine learning based regression models. Furthermore, the preference agents are general in their nature and are suitable for any reference point based interactive methods, and are not limited to the characteristics of the NIMBUS method. By examining the information provided by classification in more detail, the preference agent could build a more accurate model of the DM's preferences.

On a more general level, we feel that utilizing computational intelligence for optimizing computationally expensive multiobjective problems is a fruitful research topic, and that it should be further explored. We do not feel that the DM should be replaced with some form of artificial intelligence; instead, we feel that computational intelligence could offer the DM assistance during the solution process. With such assistance, we could reduce the time requirements imposed on the DM as well as diminish the cognitive load associated with the use of interactive methods.

YHTEENVETO (FINNISH SUMMARY)

Optimoinnin tarkoituksena on löytää paras mahdollinen ratkaisu johonkin ongelmaan. Yleensä ongelmaan ei ole löydettävissä yksiselitteistä ratkaisua, vaan sen sijaan paras ratkaisu määräytyy tekemistämme valinnoista jotka voivat johtaa uusien valintojen tekemiseen. Esimerkiksi lyhin työmatka jalkaisin voi kulkea vilkasliikenteisen tien ylitse turvallisemman reitin kiertäessä suojatien kautta. Tai käytettäessä autoa reitti on aivan erilainen, mutta lyhyin mahdollinen reitti voi olla ruuhka-aikana hitain.

Ongelmaa ratkaistessamme joudummekin yleensä pohtimaan mikä on tavoitteemme. Haluammeko että matka on lyhyt, nopea, turvallinen tai vaikkapa miljööltään miellyttävä? Jos nämä tavoitteet ovat keskenään ristiriitaisia, esimerkiksi jos lyhin reitti on turvaton, ongelmaan pyritään löytämään kompromissiratkaisu. Tällaisia ongelmia kutsutaan monitavoiteoptimointiongelmiksi.

Tässä väitöskirjassa käytämme monitavoiteoptimointiongelmien ratkaisemiseen ns. interaktiivisia menetelmiä. Tällaisia menetelmiä käytettäessä päätöksentekijälle esitetään ratkaisu ja häntä pyydetään määrittelemään kuinka ratkaisua tulisi muuttaa jotta se olisi hänen mielestään parempi. Ratkaisua pyritään muuttamaan toivotulla tavalla ja päätöksentekijälle näytetään uusi ratkaisu arvioitavaksi. Tätä jatketaan kunnes löydetään päätöksentekijää tyydyttävä ratkaisu.

Väitöskirjassani keskityn interaktiivisiin menetelmiin liittyviin käytännön kysymyksiin, erityisesti ratkottaessa teollisuuden prosesseista nousevia optimointiongelmia. Käytännössä tällaisten ongelmien ratkaiseminen vaatii ongelman muotoilemista tietokoneella toteutettuna numeerisena mallina. Väitöskirjani alkuosassa kuvaan kuinka tällaista mallia voidaan hyödyntää käytettäessä interaktiivisia menetelmiä. Esittelen myös uuden lähestymistavan, missä päätöksentekijän osallistumista ongelman ratkaisemiseen hyödynnetään myös mallin luomisen yhteydessä. Jatkan väitöskirjaani esittelemällä interaktiivisten menetelmien perusrakenteen jonka avulla menetelmien toteuttaminen sekä liittäminen numeerisiin malleihin on mahdollisimman helppoa. Esimerkkinä tämän perusrakenteen hyödyntämistä esittelen IND-NIMBUS -monitavoiteoptimintiohjelmiston.

Numeerinen malli saattaa sisältää ajallisesti pitkäkestoisia laskentoja, joista johtuen päätöksentekijä voi joutua odottamaan uusia ratkaisuja tuntien, päivien tai jopa kuukausien ajan. Tällöin alkuperäinen numeerinen malli voidaan korvata laskennallisesti vähemmän vaativalla sijaismallilla. Tyypillisesti sijaismallilla tuotetut ratkaisut eivät kuitenkaan vastaa täysin alkuperäisen mallin ratkaisuja. Käytettäessä interaktiivisia menetelmiä tämä voi osoittautua ongelmalliseksi, sillä päätöksentekijä joutuu tällöin tekemään valintoja epätarkkoihin tietoihin perustuen. Tämän epätarkkuuden vaikutuksen vähentämiseksi esittelen uuden lähestymistavan sijaismallien käyttämiseen. Tässä lähestymistavassa koneälyllä varustetut agentit tarkkailevat päätöksentekijän tekemiä valintoja ja pyrkivät tarkentamaan sijaismallia siten että päätöksentekijälle esitetyt ratkaisut olisivat mahdollisimman tarkkoja.

Väitöskirjaani liitetyissä artikkeleissa olen osallistunut usean eri optimointiongelman ratkaisemiseen. Olen soveltanut näiden tehtävien ratkaisemisessa esittelemääni interaktiivisten menetelmien perusrakennetta. Lisäksi olen hyödyntänyt sijaismalleja päätöksentekijän kokemien odotusaikojen lyhentämiseen sekä jätevesilaitoksen toimintasuunnitelman optimoinnissa että kaksivaiheisen suodatusprosessin suunnittelussa, jossa käytin agentteja sijaistehtävän tarkentamiseen.

REFERENCES

[1] A. Abiola, E. S. Fraga, and P. Lettieri. Multi-objective design for the consequential life cycle assessment of corn ethanol production. In S. Pierucci and G. B. Ferraris, editors, *20th European Symposium on Computer Aided Process Engineering*, volume 28 of *Computer Aided Chemical Engineering*, pages 1309–1314. Elsevier, 2010.

[2] H. Ackermann, A. Newman, H. Röglin, and B. Vöcking. Decision-making based on approximate and smoothed pareto curves. *Theoretical Computer Science*, 378(3):253–270, 2007.

[3] P. J. Agrell, B. J. Lence, and A. Stam. An interactive multicriteria decision model for multipurpose reservoir management: The Shellmouth Reservoir. *Journal of Multi-Criteria Decision Analysis*, 7(2):61–86, 1998.

[4] T. Aittokoski and K. Miettinen. Cost effective simulation-based multiobjective optimization in performance of internal combustion engine. *Engineering Optimization*, 40(7):593–612, 2008.

[5] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, Inc., New Jersey, 2005.

[6] N. Asprion, S. Blagov, O. Ryll, R. Welke, A. Winterfeld, A. Dittel, M. Bortz, K.-H. Küfer, J. Burger, A. Scheithauer, and H. Hasse. Decision support for process development in the chemical industry. *Chemical Engineering Transactions*, 24:301–306, 2011.

[7] P. Auvinen, M. Mäkelä, and J. Mäkinen. Structural optimization of forest machines with hybridized nonsmooth and global methods. *Structural and Multidisciplinary Optimization*, 23(5):382–389, 2002.

[8] F. B. Aydemir, A. Günay, F. Öztoprak, Ş. İker Birbil, and P. Yolum. Multiagent cooperation for solving global optimization problems: An extendible framework with example cooperation strategies. *Journal of Global Optimization*, 57(2):499–519, 2013.

[9] S. Bechikh, L. Ben Said, and K. Ghedira. Negotiating decision makers' reference points for group preference-based evolutionary multi-objective optimization. In *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, pages 377–382, 2011.

[10] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev. Linear programming with multiple objective functions: Step method (STEM). *Mathematical Programming*, 1:366–375, 1971.

[11] J. Bisschop and R. Entriken. *AIMMS The Modeling System*. Paragon Decision Technology, 1993.

[12] M. Bortz, J. Burger, N. Asprion, S. Blagov, R. Böttcher, U. Nowak, A. Scheithauer, R. Welke, K.-H. Küfer, and H. Hasse. Multi-criteria optimization in chemical process design and decision support by navigation on pareto sets. *Computers & Chemical Engineering*, 60:354–363, 2014.

[13] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, editors. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, Berlin, Heidelberg, 2008.

[14] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS – A User's Guide*. GAMS Development Corporation, 2008.

[15] J. Buchanan. A naive approach for solving MCDM problems: The GUESS method. *Journal of the Operational Research Society*, 48(2):202–206, 1997.

[16] J. Buchanan and L. Gardiner. A comparison of two reference point methods in multiple objective mathematical programming. *European Journal of Operational Research*, 149(1):17–34, 2003.

[17] J. Cabello, M. Luque, F. Miguel, A. Ruiz, and F. Ruiz. A multiobjective interactive approach to determine the optimal electricity mix in andalucía (spain). *TOP*, 22(1):109–127, 2014.

[18] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5(6):961–970, 1992.

[19] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[20] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. North-Holland, New York, 1983.

[21] S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers*. McGraw-Hill Higher Education, New York, 2010.

[22] B. Cobacho, R. Caballero, M. GonzáÂ¡lez, and J. Molina. Planning federal public investment in mexico using multiobjective decision making. *Journal of the Operational Research Society*, 61(9):1328–1339, 2010.

[23] D. Cvetković and I. Parmee. Agent-based support within an interactive evolutionary design system. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 16(5):331–342, 2002.

[24] P. Davidsson, J. A. Persson, and J. Holmgren. On the integration of agent-based and mathematical optimization techniques. In N. T. Nguyen, A. Grzech, R. Howlett, and L. C. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, pages 1–10. Springer Berlin Heidelberg, 2007.

[25] K. Deb, K. Miettinen, and S. Chaudhuri. Towards an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, 14(6):821–841, 2010.

[26] R. Drezewski and L. Siwik. Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In L. Rutkowski, R. Tadeusiewicz, L. Zadeh, and J. Zurada, editors, *Artificial Intelligence and Soft Computing–ICAISC 2008*, pages 388–397. Springer Berlin Heidelberg, 2008.

[27] P. Eskelinen, K. Miettinen, K. Klamroth, and J. Hakanen. Pareto navigator for interactive nonlinear multiobjective optimization. *OR Spectrum*, 32(1):211–227, 2010.

[28] R. Fourer, D. M. Gay, and B. W. Kerninghan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press/Cole-Thomson Publishing Company, Pacific Grove, CA, 2003.

[29] X.-Z. Gao, T. Jokinen, X. Wang, S. J. Ovaska, and A. Arkkio. A new harmony search method in optimal wind generator design. In *2010 XIX International Conference on Electrical Machines (ICEM)*, pages 1–6, 2010.

[30] M. E. Gregori, J. P. Cámara, and G. A. Bada. A jabber-based multi-agent system platform. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1282–1284, New York, NY, USA, 2006. ACM.

[31] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–107. Kluwer Academic Publishers, 1989.

[32] J. Hakanen, J. Hakala, and J. Manninen. An integrated multiobjective design tool for process design. *Applied Thermal Engineering*, 26(13):1393–1399, 2006.

[33] J. Hakanen, Y. Kawajiri, K. Miettinen, and L. Biegler. Interactive multiobjective optimization for simulated moving bed processes. *Control and Cybernetics*, 36(2):282–320, 2007.

[34] J. Hakanen, K. Miettinen, M. M. Mäkelä, and J. Manninen. On interactive multiobjective optimization with NIMBUS in chemical process design. *Journal of Multi-Criteria Decision Analysis*, 13(2-3):125–134, 2005.

[35] J. Hakanen, K. Miettinen, and K. Sahlstedt. Wastewater treatment: new insight provided by interactive multiobjective optimization. *Decision Support Systems*, 51(2):328–337, 2011.

[36] J. Hakanen, K. Sahlstedt, and K. Miettinen. Wastewater treatment plant design and operation under multiple conflicting objective functions. *Environmental Modelling & Software*, 46(1):240–249, 2013.

[37] M. Hartikainen, K. Miettinen, and M. M. Wiecek. PAINT: Pareto front interpolation for nonlinear multiobjective optimization. *Computational Optimization and Applications*, 52:845–867, 2012.

[38] M. Hasenjäger and B. Sendhoff. Crawling along the Pareto front: Tales from the practice. In *The 2005 IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, pages 174–181, 2005.

[39] E. Heikkola, K. Miettinen, and P. Nieminen. Multiobjective optimization of an ultrasonic transducer using NIMBUS. *Ultrasonics*, 44(4):368–380, 2006.

[40] J. Hämäläinen, K. Miettinen, P. Tarvainen, and J. Toivanen. Interactive solution approach to a multiobjective optimization problem in paper machine headbox design. *Journal of Optimization Theory and Applications*, 116(2):265–281, 2003.

[41] M. P. Johnson and A. P. Hurter. Decision support for a housing mobility program using a multiobjective optimization model. *Management Science*, 46(12):1569–1584, 2000.

[42] I. Kaliszewski. Out of the mist–towards decision-maker-friendly multiple criteria decision making support. *European Journal of Operational Research*, 158(2):293–307, 2004.

[43] P. Kere, M. Lyly, and J. Koski. Using multicriterion optimization for strength design of composite laminates. *Composite Structures*, 62(3-4):329–333, 2003.

[44] T. Laukkanen, T.-M. Tveit, V. Ojalehto, K. Miettinen, and C.-J. Fogelholm. Bilevel heat exchanger network synthesis with an interactive multiobjective optimization method. *Applied Thermal Engineering*, 48(1):301–316, 2012.

[45] H. Li and H. Ding. Agent-based evolutionary algorithms applied to constrained multi-objective optimization problems. *Applied Artificial Intelligence*, 26(10):941–951, 2012.

[46] I. Lobel, A. Ozdaglar, and D. Feijer. Distributed multi-agent optimization with state-dependent communication. *Mathematical Programming*, 129(2):255–284, 2011.

[47] M. Luque, F. Ruiz, and K. Miettinen. Global formulation for interactive multiobjective optimization. *OR Spectrum*, 33:27–48, 2011.

[48] C. Mack. Fifty years of moore's law. *Semiconductor Manufacturing, IEEE Transactions on*, 24(2):202–207, 2011.

[49] E. Madetoja, K. Miettinen, and P. Tarvainen. Issues related to the computer realization of a multidisciplinary and multiobjective optimization system. *Engineering with Computers*, 22(1):33–46, 2006.

[50] E. Madetoja, E.-K. Rouhiainen, and P. Tarvainen. A decision support system for paper making based on simulation and optimization. *Engineering with Computers*, 24(2):145–153, 2008.

[51] T. Máhr, J. Srour, M. de Weerdt, and R. Zuidwijk. Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, 18(1):99–119, 2010.

[52] G. Mavrotas. Generation of efficient solutions in multiobjective mathematical programming problems using GAMS. Technical report, School of Chemical Engineering, National Technical University of Athens, 2006.

[53] G. Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, July 2009.

[54] T. Mens and T. Tourwe. A survey of software refactoring. *Software Engineering, IEEE Transactions on*, 30(2):126–139, 2004.

[55] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[56] K. Miettinen. IND-NIMBUS for demanding interactive multiobjective optimization. In T. Trzaskalik, editor, *Multiple Criteria Decision Making '05*, pages 137–150. The Karol Adamiecki University of Economics in Katowice, Katowice, 2006.

[57] K. Miettinen. Interactive multiobjective optimization method NIMBUS applied to continuous casting of steel. In N. Bandyopadhyay, P. Chattopadhyay, and S. Chattopadhyay, editors, *International Workshop on Neural Network and Genetic Algorithm in Materials Science and Engineering, Proceedings*, pages 58–72, New Delhi, 2006. Tata McGraw-Hill Publishing Company.

[58] K. Miettinen. Using interactive multiobjective optimization in continuous casting of steel. *Materials and Manufacturing Processes*, 22(5):585–593, 2007.

[59] K. Miettinen. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, 36(1):3–37, 2014.

[60] K. Miettinen and J. Hakanen. Why use interactive multi-objective optimization in chemical process design. In G. P. Rangaiah, editor, *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*, pages 153–188. World Scientific, 2008.

[61] K. Miettinen and M. M. Mäkelä. Interactive bundle-based method for nondifferentiable multiobjective optimization: NIMBUS. *Optimization*, 34:231–246, 1995.

[62] K. Miettinen and M. M. Mäkelä. Interactive multiobjective optimization system WWW-NIMBUS on the Internet. *Computers & Operations Research*, 27(7-8):709–723, 2000.

[63] K. Miettinen and M. M. Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24(2):193–213, 2002.

[64] K. Miettinen and M. M. Mäkelä. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research*, 170(3):909–922, 2006.

[65] K. Miettinen, M. M. Mäkelä, and T. Männikkö. Optimal control of continuous casting by nondifferentiable multiobjective optimization. *Computational Optimization and Applications*, 11:177–194, 1998.

[66] K. Miettinen, M. M. Mäkelä, and J. Toivanen. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization*, 27(4):427–446, 2003.

[67] K. Miettinen, F. Ruiz, and A. P. Wierzbicki. Introduction to multiobjective optimization: Interactive approaches. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 27–57. Springer-Verlag, Berlin, Heidelberg, 2008.

[68] M. Monz, K. Küfer, T. Bortfeld, and C. Thieke. Pareto navigation – algorithmic foundation of interactive multi-criteria IMRT planning. *Physics in medicine and biology*, 53(4):985, 2008.

[69] H. Nakayama, K. Kaneshige, S. Takemoto, and Y. Watada. Application of a multi-objective programming technique to construction accuracy control of cable-stayed bridges. *European Journal of Operational Research*, 87(3):731–738, 1995.

[70] H. Nakayama and Y. Sawaragi. Satisficing trade-off method for multiobjective programming. In M. Grauer and A. P. Wierzbicki, editors, *Interactive Decision Analysis*, pages 113–122. Springer-Verlag, Berlin, 1984.

[71] M. Niazi and A. Hussain. Agent-based computing from multi-agent systems to agent-based models: a visual survey. *Scientometrics*, 89(2):479–499, 2011.

[72] M. M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization Analysis and Algorithms with Applications to Optimal Control*. World Scientific, Singapore, 1992.

[73] P. Y. Papalambros and D. J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, Cambridge, 2000.

[74] M. Pilat and R. Neruda. Meta-learning and model selection in multiobjective evolutionary algorithms. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 433–438, 2012.

[75] S. Poles, M. Vassileva, and D. Sasaki. Multiobjective optimization software. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 329–348. Springer-Verlag, Berlin, Heidelberg, 2008.

[76] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez and J. Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Kluwer Academic Publishers, 1994.

[77] W. Price. Global optimization by Controlled Random Search. *Journal of Optimization Theory and Applications*, 40(3):333–348, 1983.

[78] F. Ruiz, M. Luque, and K. Miettinen. Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. *Annals of Operations Research*, 197(1):47–70, 2012.

[79] H. Ruotsalainen, E. Boman, K. Miettinen, and J. Tervo. Nonlinear interactive multiobjective optimization method for radiotherapy treatment planning with Boltzmann transport equation. *Contemporary Engineering Sciences*, 2(9):391–422, 2009.

[80] H. Ruotsalainen, K. Miettinen, and J.-E. Palmgren. Interactive multiobjective optimization for 3D HDR brachytherapy applying IND-NIMBUS. In D. Jones, M. Tamiz, and J. Ries, editors, *New Developments in Multiple Objective and Goal Programming*, pages 117–131. Springer-Verlag, Berlin, Heidelberg, 2010.

[81] H. Ruotsalainen, K. Miettinen, J.-E. Palmgren, and T. Lahtinen. Interactive multiobjective optimization for anatomy-based three-dimensional HDR brachytherapy. *Physics in Medicine and Biology*, 55(16):4703–4719, 2010.

[82] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2003.

[83] S. Ruzika and M. M. Wiecek. Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, 126(3):473–501, 2005.

[84] R. A. Sarker and T. Ray. Agent based evolutionary approach: An introduction. In R. A. Sarker and T. Ray, editors, *Agent-Based Evolutionary Search*, pages 1–11. Springer Berlin Heidelberg, 2010.

[85] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization*. Academic Press, Inc., Orlando, Florida, 1985.

[86] J. D. Siirola, S. Hauan, and A. W. Westerberg. Toward agent-based process systems engineering: Proposed framework and application to non-convex optimization. *Computers & Chemical Engineering*, 27(12):1801–1811, 2003.

[87] J. D. Siirola, S. Hauan, and A. W. Westerberg. Computing Pareto fronts using distributed agents. *Computers & Chemical Engineering*, 29(1):113–126, 2004.

[88] E. Silvennoinen, K. Juslin, M. Hänninen, O. Tiihonen, J. Kurki, and K. Porkholm. The APROS software for process simulation and model development. Technical report, VTT, Espoo, Finland, 1989.

[89] K. Sindhya, V. Ojalehto, J. Savolainen, H. Niemistö, J. Hakanen, and K. Miettinen. APROS-NIMBUS: Dynamic process simulator and interactive multiobjective optimization in plant automation. In A. Kraslawski and I. Turunen, editors, *Proceedings of the 6th International Conference on Simulation, Modelling and Optimization*, pages 871–876, 2013.

[90] K. Socha and M. Kisiel-Dorohinicki. Agent-based evolutionary multiobjective optimisation. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. (CEC '02)*, volume 1, pages 109–114, 2002.

[91] J. A. Sokolowski and C. M. Banks. *Principles of Modeling and Simulation: A Multidisciplinary Approach*. John Wiley & Sons, Inc., Hoboken, NJ, USA., 2009.

[92] A. Stam, M. Kuula, and H. Cesar. Transboundary air pollution in Europe: An interactive multicriteria tradeoff analysis. *European Journal of Operational Research*, 56(2):263–277, 1992.

[93] I. Steponavice, S. Ruuska, and K. Miettinen. A solution process for simulation-based multiobjective design optimization with an application in paper industry. *Computer-Aided Design*, 47:45–58, 2014.

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

[95] E.-G. Talbi. *Metaheuristics: From Design to Implementation*, volume 74. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2009.

[96] S. Talukdar, L. Baerentzen, A. Gove, and P. De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4(4):295–321, 1998.

[97] S. Tarkkanen, K. Miettinen, J. Hakanen, and H. Isomäki. Incremental user-interface development for interactive multiobjective optimization. *Expert Systems with Applications*, 40:3220–3232, 2013.

[98] C. Thieke, K.-H. Küfer, M. Monz, A. Scherrer, F. Alonso, U. Oelfke, P. E. Huber, J. Debus, and T. Bortfeld. A new concept for interactive radiotherapy planning with multicriteria optimization: First clinical evaluation. *Radiotherapy and Oncology*, 85(2):292–298, 2007.

[99] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[100] T. Tveit, T. Laukkanen, V. Ojalehto, K. Miettinen, and C. Fogelholm. Interactive multi-objective optimisation of configurations for an oxyfuel power plant process for CO2 capture. *Chemical Engineering Transactions*, 29:433–438, 2012.

[101] H. R. Weistroffer, C. H. Smith, and S. C. Narula. Multiple criteria decision support software. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 989–1009. Springer Science + Business Media, Inc, New York, 2005.

[102] A. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3:391–405, 1982.

[103] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Inc., 2002.

[104] L. Xu, T. Reinikainen, W. Ren, B. P. Wang, Z. Han, and D. Agonafer. A simulation-based multi-objective design optimization of electronic packages under thermal cycling and bending. *Microelectronics Reliability*, 44(12):1977–1983, 2004.

[105] T. Yee and I. Grossmann. Simultaneous optimization models for heat integration. *Computers & Chemical Engineering*, 14(10):1165–1184, 1990.

[106] Y. Yun, M. Yoon, and H. Nakayama. Multi-objective optimization based on meta-modeling by using support vector regression. *Optimization and Engineering*, 10(2):167–181, 2009.

ORIGINAL PAPERS

PI

AN INTERACTIVE MULTI-OBJECTIVE APPROACH TO HEAT EXCHANGER NETWORK SYNTHESIS

by

Timo Laukkanen · Tor-Martin Tveit · Vesa Ojalehto · Kaisa Miettinen · Carl-Johan Fogelholm 2010

Computers & Chemical Engineering, 34(6)

Reproduced with kind permission of Elsevier.

PII

IMPLEMENTATION ASPECTS OF INTERACTIVE MULTIOBJECTIVE OPTIMIZATION FOR MODELING ENVIRONMENTS: THE CASE OF GAMS-NIMBUS

by

Vesa Ojalehto · Kaisa Miettinen · Timo Laukkanen 2014

Computational Optimization and Applications, 58(3)

Reproduced with kind permission of Springer.

PIII

APPLYING APPROXIMATION METHOD PAINT AND INTERACTIVE METHOD NIMBUS TO MULTIOBJECTIVE OPTIMIZATION OF OPERATING A WASTEWATER TREATMENT PLANT

by

Markus Hartikainen · Vesa Ojalehto · Kristian Sahlstedt

Engineering Optimization, to appear, DOI:10.1080/0305215X.2014.892593

Reproduced with kind permission of Taylor & Francis.

PIV

COUPLING DYNAMIC SIMULATION AND INTERACTIVE MULTIOBJECTIVE OPTIMIZATION FOR COMPLEX PROBLEMS: AN APROS-NIMBUS CASE STUDY

by

Karthik Sindhya · Vesa Ojalehto · Jouni Savolainen · Kaisa Miettinen · Hannu Niemistö 2014

Expert Systems with Applications, 41(5)

Reproduced with kind permission of Elsevier.

PV

AGENT-BASED INTERACTIVE APPROACH FOR COMPUTATIONALLY DEMANDING MULTIOBJECTIVE OPTIMIZATION PROBLEMS

by

Vesa Ojalehto · Dmitry Podkopaev · Kaisa Miettinen 2014

Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing, No. B 6/2014, University of Jyväskylä, Jyväskylä

Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing
No. B6/2014

Agent-based Interactive Approach for Computationally Demanding Multiobjective Optimization Problems

Vesa Ojalehto Dmitry Podkopaev
Kaisa Miettinen

University of Jyväskylä
Department of Mathematical Information Technology
P.O. Box 35 (Agora)
FI–40014 University of Jyväskylä
FINLAND
fax +358 14 260 2771
http://www.mit.jyu.fi/

Copyright © 2014
Vesa Ojalehto and Dmitry Podkopaev and Kaisa Miettinen
and University of Jyväskylä

ISBN 978-951-39-5814-5
ISSN 1456-436X

Agent-based Interactive Approach for Computationally Demanding Multiobjective Optimization Problems

Vesa Ojalehto

Dmitry Podkopaev

Kaisa Miettinen

Abstract

We generalize the applicability of interactive methods for solving computationally demanding, that is, time-consuming, multiobjective optimization problems. For this purpose we propose a new agent assisted interactive algorithm. It employs a computationally inexpensive surrogate problem and four different agents that intelligently update the surrogate based on the preferences specified by a decision maker. In this way, we decrease the waiting times imposed on the decision maker during the interactive solution process and at the same time decrease the amount of preference information expected from the decision maker.

The agent assisted algorithm is not specific to any interactive method or surrogate problem. As an example we implement our algorithm for the interactive NIMBUS method and the PAINT method for constructing the surrogate. This implementation was applied to support a real decision maker in solving a two-stage separation problem.

Keywords: Multiple objective programming, interactive methods, agent-based optimization, surrogate problem, NIMBUS, PAINT

1 Introduction

In the modern society, it has become more and more important to support decision makers in finding solutions which take several conflicting objectives into account and optimize the objectives simultaneously. For such problems, it is not possible to find a single optimal solution because of the conflicting nature of the objectives. Instead of a single optimal solution, these multiobjective optimization problems have

several so-called Pareto optimal solutions with different trade-offs between the objectives.

When dealing with real-world optimization problems, it is usually needed to find a single or few Pareto optimal solutions to be implemented which are called *most preferred solutions*. In order to select such a solution(s), some additional information is needed, such as how a solution should be changed in order it to get a more preferred solution for the problem, what kind of trade-offs are acceptable or what are desirable values for objective functions. This *preference information* can be obtained from a human decision maker (DM) having expertise in the problem domain. Several methods have been developed for finding the most preferable solution (see, e.g., [5, 25] and references therein).

In this paper, we concentrate on so-called *interactive* methods (see, e.g., [25, 31] and references therein), where the solution process makes progress iteratively by asking the DM to specify preference information until most preferred one is found. By exploring Pareto optimal solutions in this manner, the DM can learn about the trade-offs between the conflicting objective functions and, thus, gain insight about the problem. In addition, the DM can learn about how feasible his or her preferences are by comparing the expectations to the Pareto optimal solutions found. This means that the DM can even change his or her preferences during the solution process, if desired. Based on the learning the DM is able to make informed decisions on what kind of Pareto optimal solutions would best satisfy his or her preferences.

Interactive methods have given promising results for solving real-world optimization problems involving wide variety of engineering fields. These problem include optimal control of a continuous casting of steel [26, 27], intensity modulated radiotherapy treatment planning [37], optimizing configurations of an oxyfuel power plant process [53], operating wastewater treatment plant [17, 15], optimal design and control of a paper mill [49], among others. For more examples of use of interactive methods in various fields see [33] and references therein.

Real-world multiobjective optimization problems can be computationally demanding. The function evaluations may depend, for example, on time-consuming computations or simulations [15, 18, 49, 58]. If this is the case, an interactive multiobjective optimization process as outlaid above may become infeasible by the long waiting times needed to generate new Pareto optimal solutions according to the preference information specified by the DM. In other words, the interactive nature of the solution process suffers and the most preferred solutions may not be found. For example, the DM may be restricted to examining only very few Pareto optimal solutions and may stop the solution process prematurely.

One approach to solving computationally demanding problems is to replace computationally expensive functions by simplified ones. However, if the problem is simulation-based, that is, involves a simulator, it can be a so called black-box problem without any additional information about the problem besides decision variable and objective (and possibly constraint) function values. Another widely used approach is to utilize parallelization techniques to decrease the computation time. But it is possible that the problem is implemented in a way that does not allow for

parallelization, e.g., the used simulator may have only a limited number of licenses available.

To summarize, when solving a computationally demanding multiobjective optimization problem using an interactive method, it is quite possible that the method requires more time to generate new Pareto optimal solutions than there is to spare. If other approaches cannot be utilized or they do not provide enough improvement in the time available, a natural way of handling such problems is to replace the computationally demanding problem with a computationally less demanding surrogate. In practice, this means that the DM is shown approximate rather than Pareto optimal solutions during the interactive solution process. However, applying the surrogate problem in multiobjective optimization has significant limitations and has been elaborated only in few studies (see e.g. [11]).

A good accuracy of the surrogate problem is important in order to avoid misleading the DM. Because the preference information specified by the DM indicates what kind of solutions he or she is interested in, this information can be used to update the surrogate in an intelligent way. This means that the accuracy of the surrogate varies and is most accurate near the interesting solutions.

It has been reported in the literature that solution processes with interactive methods often take quite few iterations (see e.g. [12, 25, pp. 134–135]). One reason for this may be the cognitive load set on the DM. The load could be decreased if the amount of the preference information expected from the DM was smaller.

In this paper, we combine an interactive multiobjective objective method and a surrogate problem in an intelligent way to support the DM in order to decrease the waiting times experienced by the DM and in addition to increase the accuracy of the surrogate problem. We propose to enhance the solution process with agents, i.e., entities that try to achieve some pre-defined goals by autonomous and intelligent actions. In the proposed algorithm, we utilize the agents to update the surrogate problem near solutions that are interesting to the DM, to minimize waiting times imposed on the DM and to decrease the amount of preference information expected from the DM. We describe the proposed method as a general algorithm, as it does not depend on any specific methods or techniques. In addition to the interactive method and to the surrogate problem construction technique, the introduced agent assisted algorithm employs four different types of agents, each having their own goals.

To give more concrete ideas of how to implement agents, we demonstrate the agent assisted algorithm implemented with the classification-based NIMBUS method [25, 28, 29] selected as the interactive method and the PAINT method [16] selected as the surrogate problem construction technique. Furthermore, we apply the agent assisted algorithm involving the two above-mentioned methods to solve a computationally demanding two-stage separation problem and discuss the advantages achieved.

The rest of this paper is organized as follows. In Section 2, we present the concepts and background material utilized. This includes the interactive NIMBUS method and the PAINT surrogate construction technique that are used as examples.

In addition, we include an brief overview of agent studies in relation to this research. We introduce the new agent assisted interactive algorithm in Section 3. In Section 4, we describe the four different agents employed by the algorithm in more detail. We demonstrate the advantages of the new algorithm by giving an example of supporting a DM in solving a multiobjective two-stage separation problem in Section 5. Finally, the paper is concluded by a discussion and concluding remarks in Sections 6 and 7, respectively.

2 Background

Next we discuss the background material used in this paper. First we briefly describe the notations used and then provide information on the methods used, that is, on the interactive NIMBUS method for multiobjective optimization and the PAINT method for constructing the surrogate problem. We finish this section by defining agents in relation to our research.

2.1 Interactive Multiobjective Optimization

In this paper, we consider multiobjective optimization problems of the form

$$\begin{array}{rl} \text{minimize or maximize} & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{subject to} & \mathbf{x} \in S, \end{array} \tag{1}$$

where $f_i : S \to R$ are k (≥ 2) conflicting objective functions, and $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the *decision (variable)* vector bounded by constraints that form the feasible set $S \subset \mathbb{R}^n$. Objective vectors $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$ consist of *objective function* values calculated at $\mathbf{x}$.

A decision vector $\hat{\mathbf{x}}$ and the corresponding objective vector $\mathbf{f}(\hat{\mathbf{x}})$ are called Pareto optimal if there does not exist any other feasible $\mathbf{x}$ so that $f_i(\mathbf{x}) \leq f_i(\hat{\mathbf{x}})$ for all $i = 1, ..., k$ and $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$ for least one $j = 1, ..., k$. Such objective vectors are called Pareto optimal solutions to problem (1), and a set of Pareto optimal solutions is called a *Pareto frontier* [25]. Finding the most preferred Pareto optimal solution to problem 1 is called a *solution process*. For the solution process discussed in this research, the most preferred Pareto optimal solution is found by utilizing the *DM's preferences*, i.e. information about how a solution should be changed in order to get a more preferred solution for the problem, what kind of trade-offs between objectives are acceptable for the DM or what are desirable values for objective functions.

The ranges of objective function values in the set of Pareto optimal solutions can be shown to the DM to give general understanding about attainable solutions. The k-dimensional *ideal objective vector* contains the best values of objective values whereas the worst objective function values form a *nadir objective vector*. Components of the ideal objective vector are obtained by minimizing each of the objective functions individually subject to S whereas calculating the nadir objective vector

necessitates knowing the whole set of Pareto optimal solutions and thus, usually estimated values are used (for further information, see e.g. [3, 21, 25]).

Interactive methods typically convert the original problem with the preference information specified by the DM into single objective *subproblems* [25, 31]. By selecting the subproblems well and solving them with appropriate single objective optimization methods we get Pareto optimal solutions reflecting the preferences.

The agent assisted algorithm proposed can be used with different interactive methods following the general *core structure* of interactive multiobjective optimization methods [33]. The core structure can be described as follows:

1. Initialize the process, e.g., calculate ideal and nadir objective vectors.
2. By solving a method-specific subproblem generate an initial Pareto optimal solution to be used as a starting solution.
3. Ask the DM to specify preference information related to the starting solution (in the method-specific way).
4. Generate new solution(s) based on the preference information by solving appropriate subproblem(s).
5. Ask the DM to select the most preferred solution of the previously generated solutions and denote it as the new starting solution.
6. If the selected solution is satisfactory, stop. Otherwise continue from step 3.

It should be noted that in addition to the interactive approach described here and utilized in this research, there exists several other approaches for solving multiobjective optimization problems. When classifying different approaches by the role of the DM, in addition to the *interactive methods* where the the DM's preferences are specified in an iterative process, there exist three other classes of methods [25]. If the DM's preferences are not taken available, the method is referred to as a *no preference* method. When using an *a priori* method, the DM's preferences are asked before starting the solution process. An *a posteriori* method generates a representative set of Pareto optimal solutions, that is shown to the DM. As justified in the introduction, in this research, we consider only interactive methods.

2.2 The Interactive NIMBUS Method

In this research we use the NIMBUS method [25, 28, 29] as the interactive method. The NIMBUS method is based on the classification of the objective functions. At each iteration, the DM considers the objective function values of a starting Pareto optimal solution $\mathbf{x}^c$, and is asked to classify objective functions into up to five different classes. The classes indicate what kind of changes in the objective function values would provide a more satisfactory solution than $\mathbf{x}^c$.

For simplicity, we present the classes for functions to be minimized. The classes are for functions f_i whose values

should be improved ($i \in I^{<}$),

should be improved to some aspiration level $\hat{z}_i < f_i(\mathbf{x^c})$ ($i \in I^{\leq}$),

are satisfactory at the moment ($i \in I^{=}$),

are allowed to impair up till some bound $\epsilon_i > f_i(\mathbf{x^c})$ ($i \in I^{\geq}$),

are allowed to change freely ($i \in I^{\diamond}$).

Based on the classification information, up to four single objective subproblems are formed. By solving these subproblems we obtain four new Pareto optimal solutions, each following the classification in a slightly different way. These solutions are shown to the DM, and he or she can select one of them or one of the previously generated Pareto optimal solutions as the most preferred solution or as a starting solution of a new classification. For a more detailed description of the NIMBUS method, see [28, 29, 33].

2.3 The PAINT Surrogate Method

In this research, by a *surrogate problem* we refer to a problem that can be used to replace the original, usually computationally expensive problem for the duration of the interactive solution process. The surrogate problem is constructed in such a way that it can be solved significantly faster than the original problem while producing optimal solutions that approximate the solutions of the original problem. Using a surrogate problem eliminates the issue of DM's waiting time during the interactive solution process but, on the other hand, poses new challenges such as controlling the accuracy.

In this research, we use the PAINT method [16] to construct a surrogate problem of a computationally demanding multiobjective optimization problem. In the PAINT method, the surrogate problem is constructed based on a pre-computed set of Pareto optimal solutions. Here we refer to this set as a *constructing set*.

The constructing set can be generated with any multiobjective optimization method that generates many Pareto optimal solutions (see e.g. [40, 25, 5]). We utilize PAINT as it is applicable in both convex and nonconvex problems. More details of the PAINT method can be found in [16].

2.4 Multiagent Systems

There does not exist a single, universally agreed definition of an agent, as their usage varies from field to field. But on a general level, an *agent* is some entity, located in an some environment, where the agent tries to reach some pre-defined goal by automatic and intelligent actions [38]. Furthermore, the environment typically contains several agents interacting with each other [57]. Such an environment is called a *multiagent system*.

Agent-based computational intelligence technologies have been widely studied (see e.g. [38, 57]) and applied in many areas of science dealing with complex systems. Agent-based technologies were initially applied in information and communication, but later they have been applied in different fields related to engineering and manufacturing, such as production planning and resource allocation [42]. In addition, they have been used for single (see e.g. [2, 23, 39, 43]) and multiobjective optimization but, to our knowledge, they have not been applied in interactive multiobjective optimization discussed in this research. In [44], agents are utilized for generating Pareto optimal solutions by solving optimization problems that are similar to the subproblems used in interactive methods, but there the DM's preference information is not taken to account. In [10, 14, 22, 45, 47, 48], agents are utilized in enhancing existing evolutionary multiobjective optimization methods, where the purpose is to find a representative set of Pareto optimal solutions, that is, in an a posteriori fashion. In [4], multiple agents are utilized for supporting several DMs when solving a multiobjective optimization problem with preference-based evolutionary method. In the proposed method, agents negotiate a single reference point that should best correspond to the reference points provided by all DMs. Furthermore, in [8] agents are utilized to reduce the number of questions asked from the DM when utilizing a priori method.

In these approaches, unlike in the approach discussed in this research, the DM does not interact with the solution process in order to learn about the problem characteristics or to modify his or her preferences. Furthermore, the previous research mostly concentrates on producing either all or a representative set of Pareto optimal solutions, without discussion on how to select the most preferred Pareto optimal solution that can be the basis for practical implementation of the product or process being designed.

In our research agents directly use preference information and actively assist the DM in the interactive process of finding the most preferred solution. We define an agent to have following properties:

Emergent: agents are able to solve complex problems with a set of simple rules.

Autonomous: agents have control of their inner state and they can take actions without human intervention.

Reactive: agents take actions based on their environment.

Goal-oriented: agents aim at achieving some goal with their actions.

Communal: agents are able to communicate with other agents, be they human or artificial.

Fault tolerant: agents can attempt to recover from a failure, e.g. a failure in reaching their goal.

In the literature, it has been noted that by using multiple autonomous agents that utilize several different methods it is possible to obtain optimal solutions for complex optimization problems more efficiently in comparison to using only a single

agent (see e.g. [9, 24, 51]). This effect is usually demonstrated with empirical studies, but it has been shown that the use of multiagent systems should not adversely affect convergence properties of the optimization methods [52]. Therefore we use four different agents in our algorithm.

After having defined the main concepts to be used and introduced necessary background material, in the next section we can introduce the new agent assisted algorithm.

3 Agent Assisted Interactive Multiobjective Optimization Algorithm

The aim of this research is to provide the DM with assistance when solving a computationally demanding multiobjective optimization problem with an interactive method. As mentioned earlier, interactive methods are iterative, involving the DM in each iteration. In this section we introduce an agent assisted interactive multiobjective optimization algorithm for this purpose.

3.1 Introduction to the Agent Assisted Algorithm

An interactive method shows new Pareto optimal solution(s) to the DM, who studies it/them and then specifies information on his or her preferences. Then the DM is shown new solution(s). With this iterative procedure the DM can learn about the characteristics of the problem to form a firm idea of the Pareto optimal solutions that can be attained, and which of these solutions best match with his or her preferences. At the same time the DM can adjust one's preferences. As motivated in the introduction, waiting times can become an issue with computationally demanding problems. In this section we present the background for the new algorithm which can provide new Pareto optimal solutions without waiting times.

One approach for providing the DM with new solutions in a timely manner is to replace the computationally demanding problem with a surrogate problem, as described in Section 2.3. This approach consists of the following three phases.

1. *Construction phase.* The surrogate problem is constructed.
2. *Decision phase.* The interactive method is employed to solve the surrogate problem in communication with the DM.
3. *Projection phase.* The solution of the original problem is obtained based on the solution of the surrogate problem. If needed, the surrogate problem is updated in order to improve its accuracy and the second phase is repeated.

We distinguish the decision phase where the DM is actively involved from construction and projection phases where his or her presence is not necessary. The latter two phases can be referred to as *offline phases*. By replacing the original problem

with a surrogate problem we shift the computational burden from decision phase to offline phases, thus eliminating waiting time of the DM. On the other hand, this replacement means that, instead of Pareto optimal solutions the DM is shown approximate Pareto optimal solutions, i.e., Pareto optimal solutions of the surrogate problem. After the DM has found his or her most preferred approximate Pareto optimal solution, as the name of the projection phase suggests, the solution is projected to the Pareto frontier of the original problem. This can be done using, for example, an achievement scalarization function [55] as described in [33].

The challenge with a surrogate based approach is that the approximate Pareto optimal solution may be too far from the Pareto frontier of the original problem, i.e., the surrogate is not accurate enough. If the problem is computationally very demanding, the projection can take a long time. If the projected solution is too different from the corresponding approximate Pareto optimal solution, the DM may need to start the interactive solution process again. In the worst case, this may mean that first a new surrogate problem must be constructed before the interactive solution process can be started again and all previous preference information may be wasted. This outcome is the complete opposite to the aim of using the surrogate problem because it hinders the learning process rather than supports it. Therefore, the accuracy of the surrogate problem plays an important role.

The aim of utilizing multiple independent agents, i.e., artificial decision makers, is to improve the accuracy of the surrogate problem in the intelligent way, i.e. in those areas of the problem where the improvement is most needed. We propose to utilize four different types of agents which perform specific tasks during different phases of the solution process as described in the next section.

3.2 The Agent Assisted Algorithm

Now we are in a position to describe the proposed agent assisted interactive algorithm, to be called an *agent assisted algorithm*. This algorithm is general and not tailored for any specific interactive method or surrogate problem. The agent assisted algorithm extends the interactive method by emphasizing the intelligent updating of the surrogate problem, minimizing waiting times imposed on the DM and decreasing the amount of preference information expected from the DM, thus decreasing the cognitive load.

The overall structure of elements comprising the agent assisted algorithm can be seen in Figure 1. There are four types of agents which both perform their own tasks and communicate and share information with each other. *Preference agents* use preference information expressed by the DM to build a preference model, *interactive method agents* collect information about the parameters that the interactive method uses to generate approximate Pareto optimal solutions, based on this information *optimization agents* generate new Pareto optimal solutions, and finally, *surrogate agents* are responsible for constructing and updating the surrogate problem. Each type of agent is described in detail in the next section.

The agent assisted algorithm consists of the following six steps. We indicate for

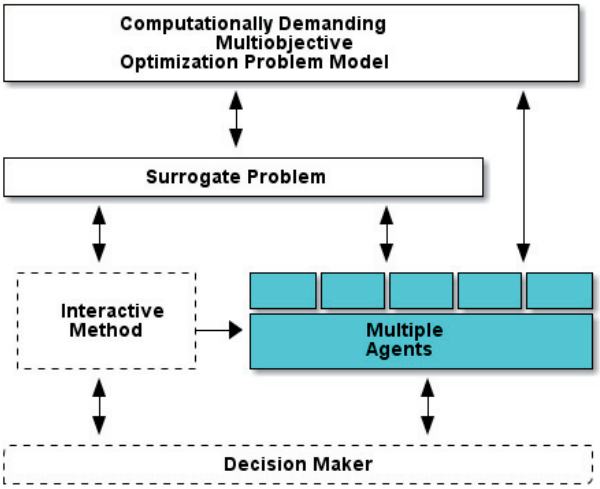


Figure 1: Overall structure of the agent assisted algorithm

each step, to which of the phases it belongs: construction phase (c.p.), decision phase (d.p.) or projection phase (p.p.).

1. (c.p.) The surrogate agent constructs the surrogate problem
   - using information from all other agents, if available.
2. (d.p.) The DM uses the interactive method and specifies preference information based on the starting solution.
   - Preference agents collect the preference information to build a model of the DM's preferences.
3. (d.p.) The interactive method generates approximate Pareto optimal solution(s) to be shown to the DM.
   - The interactive method agents collect information on how approximate Pareto optimal solutions are generated.
4. (d.p.) The DM selects one approximate Pareto optimal solution
   (a) as the new starting solution for the next iteration and continues with step 2, or
   (b) as the most preferred solution of the surrogate problem.

- Preference agents collect this information which is interpreted as preference of one solution over others.

5. (p.p.) The optimization agents generate new Pareto optimal solutions of the original problem based on the information collected by the interactive method agents and the preference agents.
6. The preference agents select a subset of Pareto optimal solutions which is shown to the DM. The DM either
   (a) continues with step 1, or
   (b) selects one as the most preferred solution of the original problem and stops.

The advantage of having separate offline and decision phases is that there are no waiting times for the DM to see approximate Pareto optimal solutions corresponding to his or her preferences. On the other hand, how long the offline phase can take is agreed with the DM. It should be noted that it is possible to choose one of the solutions in step 6 as the solution for the original problem using preference agents without involvement of the DM. Therefore, in the extreme case the DM's involvement can be restricted to steps 2 to 4.

In practice, the information from all other agents in step 1 advises where the surrogate should be updated. This means that no previously specified preference information is wasted when the DM decides to continue with step 1 from step 6.

The presented description of the algorithm is very general for it can incorporate a large variety of interactive methods and surrogate problem construction techniques. To be more specific, in the next section we select both the method and the technique which allows us to describe what agents do in detail.

4 Agents in Detail

In this section we give more information about the four types of agents utilized in the agent assisted algorithm. Figure 2 provides a more detailed view of the roles of the agents in the algorithm. Because agents depend on the interactive method and the surrogate problem selected, here we provide more information assuming that NIMBUS is the interactive method and PAINT is the method to construct the surrogate problem.

In general, each agent type can be implemented in various ways, and in practice it is advisable to utilize several different agents to compliment each other. For this reason, we refer to several agents in what follows.

4.1 Preference Agents

The main function of the preference agents is to build models of the DM's preferences. These models are built in order to identify those areas of the Pareto frontier

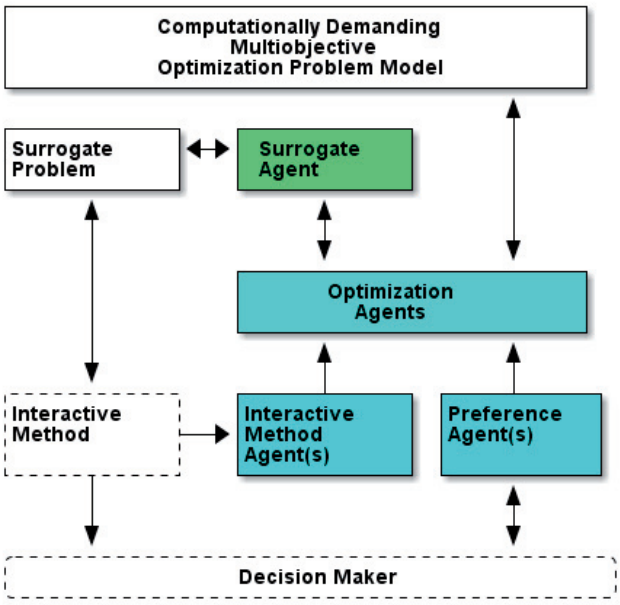


Figure 2: Detailed illustration of the agent assisted algorithm

which are interesting to the DM. The model of the DM's preferences (*preference model* for short) is usually a mathematical description of all necessary information allowing one to choose a solution or to specify preference information on the DM's behalf. A preference model is generally defined as a universal rule of selecting a subset of solutions from any given set of feasible solutions, i.e., as a *choice function* [54]. Another general, but a simpler way of modeling DM's preferences is defining a *binary relation* on the set of feasible solutions (see e.g. [1, 41]) describing the DM's preference judgments for certain pairs of solutions. These two concepts are mostly used for theoretical studies. In practical multiobjective optimization methods, people use more compact and problem-specific models of DM's preferences such as value functions (see e. g. [19, 13]) and reference points (see e.g. [56, 20]).

The process of constructing a preference model based on observations of the DM's behavior is called *preference learning*. For illustrating the agent assisted algorithm, we have implemented simple techniques based on two basic approaches: *computer learning* and *human learning*. The computer learning approach is based on the assumption that all DM's input reflects his or her steady preference model and, thus, a computer learning approach can be applied for building a DM's preference model from this data. The human learning approach assumes that as the interactive method progresses, the DM learns and adjusts one's preferences. Thus constructing the preference model is reduced to predicting its parameters for the next iteration

based on the time series of parameters in previous iterations. It should be noted that when using interactive methods, it cannot be assumed that the DM could take hundreds of iterations, and the selected preference model construction approach cannot depend on large amounts of input data.

As examples, we demonstrate the computer learning approach in the context of NIMBUS in two ways. Both of them are implemented using two different machine learning techniques: polynomial-based kriging (e.g., [36]) and support vector machines (e.g., [7]). In the first way, when training the agent, it is given as the input a set of (approximate) Pareto optimal solutions presented to the DM at each NIMBUS iteration, and as the output, the agent is given the solution that the DM selected from that set. After the computer learning agent has been trained with this data, it can be used to select one solution of a set of solutions (that would be most preferred by the DM). In what follows, this is referred to as a selecting agent. In the second computer learning way, the agent is given a Pareto optimal solution as the input and the NIMBUS classification specified by the DM in relation to that solution as the output. After the training, the agent will give as the output a classification information corresponding to any solution given as the input. This is called a classification agent. By combining the selecting and the classification agents, we can replace the DM in the offline phase of the agent assisted algorithm.

As an example of the human learning approach, an agent can be created for predicting coefficient of each objective function f_i in the achievement scalarizing function. Then for each agent, a feedforward multilayer neural network is trained. As the input, the agent is given the reference points (obtained from the classifications as per [29]) provided by the DM during the previous NIMBUS iterations. It is also given as the input the component of the reference point corresponding to the objective function f_i for the next classification. After the training, for each f_i, the agent can be given a reference point as the input, and it will give as the output the ith component of a new reference point. In this way, the history of the preference information specified during the NIMBUS iterations is utilized. Here preference learning can be understood as predicting how DM's input changes by analyzing the time series of previous input [6].

Besides step 2., step 6. of the agent assisted algorithm utilizes preference agents when selecting which Pareto optimal solutions should be shown to the DM. This is done by giving all Pareto optimal solutions generated as the input to the selecting agent.

4.2 Interactive Method Agents

The main function of the interactive method agents is to find Pareto optimal solutions during the offline phase. These solutions should correspond (as described below) to the approximate Pareto optimal solutions the DM has found during the decision phase. This can be achieved by first collecting information on how approximate Pareto optimal solutions were generated in the decision phase. Then, during the offline phase, this information is used to mimic the actions of the DM with the

original problem to generate Pareto optimal solutions which correspond to the DM's preferences. In other words, the classifications made by the DM during the NIMBUS iterations are repeated with the original problem.

In addition, the interactive method agents can also be used for projecting approximate Pareto optimal solutions obtained in the decision phase to the Pareto frontier. Using interactive method agents in the two described ways may generate two different Pareto optimal solutions per each approximate Pareto optimal solution. Which of them is shown to the DM depends on the preference agent. This increases understanding of attainable Pareto optimal solutions but, on the other hand, also increases the computational cost. If desired, the projection can be skipped.

4.3 Optimization Agents

The main function of the optimization agent is to generate Pareto optimal solutions for the original multiobjective optimization problem. When using the NIMBUS method, Pareto optimal solutions are generated by solving single objective subproblems. In the agent assisted algorithm, these subproblems are solved by the optimization agents with several different single objective optimization methods. The methods used for solving the subproblems depend on the used interactive methods. For solving the NIMBUS subproblems, we use global methods, such as differential evolution [50], controlled random search [35] and genetic algorithm [30], and local methods such as COBYLA [34] and proximal bundle method (if gradient information is available) [32] and their hybrids.

Let us briefly describe our approach to implementing optimization agents. When employed in the agent assisted algorithm, an optimization agent usually belongs to an *agent group*. For example, a new agent group is assembled for each step 2. taken by the DM. The goal of an agent in a group is to find a Pareto optimal solution corresponding to the preference information specified by the DM, i.e., each agent in the group tries to solve the same single objective NIMBUS subproblem. The agents in the same group differ by which single objective optimization method they employ, and by what parameters are given to those methods. In step 5, all agents are run simultaneously, but the rate of convergence for each agent in a group is studied on a decreasing interval and the agents converging fastest are given more computing time. To improve their convergence rate, the optimization agents are able to change their configuration, i.e. what method they are using and what are the method parameters. In addition, the optimization agents communicate with each other, providing information about the best solutions found and about the configurations the solutions have been found. This information is communicated also to the agents in other groups.

Optimization agents continue solving the NIMBUS subproblems until they cannot find configurations which provide improvement on the optimal values. In addition, step 5 is given a maximum time available for obtaining new Pareto optimal solution, and in practice, the optimization agents continue until the given time runs out.

As single objective optimization is not in the scope of this paper, optimization agents are not discussed here in more detail. They have been implemented following the results of [43].

4.4 Surrogate Agent

The main function of the surrogate agent is update the surrogate problem on those areas that DM has shown interest in. In the case of PAINT, it is intuitively obvious that the accuracy of the surrogate problem depends on the coverage of the constructing set. Therefore, adding a Pareto optimal solution to the set usually improves the accuracy in the approximate Pareto optimal solutions that can be obtained near that solution. Improving the accuracy can be achieved by including Pareto optimal solutions generated by optimization agents in the constructing set whenever appropriate.

If a preference agent has an inaccurate preference model, it can instruct optimization agents to generate a solution that does not correspond to the DM's preferences. Even if a surrogate agent adds it in the constructing set, the accuracy of the surrogate does not suffer because the solution added is Pareto optimal. The worst consequence of this is increased computational cost.

5 Case Study: Two-Stage Separation Problem

To demonstrate the benefits of the agent assisted algorithm we apply it in a two-stage separation problem, originally considered in [46]. The related multiobjective optimization problem is computationally demanding. The solution process was carried out with the implementations of agents, NIMBUS and PAINT contained in the IND-NIMBUS software framework where the DM used a graphical user interface. For further implementation details of the two methods, see [33, 17], respectively.

In the two-stage separation problem, an incoming feed of water and general impurity are separated into permeate and retentate. The process model considered here consists of two pumps pumping the feed to two filters and two pumps recycling a part of the permeate back to the filters. The goal of the two-stage separation problem is to extract a maximum amount of *retentate* (kg) from an incoming feed while minimizing the amount of *impurity* (kg) in the permeate and minimizing the *energy* (Kj) used by the pumps. The two-stage separation process is studied for the duration of a typical factory shift length, discretized over a time horizon. In addition to objective functions, the problem model consists of a single constraint and 100 decision variables of which four are continuous and 96 are binary valued. The problem is nonconvex, i.e., it contains multiple local optimal solutions. When using Intel® Core™ i7-2600 running at 3.4GHz, a single simulation of the problem model took 5 seconds on an average. In order to reliably obtain optimal solutions when using a differential evolution [50] method, an average of 15000 simulations were required. More details of the two-stage separation problem can be found in [46].

Iter	Issue	Max Permeate (kg)	Min Impurity	Min Energy (kJ)
	$\boldsymbol{z_1}$	**2222**	**11.02**	**16842**
1	Classif	$I^{\geq=2000}$	$I^{\leq=2.300}$	$I^{\leq=9500.000}$
	z_2^1	1732	3.92	12402
	z_3^1	1483	2.02	14632
	z_4^1	2096	3.39	16155
	$\boldsymbol{z_3^1}$	**1483**	**2.02**	**14632**
2	Classif	$I^{\leq=1900}$	$I^{\geq=2.35}$	$I^{\leq=9600.000}$
	z_5^1	950	6.84	11606
	z_6^1	1240	2.06	14939
	$\boldsymbol{z_6^1}$	**1240**	**2.06**	**14939**
3	Classif	$I^{\leq=1500}$	$I^{\geq=2.4}$	$I^{\leq=12000.000}$
	z_7^1	1348	2.14	9329
	z_8^1	1236	2.07	9339
	z_9^1	1234	1.85	9857
Pref.	$\boldsymbol{z_7^1}$	**1348**	**2.14**	**9329**

Table 1: Solution process of the two-stage separation problem in [46]

Originally in [46], due to time constraints, the two-stage separation problem was solved with a restricted number of function evaluations. The preferences specified by the DM and the corresponding Pareto optimal solutions generated with the interactive NIMBUS method can be seen in Table 1. Here, each section of the table represents a single iteration of the interactive method, where the first (bolded) row indicates the starting solution shown to the DM, the second row indicates the preferences specified by the DM and the following rows indicate the Pareto optimal solutions generated. For more details of the solution process, see [46].

In Table 1, each Pareto optimal solution is denoted by $\mathbf{z}_j$, of which $\mathbf{z}_7$ denotes the solution selected as the most preferred one by the DM. However, as noted in [46], the solutions generated are not actually Pareto optimal because the optimization methods did not necessarily converge, as the optimization had to be stopped prematurely due to the time constraints.

In what follows, we apply the agent assisted algorithm in the two-stage separation problem. The aim of the problem is to design a new type of separation process, and the process designer acted as the DM for during the solution process. Because we want to utilize all preference information provided in the previous research, we denote the results of [46] as the first decision phase for the agent assisted algorithm and start with the offline phase of step 5. In addition, preference agents selected only one Pareto optimal solution in step 6 with which we proceeded to step 1, where the surrogate problem was constructed for the first time with the PAINT method. In this way, we could build the surrogate problem with increased accuracy on the ares that the DM had shown interested in. The DM started the second decision phase

with step 2 and the agent assisted algorithm was followed till step 6 (b).

In what follows, we provide some details of the individual steps taken in the solution process. The offline phase in step 5 was started with the optimization agents first using the information from the interactive method agents to mimic the actions of the DM summarized in Table 1. For example, from the information of the iteration 3 of Table 1, four new interactive method agents were created. Each of them corresponded to one of the subproblems of the NIMBUS method generating a Pareto optimal solution (corresponding to the classification information). Each of the interactive agents employed a group of eight optimization agents. These 32 agents, using different optimization methods with different parameters, generated four new Pareto optimal solutions (one for each group). One of these solutions was $(1764, 0.20, 12030)$. To obtain this result, the optimization agents spent a total of 2600 function evaluations.

Let us consider this Pareto optimal solution for a while. It was used by the preference agent called classification agent to generate two new sets of preference information. As an example, the classification agent-based on support vector machines produced the classification $\left(I^{\leq 2270}, I^{\geq 2.3}, I^{\leq 7500}\right)$. Then this preference information was passed to interactive method agents to generate new Pareto optimal solutions. The corresponding actions were taken for each of the remaining three Pareto optimal solutions of iteration 3. Naturally, the corresponding steps were repeated for iterations 1 and 2 of Table 1. In this way, preference information available from the previous research was utilized.

Step 5 of the first offline phase was continued until the amount of time agreed with the DM was used up. In step 6, the preference agents selected one Pareto optimal solution z_1^2 (see Table 2) which was shown to the DM. Because the DM wanted to improve it, the second offline phase was started with step 1.

In step 1, the Pareto optimal solutions generated in step 5 of the first offline phase were used to construct a surrogate problem of the two-stage separation problem. The solution process continued with the second decision phase consisting of repeated steps from 2 till 4.

A collection of the results generated during the second decision phase as well as the preference information specified can be seen in Table 2. The Pareto optimal solutions were generated based on the preferences specified by the DM in a graphical user interface shown in Figure 5. Here, the DM has provided classification for the iteration 2 of the solution process (described in Table 2), and the method has generated four new approximated Pareto optimal solutions, that are shown to the DM. In the Figure 5, the DM is shown a single Pareto optimal solution, namely the solution a_3^2, on the left side. As can be seen, the Pareto optimal solution is shown with three vertical bars, each of which corresponding to an objective function. The first objective function is to be maximized, which is indicated by having the bar starting from the right end. The lowest (estimated) value that each objective function can achieve is shown to left of the bar, and the highest (estimated) value is shown to the right. The relative position of the current objective function value is indicated with an arrow pointing down, as well as the exact numeric value below the bar. The DM

Iter	Issue	Max Permeate (kg)	Min Impurity	Min Energy (kJ)
	Ideal	4693	0.00	0
	Nadir	0	23.56	48532
	z_4^2	**1773**	**0.29**	**8488**
1	Classif	$I^{\geq=1050}$	$I^{<}$	$I^{\leq=3400}$
	a_1^2	1051	0.25	4380
	a_2^2	983	0.23	4100
	a_3^2	109	0.03	466
	a_4^2	1030	0.25	4294
	a_3^2	**109**	**0.03**	**466**
2	Classif	$I^{\leq=1000.0}$	$I^{\geq=0.1}$	$I^{\geq=3500.0}$
	a_5^2	469	0.10	2368
	a_6^2	962	0.29	3889
	a_7^2	580	0.12	2952
	a_8^2	988	0.28	4032
	a_5^2	**469**	**0.10**	**2368**
3	Classif	$I^{\leq=1000.0}$	$I^{\leq=0.1}$	$I^{\geq=5000.0}$
	a_9^2	974	0.23	4102
	a_{10}^2	580	0.12	2952
	a_{11}^2	990	0.23	4158
	a_5^2	**469**	**0.10**	**2368**
4	Classif	$I^{\leq=2000.0}$	$I^{\geq=0.12}$	$I^{\geq=9000.0}$
	a_{12}^2	566	0.12	2881
	a_{13}^2	1909	0.57	7735
	a_{14}^2	902	0.21	3928
	a_{15}^2	1952	0.59	7887
	a_{12}^2	**566**	**0.12**	**2881**
5	Classif	$I^{\leq=2000.0}$	$I^{\geq=0.13}$	$I^{\geq=9000.0}$
	a_{16}^2	606	0.13	3033
	a_{17}^2	1909	0.57	7735
	a_{18}^2	938	0.22	4001
	a_{19}^2	1952	0.59	7887
Pref.	a_7^2	**580**	**0.12**	**2952**

Table 2: Solution process in the second decision phase with the agent assisted algorithm

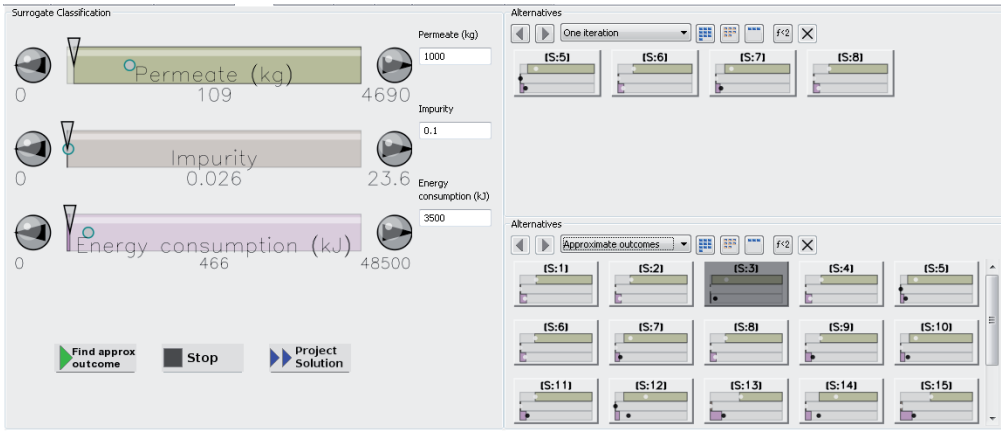


Figure 3: Graphical user interface used by the DM

indicates preferences by clicking the bar on location where the value is desired to be changed or by inputting a numeric value to the edit box located to the right of the bar. The obtained solutions are shown on the right side of the figure. For more information on the graphical user interface used during the decision phase by the DM, see [33].

Based on his understating of the problem (gained during the first decision phase), the DM decided to minimize the amount of impurity as much as possible, while maintaining reasonable bounds (to be seen in the table) for the other objectives. This was the first classification. He was shown four approximate Pareto optimal solutions of which he selected a_3^2 as it had the best impurity even though the values of the other objectives were not satisfactory.

For the second iteration, he gave an upper bound 0.10 for the impurity, while aspiring for the value 1000 for the permeate (to be maximized) and maintaining an upper bound of 3500 for the energy consumption. For the third iteration, he decided to continue with a_5^2 as this solution obeyed the upper bound of impurity even though the permeate value 469 was low.

In the third iteration, he allowed increment of energy consumption till 5000, in order to obtain the desired value 1000 for the permeate. From the results shown, he noticed that by increasing the impurity level, he might be able to obtain a better permeate value while maintaining a reasonable level of energy consumption.

The fourth iteration was started again from a_5^2 to see the effect of increasing the upper bound of impurity to 0.12. As hoped for, the permeate amount improved while the impurity bound was not violated without impairing the energy consumption. He selected a_5^2.

In the fifth iteration, he allowed the impurity to rise up to 0.13 in order to improve the permeate amount. In addition, he allowed the energy consumption to rise till 9000. This iteration was not satisfactory and he selected $(580, 0.12, 2952)$ as the final

Issue	Max Permeate (kg)	Min Impurity	Min Energy(kJ)
z_1^3	323	0.10	2858
z_2^3	479	0.14	4240
z_3^3	**1870**	**0.18**	**10697**
z_4^3	2901	0.60	26348

Table 3: Pareto optimal solutions obtained from second offline phase

approximate solution. This was the last step 4 (b) of the second decision phase.

After the second decision phase, the second offline phase of the agent assisted algorithm was started with the data summarized in Table 2. From the Pareto optimal solutions obtained in the second offline phase, the preference agents suggested four Pareto optimal solutions which would be preferred by the DM. Their number was four because it was a cognitively feasible number for the DM. These solutions are shown in Table 3. Of these, the DM selected z_3^3 as the most preferred solution for the two-stage separation problem. In it, the permeate and energy values were reasonable while maintaining a good impurity. The DM stopped the solution process because he was satisfied with the final solution.

As far as the solution process with the agent assisted algorithm is concerned, for the two-stage separation problem the algorithm could intelligently update the surrogate problem and enabled interaction with the DM without noticeable waiting times. To be more specific, the DM was able to obtain new approximate Pareto optimal solutions immediately after having specified preference information. This enabled him to explore the two-stage separation problem with five different preferences, that is, iteration, by spending only about half an hour of his time. This is a remarkable improvement compared to the original research reported in [46], where he was able to give only one classification per day.

The construction sets generated based on the preference information specified by the DM are illustrated in Figure 5. Here, Pareto optimal solutions generated based on the first decision phase are marked with diamonds and those based on the second decision phase are marked with circles. If the agent assisted algorithm had been continued, the PAINT surrogate problem would have been constructed from a combination of these two sets. The vectors of aspiration levels specified by the DM are marked with triangles pointing downwards for the first decision phase and with triangles pointing upwards for the second decision phase. As can be seen, the agent assisted algorithm was able to identify how the DM's interests changed and to generate new solutions on the Pareto frontier in those areas.

It should be noted that the amount of preference information specified by the DM did not decrease when compared to the original research. The slowness of the original solution process was explained by the computationally demanding problem. Now that the DM could see new solutions without having to wait for hours, it encouraged him to do a more thorough search of the interesting area of the approximate Pareto frontier. Nevertheless, the preference agents decreased the cognitive

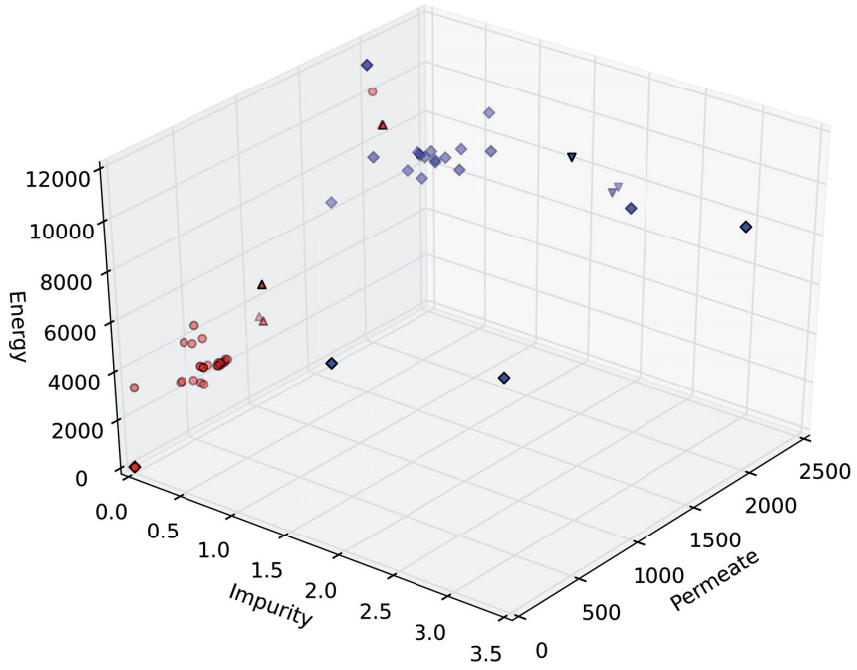


Figure 4: Construction sets created from DM's preferences

load in step 6 by allowing him to consider only a subset of the generated Pareto optimal solutions.

Originally, the DM regarded impurity as the most important objective. However, during the solution process he learned that he had to trade-off and give up in impurity in order to get reasonable values for the other objectives.

The aim of this research is not simply to decrease the overall computational cost of solving a multiobjective optimization problem. The main idea is to shift the computationally demanding elements from the decision phase to the offline phase in order to decrease waiting times imposed on the DM. Interestingly, when comparing the results obtained by the optimization agents during the offline phase to the results obtained with the interactive NIMBUS method (without the surrogate problem), the optimization agents were able to converge significantly faster and, thus, computational savings were clear. For example, optimization agents were able to generate the solution z_3^1 with 1700 function evaluations, but generating a similar solution without agents took almost 15000 function evaluations. This was not studied further, as our main focus was on supporting the DM in solving multiobjective optimization problems with interactive methods.

6 Discussion

With the new agent assisted algorithm we were able to generate satisfactory solutions to the DM and support him when solving the two-stage separation problem. The same DM was involved in the original research reported in [46] and found the new algorithm more convenient for exploring the Pareto frontier of the problem. He appreciated the fact that new approximate Pareto optimal solutions corresponding to his preference information were generated without waiting times.

It is intuitively evident that the agent assisted algorithm can increase the number of objective function evaluations used when solving a multiobjective optimization problem. In step 5 of the agent assisted algorithm, the agents generate more Pareto optimal solutions than the plain NIMBUS method would. However, by producing more Pareto optimal solutions based on the DM's preferences, the agent assisted algorithm is able to provide the DM with a more complete view of the problem. In this regard, the agent assisted algorithm could also be applied for computationally inexpensive problems by generating additional Pareto optimal solutions in the decision phase. The preference agents could then, as in step 6 of the agent assisted algorithm, select some of these Pareto optimal solutions to be shown to the DM. In this way, the agent assisted algorithm could encourage the DM to consider unexplored ares of the Pareto frontier that could be of interest. It could even be possible to employ the human learning agents to generate new preference information without additional function evaluations.

On the other hand, it is possible that the increase of the computational cost introduced by the agent assisted algorithm is not that significant. As pointed out in Section 5, it seems that the agent assisted algorithm was actually able to reduce the

computational cost of solving the two-stage separation problem by an order of magnitude. It is possible that as the optimization agents are guided based on the preference information obtained from the DM, they are able to converge faster towards the Pareto frontier. In addition, this effect may be amplified as the optimization agents share information on the best solutions found including the history of previous iterations of the interactive method. However these results cannot be directly generalized to all solution methods.

When shortening the waiting times imposed on the DM, there is naturally some price to be paid. In the agent assisted algorithm this means showing approximate Pareto optimal solutions to the DM. This may cause some impreciseness of preference information. Clearly, the DM must be informed of this. On the other hand, real-world problems contain typically nonlinear and nonconvex objective functions. Furthermore, these problems are usually considered as a black-box, i.e. the exact formulations or even characteristics of the objectives problem are usually do not know before solving the problem. Therefore, even when solving the problem without using surrogates, we cannot guarantee that the obtained Pareto optimal solutions are accurate. In practice, the time available for solving the problem usually determines the quality of the obtained solutions. As the agent assisted algorithm allows for conducting the time intensive parts of the solution process without involvement of the DM, it is possible to expand the time available and therefore provide the DM with more accurate solutions.

Naturally, if different interactive methods and surrogate problem constructing techniques are used, appropriate agents of the four types must be developed. However, the ideas of implementing agents presented here are not limited to the considered application.

7 Conclusions

In this paper, we have introduced an agent assisted interactive multiobjective optimization algorithm for solving computationally demanding problems. Motivated by the benefits of using interactive methods we, with this algorithm, enable applying them in computationally demanding problems. The algorithm is general and can be used with different interactive methods and surrogate problem construction techniques.

The algorithm employs a computationally inexpensive surrogate problem and four types of agents. The agents observe the DM's actions when using an interactive method. With these observations the agents intelligently increase the accuracy of the surrogate by updating it in the areas the DM is interested in, without a need for additional information. Besides that, the agents working with the surrogate problem shorten the waiting times imposed on the DM and decrease the amount of preference information required.

We solved a computationally demanding two-stage separation problem with the new agent assisted algorithm involving the interactive NIMBUS method and the

PAINT method to construct the surrogate problem. The DM appreciated the fast solution process and the results obtained. As the experiences were most encouraging, we consider this research direction to be fruitful and conclude that agent assisted algorithm should be applied in and tested with various computationally demanding problems.

Acknowledgments

This work was supported on the part of Vesa Ojalehto by the Jyväskylä Doctoral Program in Computing and Mathematical Sciences in Finland, by the Nyyssönen foundation and by the KAUTE foundation. The authors wish to thank Mr. Jouni Savolainen for participating in this research as the decision maker.

References

[1] K. J. Arrow. Rational choice functions and orderings. *Economica*, 26(102):121–127, 1959.

[2] F. B. Aydemir, A. Günay, F. Öztoprak, Ş. İker Birbil, and P. Yolum. Multi-agent cooperation for solving global optimization problems: An extendible framework with example cooperation strategies. *Journal of Global Optimization*, 57(2):499–519, 2013.

[3] S. Bechikh, L. Ben Said, and K. Ghedira. Estimating nadir point in multi-objective optimization using mobile reference points. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9, 2010.

[4] S. Bechikh, L. Ben Said, and K. Ghedira. Negotiating decision makers' reference points for group preference-based evolutionary multi-objective optimization. In *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, pages 377–382, 2011.

[5] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, editors. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, 2008.

[6] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5(6):961–970, 1992.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[8] D. Cvetković and I. Parmee. Agent-based support within an interactive evolutionary design system. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 16(5):331–342, 2002.

[9] P. Davidsson, J. A. Persson, and J. Holmgren. On the integration of agent-based and mathematical optimization techniques. In N. T. Nguyen, A. Grzech, R. Howlett, and L. C. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, pages 1–10. Springer Berlin Heidelberg, 2007.

[10] R. Drezewski and L. Siwik. Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In L. Rutkowski, R. Tadeusiewicz, L. Zadeh, and J. Zurada, editors, *Artificial Intelligence and Soft Computing–ICAISC 2008*, pages 388–397. Springer Berlin Heidelberg, 2008.

[11] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: a Practical Guide*. Wiley, 2008.

[12] L. Gardiner and D. Vanderpooten. Interactive multiple criteria procedures: Some reflections. In J. Climaco, editor, *Multicriteria Analysis*, pages 290–301. Springer, 1997.

[13] S. Greco, M. Kadziński, V. Mousseau, and R. Słowiński. Robust ordinal regression for multiple criteria group decision: UTAGMS-GROUP and UTADISGMS-GROUP. *Decision Support Systems*, 52(3):549–561, 2012.

[14] C. Grimme, J. Lepping, and U. Schwiegelshohn. Multi-criteria scheduling: An agent-based approach for expert knowledge integration. *Journal of Scheduling*, 16(4):369–383, 2013. cited By (since 1996)2.

[15] J. Hakanen, K. Sahlstedt, and K. Miettinen. Wastewater treatment plant design and operation under multiple conflicting objective functions. *Environmental Modelling & Software*, 46(1):240–249, 2013.

[16] M. Hartikainen, K. Miettinen, and M. M. Wiecek. PAINT: Pareto front interpolation for nonlinear multiobjective optimization. *Computational Optimization and Applications*, 52:845–867, 2012.

[17] M. Hartikainen, V. Ojalehto, and K. Sahlstedt. Applying approximation method PAINT and interactive method NIMBUS to multiobjective optimization of operating a wastewater treatment plant. *Engineering Optimization*, to appear.

[18] M. Hasenjäger and B. Sendhoff. Crawling along the Pareto front: Tales from the practice. In *The 2005 IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, pages 174–181, 2005.

[19] E. Jacquet-Lagreze and Y. Siskos. Preference disaggregation: 20 years of MCDA experience. *European Journal of Operational Research*, 130(2):233–245, 2001.

[20] I. Kaliszewski. Out of the mist–towards decision-maker-friendly multiple criteria decision making support. *European Journal of Operational Research*, 158(2):293–307, 2004.

[21] P. Korhonen, S. Salo, and R. E. Steuer. A heuristic for estimating nadir criterion values in multiple objective linear programming. *Operations Research*, 45(5):pp.751–757, 1997.

[22] H. Li and H. Ding. Agent-based evolutionary algorithms applied to constrained multi-objective optimization problems. *Applied Artificial Intelligence*, 26(10):941–951, 2012.

[23] I. Lobel, A. Ozdaglar, and D. Feijer. Distributed multi-agent optimization with state-dependent communication. *Mathematical Programming*, 129(2):255–284, 2011.

[24] T. Máhr, J. Srour, M. de Weerdt, and R. Zuidwijk. Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, 18(1):99–119, 2010.

[25] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[26] K. Miettinen. Interactive multiobjective optimization method NIMBUS applied to continuous casting of steel. In N. Bandyopadhyay, P. Chattopadhyay, and S. Chattopadhyay, editors, *International Workshop on Neural Network and Genetic Algorithm in Materials Science and Engineering, Proceedings*, pages 58–72. Tata McGraw-Hill Publishing Company, 2006.

[27] K. Miettinen. Using interactive multiobjective optimization in continuous casting of steel. *Materials and Manufacturing Processes*, 22(5):585–593, 2007.

[28] K. Miettinen and M. M. Mäkelä. Interactive multiobjective optimization system WWW-NIMBUS on the Internet. *Computers & Operations Research*, 27(7-8):709–723, 2000.

[29] K. Miettinen and M. M. Mäkelä. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research*, 170(3):909–922, 2006.

[30] K. Miettinen, M. M. Mäkelä, and J. Toivanen. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization*, 27(4):427–446, 2003.

[31] K. Miettinen, F. Ruiz, and A. P. Wierzbicki. Introduction to multiobjective optimization: Interactive approaches. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 27–57. Springer-Verlag, 2008.

[32] M. M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization Analysis and Algorithms with Applications to Optimal Control*. World Scientific, 1992.

[33] V. Ojalehto, K. Miettinen, and T. Laukkanen. Implementation aspects of interactive multiobjective optimization for modeling environments: The case of GAMS-NIMBUS. *Computational Optimization and Applications*, 58(3):757–779, 2014. doi:10.1007/s10589-014-9639-y.

[34] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez and J. Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Kluwer Academic Publishers, 1994.

[35] W. Price. Global optimization by Controlled Random Search. *Journal of Optimization Theory and Applications*, 40(3):333–348, 1983.

[36] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[37] H. Ruotsalainen, E. Boman, K. Miettinen, and J. Tervo. Nonlinear interactive multiobjective optimization method for radiotherapy treatment planning with Boltzmann transport equation. *Contemporary Engineering Sciences*, 2(9):391–422, 2009.

[38] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[39] R. A. Sarker and T. Ray. Agent based evolutionary approach: An introduction. In R. A. Sarker and T. Ray, editors, *Agent-Based Evolutionary Search*, pages 1–11. Springer Berlin Heidelberg, 2010.

[40] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization*. Academic Press, Inc., 1985.

[41] A. K. Sen. Choice functions and revealed preference. *The Review of Economic Studies*, 38(3):307–317, 1971.

[42] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie. Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics*, 20(4):415–431, 2006.

[43] J. D. Siirola, S. Hauan, and A. W. Westerberg. Toward agent-based process systems engineering: Proposed framework and application to non-convex optimization. *Computers & Chemical Engineering*, 27(12):1801–1811, 2003.

[44] J. D. Siirola, S. Hauan, and A. W. Westerberg. Computing Pareto fronts using distributed agents. *Computers & Chemical Engineering*, 29(1):113–126, 2004.

[45] C. Simons, I. Parmee, and R. Gwynllyw. Interactive, evolutionary search in upstream object-oriented class design. *Software Engineering, IEEE Transactions on*, 36(6):798–816, 2010.

[46] K. Sindhya, V. Ojalehto, J. Savolainen, H. Niemistö, J. Hakanen, and K. Miettinen. Coupling dynamic simulation and interactive multiobjective optimization for complex problems: An APROS-NIMBUS case study. *Expert Systems with Applications*, 41(5):2546–2558, 2014.

[47] L. Siwik and S. Natanek. Solving constrained multi-criteria optimization tasks using elitist evolutionary multi-agent system. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3358–3365, 2008.

[48] K. Socha and M. Kisiel-Dorohinicki. Agent-based evolutionary multiobjective optimisation. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. (CEC '02)*, volume 1, pages 109–114, 2002.

[49] I. Steponavice, S. Ruuska, and K. Miettinen. A solution process for simulation-based multiobjective design optimization with an application in paper industry. *Computer-Aided Design*, 47:45–58, 2014.

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

[51] S. Talukdar, L. Baerentzen, A. Gove, and P. De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4(4):295–321, 1998.

[52] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *Automatic Control, IEEE Transactions on*, 31(9):803–812, Sep 1986.

[53] T. Tveit, T. Laukkanen, V. Ojalehto, K. Miettinen, and C. Fogelholm. Interactive multi-objective optimisation of configurations for an oxyfuel power plant process for CO2 capture. *Chemical Engineering Transactions*, 29:433–438, 2012.

[54] H. Uzawa. Note on preference and axioms of choice. *Annals of the Institute of Statistical Mathematics*, 8:35–40, 1956.

[55] A. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3:391–405, 1982.

[56] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making: Theory and Application*, pages 468–486. Springer, 1980.

[57] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Inc., 2002.

[58] L. Xu, T. Reinikainen, W. Ren, B. P. Wang, Z. Han, and D. Agonafer. A simulation-based multi-objective design optimization of electronic packages under thermal cycling and bending. *Microelectronics Reliability*, 44(12):1977–1983, 2004.