Author(s): Ojalehto, Vesa; Miettinen, Kaisa; Laukkanen, Timo

Title: Implementation aspects of interactive multiobjective optimization for modeling environments: The case of GAMS-NIMBUS

Year: 2014

Version:

Please cite the original version:

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

Implementation Aspects of Interactive Multiobjective Optimization for Modeling Environments: The Case of GAMS-NIMBUS

Vesa Ojalehto*
Kaisa Miettinen*
Timo Laukkanen†

January 29, 2014

Abstract

Interactive multiobjective optimization methods have provided promising results in the literature but still their implementations are rare. Here we introduce a core structure of interactive methods to enable their convenient implementation. We also demonstrate how this core structure can be applied when implementing an interactive method using a modeling environment. Many modeling environments contain tools for single objective optimization but not for interactive multiobjective optimization. Furthermore, as a concrete example, we present GAMS-NIMBUS Tool which is an implementation of the classification-based NIMBUS method for the GAMS modeling environment. So far, interactive methods have not been available in the GAMS environment, but with the GAMS-NIMBUS Tool we open up the possibility of solving multiobjective optimization problems modeled in the GAMS modeling environment. Finally, we give some examples of the benefits of applying an interactive method by using the GAMS-NIMBUS Tool for solving multiobjective optimization problems modeled in the GAMS environment.

Keywords: Multiple objective programming, interactive methods, NIMBUS method, modeling languages, Pareto optimality

*Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland

†Aalto University, School of Engineering, Department of Energy Technology, P.O.Box 14400, FI-00076 Aalto, Finland

1 Introduction

Many real-world optimization problems contain several, conflicting objectives that should be optimized at the same time. For such problems it is usually not possible to find a single optimal solution. Instead, a multiobjective optimization problem typically has several compromise solutions with different trade-offs between objective functions. In order to select the best solution from the set of these so-called *Pareto optimal* solutions, some additional information is needed. This information, called preference information, is usually obtained from a decision maker (DM), who is assumed to have expertise in the domain of the optimization problem being solved.

As the role of the DM is important when solving multiobjective optimization problems, multiobjective optimization methods are often classified according to the role of the DM (see, e.g., [26]). In this paper we consider so-called *interactive* methods, where the solution process is iterative consisting of steps where some information is shown to the DM and the DM is asked to provide preference information. Pareto optimal solutions are generated based on this information until the DM finds satisfactory solution.

Interactive methods have provided promising solutions in various fields of application including reservoir management [1], wastewater treatment management [14, 15], optimal control in steel casting [28, 38], chemical engineering [31], construction of bridges [40] and analyzing air pollution [46], etc. Even though many interactive methods have been proposed in the literature during the years (see, e.g., [22, 26, 39] and references therein), implementations of interactive methods are scarce. Separating methodological issues from technical ones is suggested in [20] as a way to enhance method implementations. We follow this line here.

The objectives of this paper are twofold. First we present a core structure of characteristics common to interactive methods and introduce general guidelines on how these characteristics can be utilized to implement an interactive method. On the other hand, models of various phenomena have been created in different modeling environments over the years but, typically, the optimization capabilities available are limited to single objective optimization. Our second objective is to demonstrate with the help of the core structure that the optimization capabilities of modeling environments can be extended without too much effort by providing a possibility to use interactive multiobjective optimization methods. As a concrete example, we apply the core structure to implement the interactive NIMBUS method [33, 34, 35, 37] for the GAMS environment [5], an example of widely used modeling environments. With the resulting GAMS-NIMBUS Tool, we can provide users of the GAMS environment an access to an interactive multiob-

jective optimization method and, on the other hand, the NIMBUS method included in the GAMS-NIMBUS Tool can take advantage of the modeling capabilities of GAMS and apply single objective GAMS solvers as a part of the solution process.

The rest of this paper is organized as follows. In Section 2 we introduce basic concepts of multiobjective optimization that are relevant for the rest of the paper. In Section 3 we briefly describe general characteristics of interactive methods as well as introduce a core structure common to many interactive methods to enable their implementation. In Section 4 we provide insight on how to prepare an existing model (e.g., in a modeling environment) of an optimization problem to be solved with an interactive method utilizing the core structure. Then in Section 5 we introduce a new GAMS-NIMBUS Tool, which is an implementation of the interactive NIMBUS method for the GAMS modeling environment. The GAMS-NIMBUS Tool implementation is based on the ideas presented in the two previous sections. In Section 6 we present two numerical examples to demonstrate how one can apply our findings to solve multiobjective optimization problems in an interactive way in a modeling environment and demonstrate the advantages of this approach. The paper is concluded in Section 7.

2 Some Concepts of Multiobjective Optimization

In this section we briefly present the concepts and notation of multiobjective optimization that are relevant for the following sections. We consider multiobjective optimization problems of the form

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

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 $f_j(\mathbf{x}) \leq f_i(\hat{\mathbf{x}})$ for all $i = 1, ..., k$ and $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$ for least one $j = 1, ..., k$. These objective vectors are called Pareto optimal solutions to problem (1), and a set of Pareto optimal solutions is called a *Pareto optimal set* or a *Pareto frontier* [26]. Therefore, instead of a single optimal solution, a multiobjective

optimization problem with conflicting objectives has several different Pareto optimal solutions with different trade-offs.

Many methods have been developed for solving multiobjective optimization problems. The overall process to find a solution to be called a final solution for problem (1) is called a *solution process*.

As Pareto optimal solutions cannot be compared without some external preference information, the most preferred of the solutions can be selected based on the preference information expressed by a *decision maker* (DM). As mentioned in the introduction, multiobjective optimization methods can be classified according to the role of the DM in the solution process [26]. In this research, we consider *interactive* methods where the preference information is obtained iteratively from the DM during the solution process. The interactive solution process aims at supporting the DM in identifying the Pareto optimal solution which best corresponds to his/her preferences. This most preferred solution is here the *final solution*.

Information about the ranges of the objective function values in the Pareto optimal set may be useful for the DM. It is possible to determine the best (ideal) value of each objective function by optimizing it individually (subject to the feasible set S). The worst values of the objective functions in the Pareto optimal set can be estimated, for example, by using a so-called pay-off table [2, 26] which can be formed after individual optima have been found or by using some more advanced heuristic (see, e.g., [8].) The vectors representing the best and the worst objective function values in the set of Pareto optimal solutions are called an *ideal objective vector* $\mathbf{z}^\star \in \mathbf{R}^k$ and a *nadir objective vector* $\mathbf{z}^{\mathrm{nad}} \in \mathbf{R}^k$, respectively. For computation reasons, we define also a *utopian objective vector* $\mathbf{z}^{\star\star}$, which is strictly better than $\mathbf{z}^\star$ in each component.

A common way of solving a multiobjective optimization problem is to use scalarization, that is, to reformulate the problem together with the preference information available as a single objective optimization problem, to be called a (scalarized) *subproblem*. The objective function of the subproblem is called a *scalarizing function*. By selecting the scalarizing function carefully, one can guarantee that the decision vector which is optimal to the subproblem is Pareto optimal to problem (1). For further details, see, e.g., [26].

Interactive methods discussed in this research are based on scalarization of the multiobjective optimization problem. An example of such a scalarized subproblem that can be used to generate Pareto optimal decision vectors for (1) is called an achievement scalarizing function [49]

$$\begin{aligned} \text{minimize} \quad & \max_{i=1,\ldots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\text{nad}} - z_i^{\star\star}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} \quad & \mathbf{x} \in S, \end{aligned} \tag{2}$$

where $\bar{\mathbf{z}} \in \mathbf{R}^k$ is a *reference point* representing the DM's preferences in the form of desirable objective function values and $\rho > 0$ is a so-called augmentation coefficient. In this formula we assume that all object functions are to be minimized.

Different interactive methods utilize different scalarized subproblems involving different types of preference information. Besides reference points, examples of other types of preference information are classification, marginal rates of substitution and selecting from a small set of Pareto optimal solutions (for further details see, e.g., [22, 26, 39, 42]). When selecting an interactive method to be used one should check what kind of preference information the DM is willing and able to provide.

The scalarized subproblem must be solved with a single objective optimization method that is appropriate to the characteristics of the problem in question (e.g., differentiable or nondifferentiable, convex or nonconvex, etc.). By changing the preference information, e.g., values of the components of the reference point vector $\bar{\mathbf{z}}$ in subproblem (2), one can generate different Pareto optimal solutions and this is how the DM can direct the solution process. For further information on multiobjective optimization see, e.g., [4, 7, 26, 47] and references therein.

Before introducing the actual core structure of interactive multiobjective optimization methods we need to define concepts model and solver that are needed in the following sections. In order to solve an optimization problem with a computer it must be expressed as computational model. This can be done, for example, using a modeling language in some modeling environment or with some simulator software. Such an expression of the optimization problem is called a *model*. When modeling a multiobjective optimization problem it should be noted that the preference information to be given by the DM is typically based on objective function values, so the objective functions in the model should be meaningful for the decision maker.

In this research we do not consider single objective optimization methods. Instead, we assume that there exists an appropriate implementation of a method that can be used to solve the subproblem in question. Such an implementation is called a *solver*. For example, when solving problems modeled with the GAMS modeling language, it is natural to use single objective solvers included in the GAMS environment. The same is naturally valid for other modeling environments.

3 Core Structure of Interactive Multiobjective Optimization Methods

In this section we introduce a core structure which characterizes features common to scalarization-based interactive multiobjective optimization methods. The core structure can be used as a general guideline when building an implementation of an interactive multiobjective optimization method. The core structure is not devoted to any particular modeling environment or modeling language but because of its general nature it is suitable for any implementation environment.

As mentioned earlier, the solution process with an interactive multiobjective optimization method is iterative. On each iteration the DM is provided with Pareto optimal solutions and asked to specify new preference information to generate more satisfactory new Pareto optimal solution(s) for the next iteration. With this iterative process, the DM can influence from which part of the Pareto optimal set the final solution is being looked for. During the iterative exploration of the Pareto frontier, the DM can obtain new information and insight about the interdependencies among the objective functions. It is even possible that the new knowledge obtained affects his or her preferences, leading to solutions which were not previously considered. As mentioned in Section 2, when solving the multiobjective optimization problem (1), the solution process generally aims at supporting the DM in identifying a single Pareto optimal solution that he or she finds as the most preferred (see, e.g. [39]).

A *core structure* of an interactive multiobjective optimization method can be generally characterized to contain the following steps:

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 current solution.
3. Ask the DM to provide preference information related to the current solution.
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.

The core structure introduced is in line with the general interactive approaches presented in [22, 42] as well as with methods described, e.g., in [31, 39]. As can be seen, this core structure has two active participants: a) the DM, who is asked to provide preference information and to select the solution that best corresponds to his or her preferences and b) the algorithm, which provides the DM with the initial solution and new Pareto optimal solutions based on preference information given. Therefore, an implementation of an interactive method can be divided into two distinct parts: the *user interface* (steps 3 and 5) and the *algorithm* part (steps 1, 2 and 4).

The core structure is an answer to the need posed in [20] of separating methodological issues (algorithm) from technical ones (user interface) in order to enhance method implementations. Thus, in this research we focus on the algorithm side and do not consider user interface implementation (for such studies see, e.g., [32, 48]). Besides, user interfaces may need application-specific elements but we want to retain on a more general algorithm level without going into application-specific details. One example of an implementation of a user interface is demonstrated in Section 5.

It should be noted that different methods can require different types of user input in addition to the preference information. For example, the solution process may contain parameters that the DM can change, such as the number of new Pareto optimal solutions to be obtained per each iteration. If the additional information is to be given once, it is a part of the initialization in step 1. Otherwise, it is a part of step 3.

As mentioned earlier, we consider scalarization-based interactive multiobjective optimization methods where Pareto optimal solutions are generated by solving single objective subproblems. The algorithm side of an interactive method uses the original multiobjective optimization problem and the preference information to formulate a scalarized single objective subproblem. When this subproblem is solved with an appropriate single objective solver, the Pareto optimal solution generated should correspond to the given preference information. As mentioned, each interactive method usually has a distinct fashion for the DM to express his or her preference information and different methods use different scalarizations to create the subproblem (see, e.g. [6, 37, 39, 41, 42, 49]).

Steps 1 and 2 of the core structure of an interactive method presented do not depend on the DM, so these steps can be regarded as a single *initialization stage* at the beginning of the solution process. During the initialization, in step 1, the ideal and the nadir objective vectors can be determined to provide information of the ranges of the objective function values attainable in the Pareto optimal set as described in Section 2, or their estimates can be obtained from the DM. In step 2, the initial Pareto optimal solution

is generated as the starting point of the solution process by solving some method-specific subproblem providing a Pareto optimal solution. Alternatively, a starting solution can be asked from the DM. In the latter case, the Pareto optimality must be checked by projecting the solution to the Pareto frontier using, e.g. subproblem (2) (by setting the objective vector of the starting solution as the reference point).

Whenever an interactive method is implemented, the algorithm part of the implementation requires various models. Creating these models means implementing method-specific subproblems in the environment in question. Actually this is all that the implementation of the algorithm part needs. Naturally, to get started we need a *multiobjective problem model* defining the original multiobjective optimization problem. This model is used by all other models to determine the feasible set of the problem and to calculate objective function values.

If the initialization stage requires calculating ideal and nadir objective vectors, a model is needed for optimizing each individual objective function separately. As mentioned in Section 2, the ideal objective vector and an estimated nadir objective vector can then be obtained. One more model is needed in the initialization stage for step 2 to create the initial Pareto optimal solution. This typically involves implementing a method-specific subproblem which does not involve preference information (as it is not yet available at the very beginning of the solution process). Furthermore, step 4 requires implementing a *scalarization model* for creating Pareto optimal solution(s) according to the preference information specified. In some interactive methods step 4 uses several subproblems and in such cases we must implement several scalarization models.

The structure of a general interactive multiobjective optimization method following the core structure can be seen in Figure 1. The results of the initialization stage are passed to the user interface, where the DM can express his or her preference information in step 3. This information is then passed to the scalarization model, implementing step 4 of the core structure. The solution obtained from the scalarization model is a new Pareto optimal solution, which is passed back to the user interface for step 5 where the DM can select the best solution from the set of generated Pareto optimal solutions. If the selected solution is not satisfactory, the solution process is continued from step 3. Otherwise, the selected solution (both decision and objective vectors) is set as the final solution.

As mentioned, steps 2 and 4 provide the decision maker with new Pareto optimal solutions by solving scalarized subproblems. It is possible that these steps use very similar subproblems. Therefore, it should be noted that the models involved can share some common elements, which makes the imple-

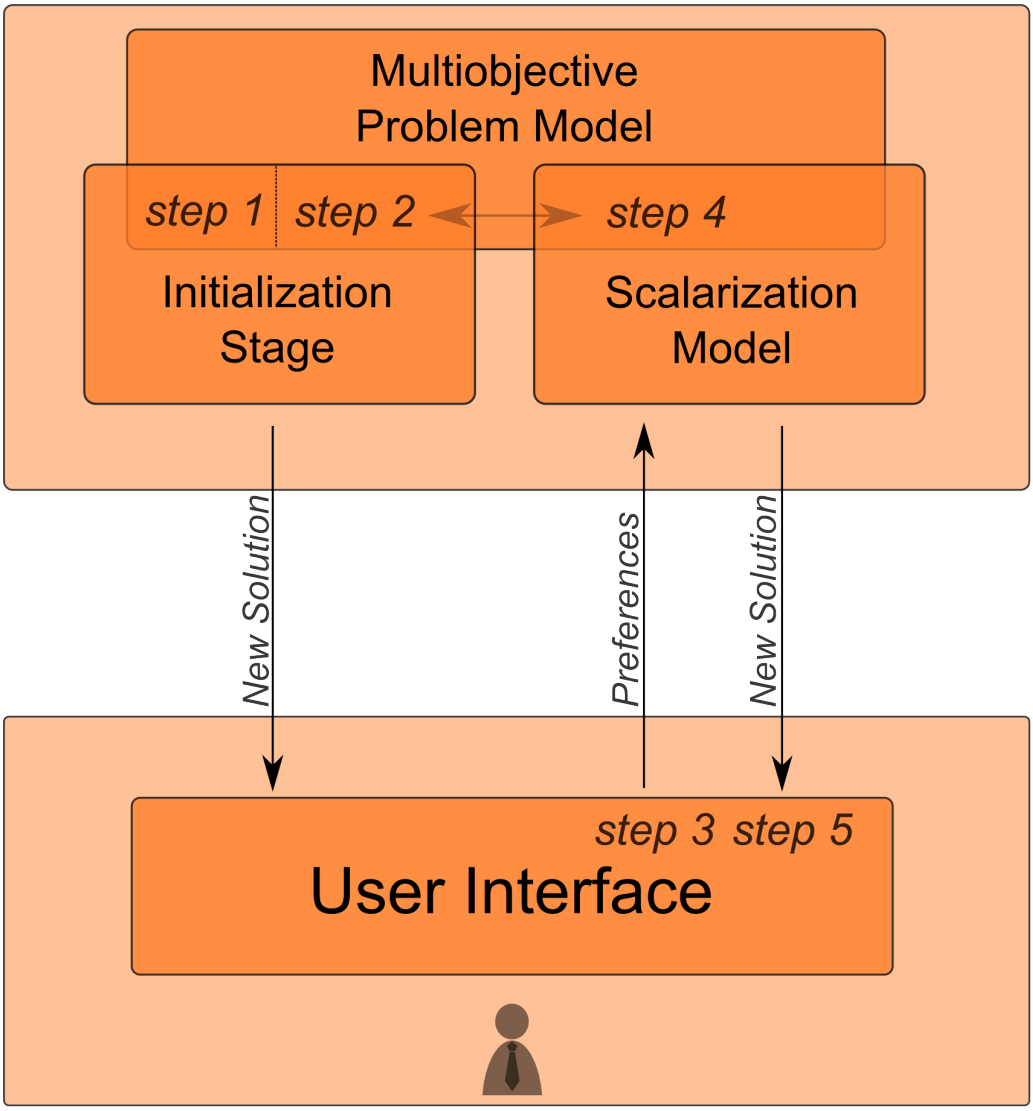

Figure 1: Core structure and information exchange in an interactive method

mentation even easier.

With the core structure and division of the effort and roles between a user interface and an algorithm, implementing scalarization-based interactive multiobjective optimization methods should be straightforward. An example of a multiobjective optimization method containing the six steps described in the core structure is the NIMBUS method [34, 35, 37] outlined in Appendix A. Next we give a short description of how an existing problem model implemented in a modeling environment can be prepared to be solved using an implementation of an interactive multiobjective optimization method following the core structure.

4 Preparing the Multiobjective Problem Model

We have noticed how optimization problems are often modeled as single objective optimization problems even though their real character would necessitate considering multiple conflicting objectives simultaneously. One reason for such a simplification is the lack of appropriate multiobjective optimization tools available. In particular, models of various phenomena have been created in different modeling environments over the years but, typically, the optimization capabilities available are limited to single objective optimization. In the previous section we introduced a core structure for interactive multiobjective optimization methods and we can summarize that implementing an interactive method, e.g., in a modeling environment simply involves implementing the multiobjective optimization problem and the method-specific subproblems in the modeling language in question. In this section, we give a short overview on how to prepare an existing (possibly originally single objective) optimization problem model for a modeling environment to be solved with an implementation of an interactive multiobjective optimization method.

In what follows, we present how a multiobjective problem model (as defined for the core structure in the previous section) can be created in a modeling environment based on a previously implemented model. Here this model is referred as *original problem model*. It should be noted that the implementation guidelines given are not specific to any modeling environment. The given information can be easily applied for most modeling environments, such as GAMS [5], AMPL [10] and AIMMS [3]. In fact, in addition to the case GAMS-NIMBUS described in this research, the guidelines given here have already been applied to implement the interactive NIMBUS method using the OPL modeling language [19] to solve multiobjective optimization problems.

Here we assume that the DM wants to solve some problem involving

multiple conflicting objectives and an interactive multiobjective optimization method is desired to be applied. Furthermore, we assume that the DM already has some problem model available. The original problem model can be a single objective model, to which the DM desires to add additional objective functions, or to convert some of the constraints of the original problem model as objective functions. Alternatively, some noninteractive multiobjective optimization method may have already been applied to the problem model (i.e., the problem is already in the form (1), and the DM now wishes to solve the problem with an interactive multiobjective optimization method implemented within a modeling environment. In any case, we assume that the DM has already established which objective functions (depending on decision variables) are to be optimized, and those objective functions (and possible constraints) are expressed with some modeling language.

The process of preparing a multiobjective problem model from an existing original problem model to be solved with an interactive multiobjective optimization method implementation involves the following steps:

1. Create a new multiobjective problem model from an original problem model.
   (a) Select an original problem model to be the basis for the multiobjective problem model.
   (b) Identify equations from the problem model to be directly selected as objective functions.
   (c) Write additional objective functions as new equations.
2. Edit the properties of the multiobjective problem model.
   (a) Objective function properties can be edited, e.g., whether the function should be minimized or maximized.
   (b) Single objective solver and its parameter values can be changed.

To be more specific, the first step of preparing a multiobjective problem model is to select an original problem model and identifying which equations stand for objective functions of the multiobjective optimization problem. The objective functions can be either selected from a list of existing equations obtained from the problem model, or they can be written as new equations. The list of equations can be generated, for example, by finding scalar equations where the left or the right side of the equation contains only a single item. Additionally, if the original model has been previously used to solve

the problem in question with some single objective or noninteractive multiobjective optimization methods, the related commands (e.g., GAMS solve statements if GAMS is used) should not be included in the multiobjective problem model.

Next, additional information related to the model of the multiobjective optimization problem may be needed, for example, whether the objective functions are to be maximized or minimized. This should be noted that the interactive multiobjective optimization method can take this into account when solving the problem. If objective functions are assumed to be minimized by default and an objective function is indicated to be maximized, the objective function values of the latter must be multiplied by -1 in the model. In the user interface, objective function values should be shown to the DM without this conversion.

Once the multiobjective problem model is finished, the corresponding multiobjective optimization problem can be solved using an implementation of an interactive method. In the next section, we provide a concrete example of an implementation of an interactive method and in the following section we solve two problems with the resulting tool.

5 GAMS-NIMBUS Tool

In this section we demonstrate how an implementation of an interactive multiobjective optimization method can be prepared in a modeling environment based on the ideas presented in Sections 3 and 4. As a concrete example of applying the core structure we implement the so-called synchronous NIMBUS method [37] (described in Appendix A) in GAMS to create a new GAMS-NIMBUS Tool that can be used to solve optimization problems involving multiple objectives. As for the user interface for the GAMS-NIMBUS Tool, we use the interface previously developed for the IND-NIMBUS software framework [27]. (If a user interface was not available, it could be implemented, for example, with the ASK utility of the GAMS environment or, alternatively, simply by manually editing input files.)

Some other interactive multiobjective optimization method (e.g., methods presented in [22, 42]) could be easily used instead of NIMBUS by simply replacing the models of the scalarized subproblems used in NIMBUS by the models of the subproblems of the other method. (Naturally, if the preference information required from the DM is not similar to the information used in the NIMBUS method, the user interface must be adapted in an appropriate way.)

5.1 Implementing NIMBUS with the Core Structure for the GAMS Environment

As mentioned in Section 3, the implementation of the algorithm part of the NIMBUS method necessitates implementing various models. Firstly there is the multiobjective problem model, which models the original problem to be optimized. Secondly, there are the models utilized by the initialization stage and scalarization, which create new Pareto optimal solutions.

In the GAMS environment, the multiobjective problem model defines the problem to be solved as a GAMS model. That is, the multiobjective problem model contains GAMS expressions defining the objective functions and constraints of the problem as functions of the decision variables. In addition, the multiobjective problem model contains other information, e.g., the model should contain ideal and nadir objective vector values when available, so there is no need to calculate them for every interactive method iteration. Therefore, the multiobjective problem model is included in the models of the initialization stage as well as in the scalarization model as a submodel in order to grant those models an access to the GAMS equations and formulations defining the original optimization problem as well as other information.

In step 1 of the initialization stage, the ideal and the nadir objective vectors are calculated and estimated, respectively, as mentioned in Sections 2 and 3. In the NIMBUS method, step 2 means solving a subproblem (2) using $\bar{z}_i = (z_i^{\text{nad}} + z_i^{\star\star})/2$ for $i = 1, ..., k$ as components of the reference point to obtain a so-called neutral compromise solution [50] as a starting point for the solution process. In the GAMS-NIMBUS Tool, this subproblem, as well as the single objective optimization problems for calculating the components of the ideal objective vector are naturally implemented in the GAMS modeling language.

As mentioned earlier, the scalarization model contains the single objective subproblem used in step 4 to generate a new Pareto optimal solution based on the preference information specified (obtained from the DM via the user interface). As described in Appendix A, the NIMBUS method utilizes up to four different subproblems, which can generate up to four different Pareto optimal solutions for the same preference information (as justified in [36]). Thus, we need four scalarization models in GAMS-NIMBUS, all expressed in the GAMS modelilng language. It should be noted that one of the subproblems includes additional constraints based on preference information besides the original constraints.

5.2 Description of the GAMS-NIMBUS Tool

The NIMBUS method is based on the classification of the objective functions (see Appendix A). In other words, the DM indicates with the help of a classification how the current Pareto optimal solution should be altered to get a more preferred solution. An example of the classification phase of the NIMBUS method in the GAMS-NIMBUS Tool software (similar to the IND-NIMBUS software) is given in Figure 2, illustrating the solution process of Example 1 to be described later. Here each bar represents an objective function value and its ranges (i.e. components of ideal and nadir objective vectors). As can be seen, Example 1 has three objective functions. Of these, the third is to be maximized, which is indicated by placement of the colored bar on the right side. Thus, the interpretation is the same for both objective functions to be minimized and maximized: the shorter the bar, the better the value. The classification is done by clicking different parts of objective function bars, depending on how the DM desires to change the corresponding objective function values in order to get a more desirable Pareto optimal solution. If a value is desired to be decreased as much as possible, the DM clicks the arrow pointing to the left. If, instead, an increase is desired, he or she clicks the arrow pointing to the right. If the DM desires to give some bound or a level till which the objective function value should be improved, he or she can click the actual bar and the corresponding value is then shown in the edit box next to the objective function bar. The DM can then edit the value, if he or she so desires. If the current objective function value is deemed suitable, the DM clicks the arrow pointing downwards on the top of the bar.

Once the DM is satisfied with his or her classification, he or she can simply click the play button at the bottom of the application screen or on the tool-bar, and the GAMS-NIMBUS Tool calculates new Pareto optimal solutions based at the given preference information. New Pareto optimal solutions are obtained by solving the constructed GAMS scalarization models.

Before choosing to calculate new Pareto optimal solutions, the DM can, if he or she so wishes, change the maximum number of new solutions to be calculated. As a default, the NIMBUS method forms four different subproblems from the same preference information and each of them can provide the DM with a different Pareto optimal solution to give more feedback of what kind of Pareto optimal solutions are attainable. It is also possible to change the single objective solver used to solve subproblems related to the classification. These selections can be made with the tool bar on the upper most part of the user interface.

It should be noted that in this kind of a user interface the DM is not

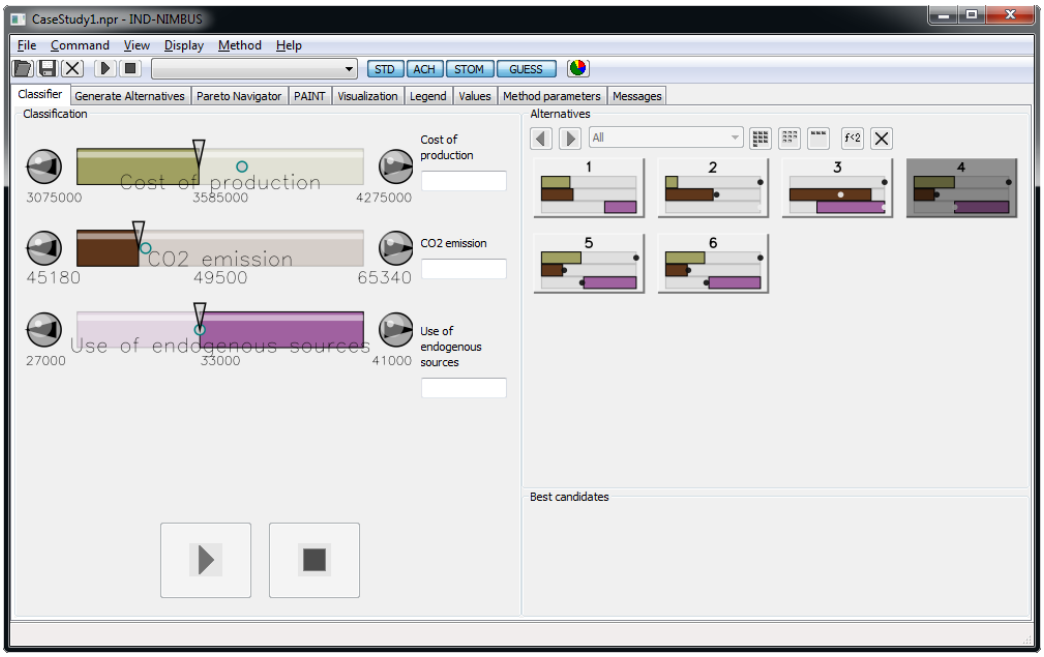

Figure 2: GAMS-NIMBUS Tool classification view

asked specifically to classify the functions. Instead, the interface hides the algorithmic details from the DM and silently converts the user input to a corresponding NIMBUS classification. Therefore, this interface can be used for other methods using similar preference information, e.g., reference points.

During the solution process, the DM has access to all solutions generated during the solution process. Therefore, any solution can be selected as a starting solution for a new classification when using the NIMBUS method, or as an end point for generating intermediate solutions between any two solutions, at any time. In this way, the DM can generate representations of interesting parts of the Pareto optimal set according to his or her preferences for further study.

5.3 IND-NIMBUS Software Framework

As mentioned, the GAMS-NIMBUS Tool uses the user interface implemented in the IND-NIMBUS software framework. IND-NIMBUS [27] is a cross platform desktop software framework intended to provide method developers with a tool-set that can be used for implementing different interactive multiobjective optimization methods. At the moment, the IND-NIMBUS software framework has been used to implement the NIMBUS method and prototypes

of the Pareto Navigator [9] and PAINT [17] methods.

In addition to different interactive multiobjective optimization methods, the IND-NIMBUS software framework contains integration modules, which can be used to connect interactive multiobjective optimization methods with various modeling and simulation tools. The computational model of the problem to be solved is assumed to be formulated with one of these tools. Examples of using such tools in various problem domains include the Matlab® environment for radiotherapy treatment planning [43, 44], the GPS-X™ wastewater treatment plant simulation and optimization application[14, 15], the Balas® process simulation software in chemical process design [11, 13], the Numerrin environment in optimal shape design [18], and almost any programming language [16, 23, 28, 38].

The IND-NIMBUS framework architecture follows the core structure and the structure described in Figure 1, as it consists of a separate user interface and an algorithm part. In the IND-NIMBUS framework, these parts are separated so that they can be easily changed independently. For developing the GAMS-NIMBUS Tool we have replaced the algorithm part of the IND-NIMBUS framework with GAMS models, implementing steps 1, 2 and 4 of the core structure. This enables the GAMS-NIMBUS Tool to use single objective solvers of the GAMS environment to solve scalarized subproblems realized with the GAMS modeling language, while utilizing the graphical user interface developed for the IND-NIMBUS software framework.

In addition to the user interface and algorithm parts, the IND-NIMBUS framework includes an automatic testing module that is used to verify the results generated in the algorithm part. This is achieved by solving a set multiobjective optimization problems with predefined preference information and verifying that the obtained Pareto optimal solutions correspond to Pareto optimal solutions known to be obtained form these preferences. The testing module is used to verify that methods included in the IND-NIMBUS framework are implemented correctly.

In order to collate the Pareto optimal solutions generated, the DM can create different filters, for example, to show only those solutions where the first objective function has values above a certain limit. There is also a possibility to view only those solutions generated in the last classification or solution generation. The DM can also store the best solution candidates in a specific list at any time during the solution process. Furthermore, undesirable solutions can be deleted from the solution list. The IND-NIMBUS framework contains also different graphical views for visualizing the solutions for comparison [29], and the solutions can be exported to be visualized with external software, such as Microsoft Excel.

6 Numerical Examples

To demonstrate the applicability of the GAMS-NIMBUS Tool and benefits of interactive multiobjective optimization methods in general, we solve two examples of GAMS problem models using the GAMS-NIMBUS Tool. As for the first example, we use the power generation problem [24] included in the GAMS model library. The second example is a multiobjective optimization version of the heat exchanger network synthesis problem, previously solved using a customized implementation of the NIMBUS method for the GAMS environment [21]. This problem is also available in the GAMS model library. The solutions tabulated along the examples can be reproduced by using the demonstration version of the GAMS-NIMBUS Tool, available at `http://ind-nimbus.it.jyu.fi/` as part of the IND-NIMBUS software framework.

Example 1: Power Generation Problem

In the first example the task is to determine the number of power generation units in a region. This simplified linear model has three objective functions: minimize *cost of production*, minimize CO_2 *emission* and maximize *use of endogenous sources*. The model is constrained by the power generation unit capacity of each type of power generator unit and three constraints set by power production demands [24]. The model can be found in the GAMS Model library as model EPSCM.

The EPSCM model contains instructions of how to solve this multiobjective optimization problem using the augmented ε-constraint method AUGMECON [25]. Here we first solve the model using the AUGMECON method and then with the GAMS-NIMBUS Tool. It should be noted that the AUGMECON and the NIMBUS methods solve a multiobjective optimization problem differently, and therefore provide different results. The AUGMECON method is an a posteriori method, providing the DM with a representative set of Pareto optimal solutions, from which he or she must select the solution he or she prefers the most. With an interactive multiobjective optimization method, such as the NIMBUS method, the DM can study various Pareto optimal solutions, which are generated based on his or her preference information, until he or she is satisfied with some solution.

The AUGMECON method could also be modified to act as an interactive method. As the interactive version of AUGMECON is not included in the GAMS Model library, we do not, as such, compare results obtained with the GAMS-NIMBUS Tool to the results obtained with the AUGMECON method. Instead, the EPSCM model is used as a conveniently available example to demonstrate how the GAMS-NIMBUS Tool can be used to apply

an interactive multiobjective optimization method to solve a multiobjective optimization problem in GAMS. It should be noted that interactive AUGMECON method could be implemented following the guidelines given in the Section 3.

Minimize	*Minimize*	*Maximize*
Cost of production	*CO_2 emission*	*Use of endogenous sources*
3075000.0	62460.0	31000.0
3078000.0	62316.0	31200.0
3099000.0	61308.0	32600.0
3111000.0	60732.0	33400.0
3120000.0	60300.0	34000.0
3141000.0	59292.0	35400.0
3147000.0	59004.0	35800.0
3162000.0	58284.0	36800.0
3183000.0	57276.0	38200.0
3204000.0	56268.0	39600.0
3219000.0	55548.0	40600.0
3225000.0	55260.0	41000.0
3315000.0	53820.0	39000.0
3423000.0	52092.0	36600.0
3531000.0	50364.0	34200.0
3639000.0	48636.0	31800.0
3747000.0	46908.0	29400.0
3855000.0	45180.0	27000.0

Table 1: Solutions obtained with the augmented ε-constraint method for Example 1

A set of 18 Pareto optimal solutions generated by the AUGMECON method is given in Table 1. Each row of Table 1 represents a Pareto optimal solution (objective vector) of the EPSCM model. As can be seen, the AUGMECON method has generated more or less evenly distributed solutions from the Pareto frontier. In contrast, the GAMS-NIMBUS Tool generates solutions only from the areas that are interesting to the DM, as is shown in Table 2. Table 2 summarizes the preference information provided and the solutions generated in each iteration of the solution process.

When using the GAMS-NIMBUS Tool, the user first opens the GAMS model, in this case the epscm.gms model file from the GAMS Model Library, to be found in a sub-directory of the GAMS installation directory. When opening the epscm.gms model file, the GAMS-NIMBUS Tool is able to find a single model, named *example* and three suitable expressions, *z('cost')*,

z('CO2emission') and *z('endogenous')*. These three expressions are selected as the objective functions. As *z('endogenous')* is an objective to be maximized, this is indicated by setting the type of the objective function to be maximized. The original constrains are used without modifications, so the multiobjective problem model is now completed and the GAMS-NIMBUS Tool can be used to solve Example 1.

To get started, the GAMS-NIMBUS Tool first provided the DM with information of the ranges of objective function values in the set of Pareto optimal solutions and a neutral compromise solution in the initialization stage. From this solution (objective function values given in Table 2), the DM decided that he would like to give an upper bound (denoted by $I^{\geq=}$ in the table) of 56000 to CO_2 *emission*, while maximizing (denoted by $I^{<}$) *use of endogenous sources*. The *Cost of production* is allowed to change freely (denoted by $I^{\diamond}$). As can be seen in iteration 2 of the table, with this preference information two new Pareto optimal solutions were generated and in one of them the *use of endogenous sources* has reached the maximum, i.e., ideal objective value, while CO_2 *emission* stayed below the desired bound. While looking at these solutions, the DM noticed that he might be able improve ($I^{\leq=}$) the CO_2 *emission* until only 50000 tons of CO_2 would be emitted, while allowing deterioration till ($I^{\geq=}$) 33000 tons of *endogenous sources*. This preference information was given in iteration 3, specifying that three new solutions are desired. The first of the Pareto optimal solutions was then selected as the final solution (shown in Table 2). The GAMS model used to generate solutions for iteration 2 can be seen in Appendix B.

It should be noted that each solution generated by NIMBUS is Pareto optimal [37], so even though the DM did not specify any preferences related to the first objective, *cost of production*, the cost is always as low as possible, in relation to the other objectives. When comparing to the solutions in Table 1, it can be seen that AUGMECON generated similar, but not the same solutions as the GAMS-NIMBUS Tool. However, with the GAMS-NIMBUS Tool the DM did not need to compare as many solutions simultaneously as with AUGMECON and he could direct the search with his preference information conveniently. Finally, much fewer solutions had to be generated when using the NIMBUS method.

Example 2: Heat Exchanger Network Synthesis

In the second example we demonstrate how the GAMS-NIMBUS Tool can be used to solve an optimization problem based on a more complicated GAMS model. We use the heat exchanger network synthesis (Synheat) problem, with the multiobjective problem formulation presented in [21]. The Syn-

Iter		*Minimize* *Cost of production*	*Minimize* CO_2 *emission*	*Maximize* *Use of endog sources*
	Ideal	3075000.0	45180.0	41000.0
	Nadir	4275000.0	65340.0	27000.0
1	Init. Sol.	3435000.0	51900.0	36333.33
2	Cur. Sol.	3435000.0	51900.0	36333.33
	Classif	$I^{\diamond}$	$I^{\geq=56000.0}$	$I^{<}$
		3225000.0	55260.0	41000.0
		3075000.0	62460.0	31000.0
3	Cur. Sol.	3225000.0	55260.0	41000.0
	Classif	$I^{\diamond}$	$I^{\leq=50000.0}$	$I^{\geq=33000.0}$
		3585000.0	49500.0	33000.0
		3574583.3	49666.67	33231.48
		3567986.8	49772.21	33378.07
	Final Sol.	**3585000.0**	**49500.0**	**33000.0**

Table 2: Solutions of the GAMS-NIMBUS Tool for Example 1

heat model is available in the GAMS Model library as model number 319, named SYNHEAT. Originally, the problem has been solved in [51] using the weighting method (optimizing the weighted sum of the objective functions). It should be noted that as the problem is nonconvex, and as we use a local solver due to computational demands of the model, some of the subproblems may generate only locally Pareto optimal solutions. However, even this is much better than the weighting method which may completely miss Pareto optimal solutions of nonconvex problems no matter how the weights are set. Additionally, such a straightforward conversions can lose information about interdependencies among the objectives which might affect the validity of the results obtained (see, e.g., [26, 47]).

The goal in the heat exchanger network synthesis problem is to minimize the total cost by simultaneously minimizing *number of heat exchanger units* (Units), *heat exchanger surface area* (Area) *cold utility consumption* (CU) and *hot utility consumption* (HU). In addition to these four objective functions, the Synheat model has seven sets of constraints totaling to almost seven hundred individual constraints in this example.

In the previous work [21], the NIMBUS method was implemented as a part of the Synheat model, where the NIMBUS subproblems were added to the original Synheat model by hand, combining the Synheat model and the NIMBUS implementation to a single Synheat-NIMBUS model. In the Synheat-NIMBUS model, scalarizations were handled by using different if-

Iter	*Issue*	*Minimize Units [-]*	*Minimize Area* $[m^2]$	*Minimize HU [kW]*	*Minimize CU [kW]*
	Ideal	4	2385.0	1756.0	1856.0
	Nadir	8	382781.0	7100.0	7200.0
1	Init. Sol.	6	258474.0	5354.0	5454.0
2	Cur. Sol.	6	258474.0	5354.0	5454.0
	Classif	$I^{\geq 7}$	$I^{<}$	$I^{<}$	$I^{\diamond}$
		7	14492.0	1933.0	2033.0
3	Cur. Sol.	7	14492.0	1933.0	2033.0
	Classif	$I^{=}$	$I^{<}$	$I^{\geq 2500}$	$I^{\diamond}$
		7	7475.0	2500.0	2600.0
4	Cur. Sol.	7	7475.0	2500.0	2600.0
	Classif	$I^{\leq 5}$	$I^{<}$	$I^{\geq 3000}$	$I^{\diamond}$
		5	7475.0	3000.0	3100.0
5	Cur. Sol.	5	7475.0	3000.0	3100.0
	Classif	$I^{=}$	$I^{\geq 8000}$	$I^{<}$	$I^{\diamond}$
		5	8000.0	2443.0	2543.0
	Final Sol.	**5**	**8000.0**	**2443.0**	**2543.0**

Table 3: Synheat-NIMBUS model [21] solutions for Example 2

then and loop structures. As constructing them manually is laborious and error-prone, the Synheat-NIMBUS model implemented the NIMBUS method only partially. For example, the results obtained with the original Synheat-NIMBUS model are only weakly Pareto optimal and the model implemented only one NIMBUS subproblem (and therefore could generate only one solution per iteration).

In contrast, when using the GAMS-NIMBUS Tool, the NIMBUS method implementation is contained in the algorithm part, separated from the original problem model, as described earlier. With this approach, it is possible to modify the Synheat model without modifying the subproblem models of the GAMS-NIMBUS Tool, and vice versa.

The results obtained with the Synheat-NIMBUS model [21] can be seen in Table 3 (collecting preference information specified and solutions generated in each iteration). In the first iteration, the DM was presented with the ideal and the nadir objective vector components, and a neutral compromise solution as a starting solution. Next, (iteration 2) the DM allowed the *Units* objective to increase up to 7. The DM also specified that objectives *Area* and *HU* should both be minimized as much as possible ($I^{<}$) and that the fourth objective was allowed to vary freely (as in all subsequent iterations,

as *CU* and *HU* are directly dependent on each other). In iteration 3, the DM decided that the objective *Units* was satisfactory at its current level (denoted by $I^{=}$), but *Area* should be improved. As a trade-off, the DM was willing to increase the objective *HU* up to 2500. Then in iteration 4 the DM decided to see if the number of units could be lowered, by allowing the other objectives to be increased. Results obtained were almost satisfactory, and after the classification in iteration 5 the DM was finally pleased with the solution obtained.

Iter	*Issue*	*Minimize Units [-]*	*Minimize Area* $[m^2]$	*Minimize HU [kW]*	*Minimize CU [kW]*
	Ideal	4	2385.0	1756.0	1856.0
	Nadir	8	382781.0	7100.0	7200.0
1	Init. Sol.	5	12962.15	2402.0	2502.0
2	Cur. Sol.	5	12962.15	2402.0	2502.0
	Classif	$I^{\geq=7}$	$I^{<}$	$I^{<}$	$I^{\diamond}$
		7	12962.15	2053.64	2153.64
		7	50277.34	1756.81	1856.81
		7	3706.8	3564.58	3664.58
3	Cur. Sol.	7	12962.15	2053.64	2153.64
	Classif	$I^{=}$	$I^{\leq=7400.0}$	$I^{\geq=3000.0}$	$I^{\diamond}$
		5	6496.83	2795.72	2895.72
4	Cur. Sol.	5	6496.83	2795.72	2895.72
	Classif	$I^{=}$	$I^{\geq=8000.0}$	$I^{<}$	$I^{\diamond}$
		5	8000.0	2457.43	2557.43
		7	51167.46	1756.67	1856.67
		9	51032.99	1755.5	1855.5
5	Cur. Sol.	5	6496.83	2795.72	2895.72
	Classif	$I^{=}$	$I^{\geq=7900.0}$	$I^{<}$	$I^{\diamond}$
		5	7900.0	2463.68	2563.68
		9	51032.99	1755.5	1855.5
		7	54862.92	1755.5	1855.5
	Final Sol.	**5**	**7900.0**	**2463.68**	**2563.68**

Table 4: GAMS-NIMBUS Tool solutions for Example 2

The solutions generated using the GAMS-NIMBUS Tool are shown in Table 4. When comparing these solutions to the previously obtained solutions, it is immediately apparent that the initial solution (iteration 1) in Table 3 is not Pareto optimal, and in effect, the first two iterations of the previous research did not provide the DM with correct information of the problem.

This is due to a mistake in the manually constructed subproblem model used to generate the initial solution in the Synheat-NIMBUS model. In addition, when using the GAMS-NIMBUS Tool to get results collected in Table 4, the DM decided to utilize the option of generating more than a single Pareto optimal solution for the same preference information at each iteration.

Otherwise, the solutions generated with the GAMS-NIMBUS Tool are similar to the previous solutions, even when starting from a different initial solution. As the DM had previously solved this problem, he had already some insight of the trade-offs between the objective functions and he had a preset notion of which kind of solutions he could obtain a desirable value for. Interestingly, by generating several Pareto optimal solutions per iteration, he was able to obtain *Area* with five units in iteration 3 without implicitly specifying this as a preference. In the following iterations he refined this solution, until obtaining the final solution, which he considered to be better than the final solution obtained in the previous research [21].

When comparing these two approaches, it its evident that by generating additional solutions for each given set of preference information the DM could conveniently obtain valuable information of trade-offs between objective functions with little additional effort. Importantly, the process to manually construct subproblem models proved out to be error prone. Instead, implementing NIMBUS by utilizing the distinction of roles in the core structure was a straightforward task. The same is valid for other methods that may be implemented within problem models in modeling environments. Furthermore, with the GAMS-NIMBUS Tool one could be more confident that the subproblem model formulations are constructed correctly, as it includes the testing module of the IND-NIMBUS framework.

Overall, with the GAMS-NIMBUS Tool the DM could conveniently direct the solution process, see Pareto optimal solutions that were interesting to him and gain insight into problems and trade-offs in them. The amount of information was limited in each iteration keeping the cognitive load small. Because of the learning that takes place during the solution process, the DM could be confident on the final solution obtained.

With the examples we demonstrated the advantages of interactive multiobjective optimization. In a similar way, solving other problems can benefit from a multiobjective optimization formulation and an interactive solution method.

7 Conclusions

In spite of promising results reported in the literature of using interactive multiobjective optimization methods, few implementations are available. We have introduced a core structure which is valid for many scalarization-based interactive multiobjective optimization methods and identified various roles to make method implementation a more straightforward task, in particular, in modeling environments where many mathematical models already exist ready to be solved with an interactive multiobjective optimization method. The core structure should facilitate wider availability of implementations of interactive multiobjective optimization methods as the main task is simply to express subproblems of the method in question as models of the modeling environment.

We have discussed how models available in modeling environments can be prepared for multiobjective optimization. As an example of the ideas presented and as a concrete example of an implementation prepared with the help of a core structure in a modeling environment we have introduced the GAMS-NIMBUS Tool, an implementation of the interactive NIMBUS method in GAMS. With the GAMS-NIMBUS Tool, it is possible to use the NIMBUS method to consider models containing several conflicting objectives expressed with the GAMS modeling language. A demonstration version of the tool is available at `http://ind-nimbus.it.jyu.fi/`, including examples reported in this research.

As mentioned, as an example of modeling environments we have considered the GAMS environment providing a common platform for optimization solvers, capable of solving a wide array of optimization problems. Previously, the GAMS environment has not been widely used to solve multiobjective optimization problems; specifically using interactive multiobjective optimization methods has been nonexistent. In this research, we have demonstrated the steps needed for implementing interactive multiobjective optimization for the GAMS environment. This enables benefiting from the advantages of multiobjective optimization and interactive methods when solving optimization problems expressed as GAMS models. On the other hand, when utilizing scalarization-based interactive methods, working in the GAMS environment gives access to many single objective solvers so that the solver best suited for the characteristics of the problem at hand can be selected among GAMS solvers. This makes many interactive methods more efficient than with standard solvers. The implementation ideas presented are valid for other modeling environments like AMPL or AIMMS as well.

The GAMS-NIMBUS Tool introduced is based on the IND-NIMBUS software framework, developed at the University of Jyväskylä. The IND-

NIMBUS software framework contains tools implementing and testing interactive optimization methods, which can be utilized also by the GAMS-NIMBUS Tool. Currently, there are several new interactive methods being implementated for the IND-NIMBUS framework, such as Pareto Navigator [9], PAINT [17] and Nautilus [30], and in the future, the GAMS-NIMBUS Tool can be used to apply these methods along with the NIMBUS method.

Acknowledgments

The research was partly supported by the Academy of Finland, grant number 128495 (on the part of Vesa Ojalehto).

References

[1] 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.

[2] 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.

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

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

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

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

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

[8] K. Deb, K. Miettinen, and S. Chaudhuri. Toward 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.

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

[10] R. Fourer, D. M. Gay, and B. W. Kerninghan. *AMPL: A Modeling Language for Mathematical Programming*. Boyd & Fraser Publishing Company, 1993.

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

[12] 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.

[13] 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.

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

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

[16] 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*, 116:265–281, 2003.

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

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

[19] P. V. Hentenryck. *The OPL Optimization Programming Language*. MIT Press, Cambridge, Massachusetts, 1999.

[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] T. Laukkanen, T.-M. Tveit, V. Ojalehto, K. Miettinen, and C.-J. Fogelholm. An interactive multi-objective approach to heat exchanger network synthesis. *Computers & Chemical Engineering*, 34(6):943–952, 2010.

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

[23] 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.

[24] 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.

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

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

[27] 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.

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

[29] K. Miettinen. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, to appear, DOI:10.1007/s00291-012-0297-0.

[30] K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque. NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research*, 206(2):426–434, 2010.

[31] 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.

[32] K. Miettinen and K. Kaario. Comparing graphic and symbolic classification in interactive multiobjective optimization. *Journal of Multi-Criteria Decision Analysis*, 12(6):331–335, 2003.

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

[34] K. Miettinen and M. M. Mäkelä. Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation. *Journal of the Operational Research Society*, 50:949–959, 1999.

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

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

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

[38] 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.

[39] 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, Berlin, 2008.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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 Berlin Heidelberg, 2010.

[45] 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.

[46] 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.

[47] R. E. Steuer. *Multiple Criteria Optimization; Theory, Computation, and Application.* John Wiley & Sons, Ltd., New York, 1986.

[48] 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.

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

[50] A. Wierzbicki. Reference point approaches. In T. Gal, T. Stewart, and T. Hanne, editors, *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*, pages 9–9–9–39. Kluwer Academic Publishers, Boston, 1999.

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

A Synchronous NIMBUS method

The NIMBUS method is an interactive multiobjective optimization method (see [26, 33, 34, 35, 37]) that has been used in solving several real-world problems. Among others, it has been applied in optimal shape design of ultrasonic transducers [18], in designing a paper machine headbox [16], optimal control in continuous casting of steel [28, 38], problems in chemical engineering [11, 13, 31], separation of glucose and fructose [12], intensity modulated radiotherapy treatment planning [43], brachytherapy [45], wastewater treatment plant simulation and optimization application [14, 15] and optimizing heat exchanger network synthesis [21].

The idea of the NIMBUS method is that the DM examines the values of the objective functions calculated at the current Pareto optimal decision vector $\mathbf{x}^c$ and classifies objective functions into up to five classes. This means that the DM is asked to indicate (by means of a classification) what kind of a solution would be more satisfactory than the current one. The classes are functions f_i (to be minimized) 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 $i \in I^\diamond$.

Classification must be done so that at least one objective function value should be improved, and at least one is allowed to impair. Otherwise, the method cannot generate a new Pareto optimal solution as there does not exist new Pareto optimal solution whose objective function values could be better than those of the Pareto optimal solution shown to the DM.

In the synchronous NIMBUS method [37], the preference information expressed as a classification with the corresponding aspiration levels and bounds is used to formulate from one to four different singe objective subproblems. These subproblems can then be solved with an appropriate single objective solver, producing new Pareto optimal solutions [37]. It should be

noted that by using different subproblem formulations, it is possible to generate somewhat different Pareto optimal solutions from the same preference information [36] obeying the preference information in somewhat different ways giving the DM more information about what kind of Pareto optimal solutions are available.

Originally in [37], NIMBUS subproblems involved single objective min-max functions. They are nondifferentiable and must be solved by an appropriate single objective solver. To avoid introducing nondifferentiability to the problem to be solved, we modify the original nondifferentiable subproblems to their differentiable equivalents, by adding a new decision variable α to the problem, and treating the min-max functions as constraints. Avoiding nondifferentiability in this way is naturally possible only if all functions of the original problem are differentiable.

As an example, the differentiable version of one of the NIMBUS subproblems is

$$\begin{array}{ll} \text{minimize} & \alpha + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} & \frac{f_i(\mathbf{x}) - z_i^{\star}}{z_i^{\text{nad}} - z_i^{\star\star}} \leq \alpha \text{ for all } \quad i \in I^{<}, \\ & \frac{f_j(\mathbf{x}) - \hat{z}_j}{z_j^{\text{nad}} - z_j^{\star\star}} \leq \alpha \text{ for all } \quad j \in I^{\leq}, \\ & f_i(\mathbf{x}) \leq f_i(\mathbf{x}^c) \text{ for all } \quad i \in I^{<} \cup I^{\leq} \cup I^{=}, \\ & f_i(\mathbf{x}) \leq \varepsilon_i \text{ for all } \quad i \in I^{\geq}, \\ & \mathbf{x} \in S. \end{array} \tag{3}$$

The synchronous NIMBUS method has three other subproblems, but as their differentiable formulations are quite straightforward to construct, they are not presented here.

In addition to the classification, the DM can ask new Pareto optimal solutions to be generated as intermediate solutions between any two existing Pareto optimal solutions. This is done by taking steps of equal length in the objective function space between two Pareto optimal solutions, and giving these as reference points ($\hat{z}$) to a differentiable counterpart of problem (2)

$$\begin{array}{ll} \text{minimize} & \alpha + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\text{nad}} - z_i^{\star\star}} \\ \text{subject to} & \frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\text{nad}} - z_i^{\star\star}} \leq \alpha \text{ for all } i = 1, \ldots, k \\ & \mathbf{x} \in S. \end{array} \tag{4}$$

The solution process is terminated if the DM does not want to improve any objective function value or is not willing to let any objective value impair. Otherwise, the search continues iteratively by moving around the Pareto optimal set. In this way, the DM can learn about the problem, adjust one's

hopes and finally identify the most desirable solution. It should be noted that when using the NIMBUS method, the DM is shown and asked to classify the actual objective function values and no artificial concepts requiring cognitive mapping between different concepts are used. For more details of the NIMBUS method, see [37].

B GAMS-NIMBUS Tool model listing

Listing 1 shows the GAMS model used to generate solutions for iteration 2 of Table 2 without the equations specific to the original problem model and commands used to pass new Pareto optimal solution values to the IND-NIMBUS software framework. The listing has four scalarization models of the GAMS-NIMBUS tool, corresponding to the scalarized subproblems used by the NIMBUS method (see e.g. problems (3) and (4)). In Table 2 there are only three new solutions shown for iteration 2, as two of the subproblems provided similar results.

Listing 1: Iteration 2 of Example 1

```
* Listing shows the GAMS model used to generate solutions for iteration 2
* of Table 2: "Solutions of the GAMS-NIMBUS Tool for Example 1" the
* equations specific to the original problem model but without commands
* used to pass new Pareto optimal solution values to the GAMS-NIMBUS Tool.
* The listing has four scalarization models corresponding to the
* scalarized subproblems used by the NIMBUS method (see e.g.\ subproblems
* (3) and (4). In the Table 2 there are only two new solutions shown for
* iteration 2, as two of the scalarization models did not provide new
* Pareto optimal solutions, therefore disregarded

$STitle Set NIMBUS Parameters
Parameter DUTO Difference of ideal and utopian values
          / 0.001 /;
Parameter RAUGMT Coefficient of the augmentation term
          / 0.001 /;

$STitle Multiojective Problem Model

* Model Example automatically generated from the epscm.319 model included in the
* GAMS Model Library by
*
* Mavrotas, G, Generation of efficient solutions in Multiobjective Mathematical
* Programming problems using GAMS. Tech. rep., School of Chemical Engineering,
* National Technical University of Athens, 2006.

$inlinecom [ ]
$eolcom //


Sets
   p        power generation units / Lignite, Oil, Gas, RES /
   i        load areas / base, middle, peak /
   pi(p,i)  availability of unit for load types
            / Lignite.(base,middle), Oil.(middle,peak), Gas.set.i, RES.(base, peak) /
   es(p)    endogenous sources / Lignite, RES /
   k        objective functions / cost, CO2emission, endogenous /

$set min -1
$set max +1
Parameter dir(k) direction of the objective functions
   / cost %min%, CO2emission %min%, endogenous %max% /;

Set pheader / capacity, cost, CO2emission /;
Table pdata(pheader,p)
                      Lignite          Oil           Gas          RES
capacity [GWh]          31000        15000         22000        10000
```

```
cost [$/MWh]                30          75          60          90
CO2emission [t/MWh]       1.44        0.72        0.45           0;

Parameter
   ad          annual demand in GWh / 64000 /
   df(i)       demand fraction for load type / base 0.6, middle 0.3, peak 0.1 /
   demand(i) demand for load type in GWh; demand(i) = ad*df(i);

Variables
   z(k)        objective function variables
Positive Variables
   x(p,i)      production level of unit in load area in GWh
Equations
   objcost     objective for minimizing cost in K$
   objco2      objective for minimizing CO2 emissions in Kt
   objes       objective for maximizing endogenous sources in GWh
   defcap(p) capacity constraint
   defdem(i) demand satisfaction
;

* Objective functions
objcost.. sum(pi(p,i), pdata('cost',p)*x(pi)) =e= z('cost');
objco2..  sum(pi(p,i), pdata('CO2emission',p)*x(pi)) =e= z('CO2emission');
objes..   sum(pi(es,i), x(pi)) =e= z('endogenous');

defcap(p).. sum(pi(p,i), x(pi)) =l=  pdata('capacity',p);
defdem(i).. sum(pi(p,i), x(pi)) =g=  demand(i);

Model example / all /;


Set
  njob         Number of jobs /1*4/
  nobj         Number of objective functions /1*3/
  nim_nf       Number of functions / f1,f2,f3/
;

Parameter
   nim_job_obj(njob,nobj)  Results for each job
   nimgams_ideal(nobj)             Ideal Vector
   nimgams_nadir(nobj)             Nadir Vector
   nimgams_utopia(nobj)            Utopia Vector
   nimgams_aspir(nobj)               Aspiration level
   ref(nobj)                 Reference point
   fvalue(nobj)              Current point
   nimclass(nobj)           NIMBUS Scalarization classes
Variables
   nim_obj(nobj)
   alpha
   a_objval      auxiliary variable for the objective function
   GN_obj        auxiliary variable during the construction of scalarization functions

Equations
nim_obj_f1
nim_obj_f2
nim_obj_f3
;

nim_obj_f1.. nim_obj('1') =e= z('cost');
nim_obj_f2.. nim_obj('2') =e= z('CO2emission');
nim_obj_f3.. nim_obj('3') =e= -1.0* z('endogenous');
;
Model NIMBUS / Example ,nim_obj_f1,nim_obj_f2,nim_obj_f3/;

nimgams_nadir('1') = 4275000.0;
nimgams_ideal('1') = 3075000.0;
nimgams_nadir('2') = 65340.0;
nimgams_ideal('2') = 45180.0;
nimgams_nadir('3') = -27000.0;
nimgams_ideal('3') = -41000.0;
loop (nobj, nimgams_utopia(nobj)   = nimgams_ideal(nobj) - DUTO);

$STitle Four Scalarization Models
* Preference information obtained from the DM
* These values are changed for each iteration
ref('1')=3435000.00;
ref('2')=51900.00;
ref('3')=-36333.3300;

fvalue('1')=3435000.00;
fvalue('2')=51900.00;
fvalue('3')=-36333.3300;
```

```
$STitle NIMBUS scalarization model
Equations
   nim_nim
   nim_nim_con1
   nim_nim_con2
   nim_nim_con3
   nim_nim_con4
;
   nim_nim..   alpha + RAUGMT * sum(nobj,  nim_obj(nobj) /
                    ( nimgams_nadir(nobj)-nimgams_utopia(nobj) ) ) =e= GN_obj;
   nim_nim_con1.. (nim_obj('2') - ( 56000.0) ) /
                  ( nimgams_nadir('2')-nimgams_utopia('2') )         =l= alpha;
   nim_nim_con2.. nim_obj('2') =l= fvalue('2');
   nim_nim_con3.. (nim_obj('3') - nimgams_ideal('3') ) /
                                  ( nimgams_nadir('3')-nimgams_utopia('3') ) =l= alpha;
   nim_nim_con4.. nim_obj('3') =l= fvalue('3');
model NIM  / NIMBUS, nim_nim, nim_nim_con1,nim_nim_con2,nim_nim_con3,nim_nim_con4 /;

  NIM.optfile=1;
* Solve NIMBUS scalarization model
  solve NIM using nlp minimizing GN_obj;
display 'Objectives:', nim_obj.l;

* Use different reference point for reference point based scalarizations
ref('1') = 4275000.00;
ref('2') = 56000.0;
ref('3') = -41000.000;

$STitle Achievement scalarization model
Equations
   nim_ach
   nim_ach_con(nobj)
;

nim_ach..   alpha + RAUGMT * sum(nobj,  nim_obj(nobj) /
                  ( nimgams_nadir(nobj)-nimgams_utopia(nobj) ) ) =e= GN_obj;
nim_ach_con(nobj).. (nim_obj(nobj) - ref(nobj) ) /
                    ( nimgams_nadir(nobj)-nimgams_utopia(nobj) ) =l= alpha;


model ACH  / NIMBUS, nim_ach, nim_ach_con /;


  ACH.optfile=1;
* solve achievement scalarization model
  solve ACH using nlp minimizing GN_obj;
display 'Objectives:', nim_obj.l;

$STitle Smooth STOM scalarization model
Equations
   nim_stom
   nim_stom_con(nobj)
;

nim_stom..   alpha + RAUGMT * sum(nobj, nim_obj(nobj) /
                   ( ref(nobj)-nimgams_utopia(nobj) ) ) =e= GN_obj;
nim_stom_con(nobj).. (nim_obj(nobj) - nimgams_utopia(nobj)) /
                     ( ref(nobj)-nimgams_utopia(nobj) ) =l= alpha;

model STOM   / NIMBUS, nim_stom, nim_stom_con /;


  STOM.optfile=1;
* Solve STOM scalarization model
  solve STOM using nlp minimizing GN_obj;
display 'Objectives:', nim_obj.l;

$STitle Smooth GUESS scalarization model
Equations
   nim_guess
   nim_guess_con(nobj)
;

nim_guess..   alpha + RAUGMT * sum(nobj, nim_obj(nobj) /
                    ( nimgams_nadir(nobj)+DUTO-ref(nobj) ) ) =e= GN_obj;
nim_guess_con(nobj).. (nim_obj(nobj) - nimgams_nadir(nobj)) /
                      ( nimgams_nadir(nobj)+DUTO-ref(nobj) ) =l= alpha;

model GUESS  / NIMBUS, nim_guess, nim_guess_con /;


  GUESS.optfile=1;
```

```
*Solve GUESS scalarization model
  solve GUESS using nlp minimizing GN_obj;

display 'Objectives:', nim_obj.l;
```