DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

Sequential versus simultaneous application of multi-objective optimization and multicriteria decision making: an empirical investigation

Kenneth Sörensen, Johan Springael & Sylvie Busschaert

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

EARNED EXCELLENCE
AACSB ACCREDITED
EST. 1916
THE BEST BUSINESS SCHOOLS IN THE WORLD

Stadscampus
Prinsstraat 13, B.226
BE-2000 Antwerpen
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
http://www.ua.ac.be/tew

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

Sequential versus simultaneous application of multi-objective optimization and multicriteria decision making: an empirical investigation

Kenneth Sörensen, Johan Springael & Sylvie Busschaert

RESEARCH PAPER 2010-023
OCTOBER 2010

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@ua.ac.be

The papers can be also found at our website:
www.ua.ac.be/tew (research > working papers) &
www.repec.org/ (Research papers in economics - REPEC)

D/2010/1169/023

Sequential versus simultaneous application of multi-objective optimization and multicriteria decision making: an empirical investigation

Kenneth Sörensen Johan Springael
Sylvie Busschaert
University of Antwerp, Faculty of Applied Economics

October 2010

Abstract

The multi-objective optimization paradigm prescribes that a multi-objective optimization problem should be solved in two steps executed in sequence. First an approximation of the Pareto set is determined, that contains as many non-dominated solutions as possible. Secondly, a solution is chosen among these Pareto-optimal solutions. Although a large majority of papers on multi-objective optimization focuses exclusively on the first step, the second step is equally important: a decision maker generally can only implement a single solution and will need a way to select one according to his preferences.

In this paper, we empirically test the soundness of the sequential approach to multi-objective optimization and provide convincing evidence that it can be outperformed by a simultaneous approach, in which the decision maker's preferences are taken into account *during* the multi-objective optimization. To this end, we develop a simple tabu search algorithm for the multi-objective knapsack problem and combine it with the PROMETHEE multicriteria decision making method, both sequentially and simultaneously. The results of both approaches are compared both in terms of computing times and solution quality. The simultaneous approach is shown to strongly outperform the sequential one.

1. Multi-objective optimization and multicriteria decision making

Many real-life optimization problems have multiple, often conflicting, objectives. In vehicle routing for example, it may be appropriate to simultaneously minimize the total distance traveled, minimize the number of vehicles used, and minimize the difference between the duration of the longest and the shortest trip, i.e. balance the routes. Given the fact that these *multi-objective optimization* problems are in most cases computationally difficult, metaheuristics are often used to solve them. Such *multi-objective metaheuristics* (MOMH) generally generate a large number of solutions, from which the decision maker is left to choose

one. For choosing between different alternatives, *multicriteria decision making* (MCDM) methods are used.

The solution to an optimization problem can be represented as a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ in the *solution space* $\boldsymbol{X}$, the set of all solutions. In a multi-objective optimization problem, rather than assigning a single real number to each solution, a function $\boldsymbol{f} : \boldsymbol{X} \to \boldsymbol{Y}$ maps any solution onto a k-dimensional vector $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) = (y_1, y_2, \ldots, y_k)$ in the objective space.

Since multi-objective optimization problems have more than one objective, the notion of optimality needs to be abandoned. It is generally replaced with the notion of *dominance*. If we assume that all $Y \subseteq \mathbb{R}^k$ and (without loss of generality) that $f_1, f_2, \ldots, f_k$ should all be maximized, then an objective vector $\boldsymbol{y}^1$ is said to *dominate* a vector $\boldsymbol{y}^2$ if no component of $\boldsymbol{y}^1$ is smaller than the corresponding component of $\boldsymbol{y}^2$ and at least one component is greater. We denote this dominance relationship with $\boldsymbol{y}^1 \succ \boldsymbol{y}^2$. A solution $\boldsymbol{x}^1$ is said to dominate another solution $\boldsymbol{x}^2$ if $\boldsymbol{f}(\boldsymbol{x}^1) \succ \boldsymbol{f}(\boldsymbol{x}^2)$. In multi-objective optimization, the set of non-dominated solutions is called the *Pareto set* and the projection of this set onto the objective function space is called the *Pareto front*.

While single-objective optimization problems have a clear aim of finding the solution with the largest or smallest objective function value, i.e. the optimal solution, the aim of MOMH is much more difficult to define formally. It is however generally accepted that the aim of any multi-objective algorithm should be to *find all solutions in the Pareto set*. Since the computational complexity of most multi-objective optimization problems renders this aim intractable, it is usually relaxed to *approximate the Pareto front as well as possible* [28]. The outcome of any multi-objective algorithm is therefore generally a set of mutually non-dominated solutions, the *Pareto set approximation*. To measure the quality of such an approximation, many different measures exist [15] that essentially measure one or more of the following criteria:

- Objective function vectors of solutions in the set should be as close as possible to objective function vectors of solutions in the Pareto set (which are usually not known).
- There should be as many solutions in the set as possible.
- Solutions should cover the entire Pareto front and be evenly spaced.

A question left unanswered in most papers on multi-objective optimization, is what a decision maker should do with such a Pareto set approximation generated by a multi-objective algorithm. Such sets can easily contain thousands of solutions, of which usually only one can be implemented. Implicitly, it is assumed that a decision maker will be able to choose a solution from this set that satisfies his preferences in the best possible way. In other words, the multi-objective optimization literature assumes that an *a posteriori* approach will be used, in which the decision maker uses some form of MCDM method to model his preferences and chooses a solution based on the results of this method, from the set of solutions generated using the MOMH. Many methods for MCDM have been proposed, some of the more prominent ones being AHP [26], PROMETHEE [4] and ELECTRE [25]. MCDM methods generally use both between-criteria and within-criteria information to model a decision maker's

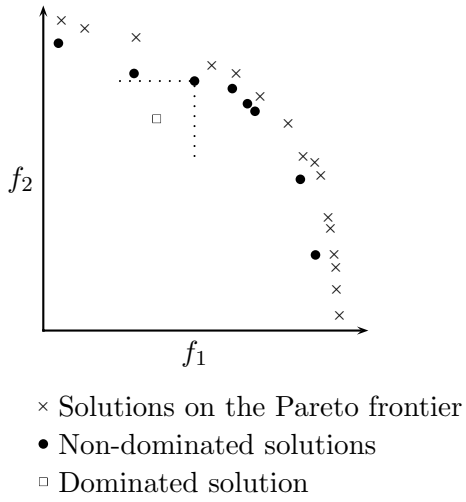


Figure 1: Multi-objective optimization — the aim of the optimization procedure is to approximate the Pareto frontier

preferences. Using this information, the different alternatives are compared and ordered, so that the most appropriate one can be determined.

However, MCDM methods were not designed with the output of a typical MOMH in mind, and are generally only applied to choose between a handful of solutions. As the goal of such a metaheuristic generally includes generating as many non-dominated solutions as possible (notwithstanding the fact that some methods have been proposed to limit the number of solution reported by the MOMH [e.g., 18, 20]), this raises some questions with respect to the potential usefulness of the "MOMH-followed-by-MCDM" approach. Two alternatives to the a posteriori approach do exist. In the *interactive* approach, the decision maker is supposed to make some additional decisions during the optimization process, revealing his preferences. In the *a priori* approach, the decision maker is asked to state his preferences before optimization starts.

In this research, we investigate experimentally whether MOMH and MCDM should be applied sequentially, as proposed in the MOMH literature, or instead be integrated into a single method and hence applied simultaneously. We test in a controlled setting which of the two methods performs best on the *multi-objective knapsack problem*, a generalization of the single-objective knapsack problem in which each item has several profit values. To this end, we develop a simple *tabu search* MOMH for this problem and combine this method with the well-known PROMETHEE MCDM method. In the *sequential* setting, the PROMETHEE method is applied to the set of Pareto-optimal solutions generated by the tabu search metaheuristic. In the *simultaneous* setting, we integrate the PROMETHEE method into the tabu search metaheuristic. The resulting method searches for a single solution without generating the Pareto-set first. For comparison purposes, we also implement the simplest way to avoid generating the Pareto frontier: the MAUT (multi-attribute utility theory) method transforms the problem into a single-objective one by using a weighted average of the objective functions. Essentially, the sequential method is an a posteriori method, as the preferences are only needed after the MOMH method has finished. The MAUT and

simultaneous methods on the other hand are a priori methods as the decision maker's preferences need to be known in advance.

The simultaneous, MAUT and sequential options are compared by applying them to a set of benchmark instances of the multi-objective knapsack problem, making sure that as many components as possible are the same in both solution techniques. The methods are compared both with respect to the quality of the solutions they produce and the computing times they require and results are reported.

2. Integration of user preferences in MOMH: literature review

Although the literature on MOMH and MCDM has grown considerably over the last years, the combination of user preferences and MOMH has received a lot less attention. In this section contains the most important research efforts on this subject.

Generally, a sequence-based classification is used to classify the different approaches. As explained earlier, we can make a distinction between a posteriori, interactive and a priori methods [27]. Alternatively, we can also classify the different approaches according to the way in which the decision maker articulates his preferences. In the literature, we find examples of preferences being represented as solution attributes [2, 11], trade-offs between objectives [1], optimization goals [10, 12, 13], outranking relations [5, 19, 24], fuzzy preferences [7–9, 16] or utility functions [14].

- **Solution attributes**
  This type of approach allows to focus the search process on the solutions of the Pareto-front that comply the desired solution features defined by the decision maker. According to Deb and Gupta [11], no decision maker is interested in finding the whole Pareto-front for any given optimization problem. Instead, the search of the solution space should be guided towards the *robust* non-dominated solutions, i.e. solutions which are almost insensitive to small variable perturbations. Branke et al. [2] for their part argue that only the *knees* of the Pareto-front are of importance to the decision maker. The knee solutions are those from which there is not much motivation for a decision maker to move out, as by definition one unit of improvement in one objective from a knee point requires a large sacrifice in other objectives.

- **Trade-offs between objectives**
  As defined by Rachmawati and Srinivasan [23], a trade-off specifies the amount of improvement in one or more objective(s) which the decision maker is ready to relinquish to attain a unit improvement in another objective. Although it is hard for decision makers to define the objectives' weights before optimization has started, Branke et al. [1] argue that they must however have some idea about what range of weighting is reasonable. So, instead of asking to define crisp, single valued weights, they propose to implement the preferences of the decision maker as linear maximum and minimum trade-off functions in an evolutionary metaheuristic. Since this information is used to modify the definition of dominance, it is possible to guide the search process to any part of the Pareto-front that corresponds to the predefined trade-offs.

- **Goal Specification**
  In the case of goal specification, the decision maker defines aspiration levels $\overline{z}_i$ for each objective function f_i, which define the goals $f_i(x) \leq \overline{z}_i$. The idea to use goal programming to incorporate preferences in multiobjective optimization was first formulated by Fonseca and Fleming [12]. They reduce the Pareto-front by allowing a deterioration in vector components that have already attained their goal in favour of components for which this is not yet the case. They introduce a MOGA (Multiple Objective Genetic Algorithm) in which the goal specification is performed in a progressive way. However, in 1998, Fonseca and Fleming [13] introduce a new approach allowing the decision maker to specify objective priorities. Alternative solutions are first compared pairwise in terms of the objectives with the highest priority while neglecting objectives of lower priorities. If both solutions meet all goal values or if they both violate the goals in the same way, objectives of lower priority classes are considered. Finally, also Deb [10] make use of goal programming theory. However, instead of comparing the actual criteria, they compare the distances from the goals.

- **Outranking relations**
  The modelling of preferences as binary outranking relations is common in MCDM. Outranking relations can be either crisp or fuzzy and result from the pairwise comparison of alternatives and/or objectives. Although outranking approaches regroup a wide variety of techniques, we can only find examples of PROMETHEE II [4] in the MOMH literature. Rekiek et al. [24] use PROMETHEE II interactively with a genetic algorithm to solve the multiobjective problem of assigning tasks to stations and selecting assembly equipment for each station in hybrid assembly lines. Coelho et al. [5] on their part introduce a new optimization method called PAMUC — Preferences Applied to MUltiobjectivity and Constraints — for mechanical structures design. Multiple objectives and constraints are jointly tackled by implementing PROMETHEE II in an evolutionary algorithm using an adaptive weighing scheme. Finally, Massebeuf et al. [19] propose an a posteriori incorporation of PROMETHEE II in a diploid genetic algorithm.

- **Fuzzy preferences**
  It is commonly agreed on that, although human beings are qualitatively strong, they perform a lot worse when it comes to providing exact quantitative numbers. Fuzzy preferences take into account this human limitation by allowing decision makers to express their preferences on an ordinal scale. Cvetkovic and Parmee [7, 8] introduce a new method for transforming these fuzzy preferences into crisp ones, which are afterwards implemented in both weighted-sum optimization or Pareto-optimization evolutionary metaheuristics. However, since the conversion of fuzzy preferences into single-valued crisp weights is not always consistent, Jin and Sendhoff [16] propose to use real-valued weight intervals instead. These weight intervals are combined with the evolutionary dynamic weighted aggregation to construct the preferred Pareto-front.

- **Utility functions**
  MAUT or Multi Attribute Utility Theory is one of the prominent MCDM-methods in practice. It belongs belongs to the class of aggregation techniques. These techniques transform the multi-objective problem into a single-objective one by aggregating the multiple objectives into one 'synthesized' objective. In the specific case of MAUT, this

aggregation is done by using a utility function. Greenwood et al. [14] use elements of ISMAUT — Imprecisely Specified MAUT — in an evolutionary algorithm to perform an imprecise ranking of attributes. The idea is to rank a set of solutions instead of the objectives (which is done implicitly). Preference information is also used in the survival scheme of the algorithm.

Given the large diversity in techniques used for modelling the decision maker's preferences, the lack of diversity found in techniques used for multi-objective optimization is rather surprising. Because of their ability to generate multiple non-dominated solutions in one simulation run [12, 13, 21], genetic and evolutionary multi-objective techniques are by far the most popular metaheuristics for multi-objective optimization. As a consequence, two review papers have been devoted solely to the incorporation of decision making in evolutionary algorithms [6, 23], and very few authors recognize the benefit of using alternative metaheuristics such as local search techniques [17].

Although the multi-objective optimization paradigm implicitly assumes an a posteriori strategy, we can find some examples of the a priori/simultaneous approach in the previous enumeration. However, a question which remains unanswered is whether there is reason to do so. First of all, these methods require that the decision maker's preferences are measured before the optimization starts, which is not always possible. Additionally, they are inherently more complicated than their sequential counterparts. Therefore, a simultaneous approach only makes sense if this yields better solutions, i.e. if the approach is able to locate solutions that are preferred over the solutions that are found using the sequential approach. Answering this question is the main aim of this paper. To this end we develop a simple tabu search method for the multi-objective knapsack problem in section 3. In section 5 this method is combined with the PROMETHEE method (described in section 4) according to three different schemes: sequential, MAUT and integrated. The performance of these three schemes is compared in section 6. Section 7 concludes and gives some pointers for future research.

3. A simple tabu search metaheuristic for the multi-objective knapsack problem

The (single-objective) knapsack problem is one of the best-known combinatorial optimization problems. Given n items, each with a certain cost c_i, $i \in [1, n]$ and a certain value or profit p_i, the objective is to select a subset of items of which the total profit is maximal, while having a cost below a certain threshold, called the knapsack capacity C.

The *multi-objective* knapsack problem is a simple extension of the single-objective knapsack problem, in which each item has j profits, and the objective is to "maximize" all m objectives simultaneously. In this problem, p_{ij} represents the profit of item i in objective function j.

$$\text{“maximize”} \quad f_j(\boldsymbol{x}) = \sum_{i=1}^{n} p_{ij} x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} c_i x_i \leq C \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

Without loss of generality, we assume in the rest of this paper that all objective functions need to be maximized.

The MOMH methods described in section 5 share a common underlying tabu search metaheuristic, that they use in different ways to find good solutions. This simple procedure is a local search metaheuristic in that it iteratively improves the quality of a solution by adding or removing one item at a time.

The tabu search procedure starts by sorting all items in decreasing order of *value*. The value of an item is a weighted sum of its different objectives (profits), divided by its cost. The (current) weight vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_m\}$ determines the weight of each profit function and determines the search direction in the objective function space. What sets the different MOMH methods of section 5 apart is the way in which they determine the value of vector $\boldsymbol{\alpha}$ at each iteration.

Using this weight vector, the value of item i is calculated as $v_i = \sum_{j=1}^{m} p_{ij}\alpha_j / c_i$. At each step, the tabu search procedure either adds the item that is not tabu and has the largest value or removes the item that is not tabu and has the smallest value. An item is only removed if no items can be added without exceeding the capacity. At each point in the tabu search procedure, the solution therefore remains feasible.

When an item is added or removed, its index number is added to the tabu list. The tabu list has a fixed tenure (length) t and therefore always contains the t most recently added or removed items. Items that appear on the tabu list cannot be added or removed.

The tabu search algorithm also features a perturbation move. This move modifies the solution by changing the status of each item with a fixed probability P_{perturb}. Items that are absent in the solution are only added to the solution if this does not violate the capacity constraint. The tabu search continues until a fixed number of iterations $N_{\max}$ without improvement is registered. At the end of a tabu search run, the solution with the highest weighted objective function value is reported.

The tabu search algorithm requires only three parameters to be set: the tabu tenure t, the probability to change the status of each item in the perturbation move P_{perturb} and the maximum number of iterations without improvement $N_{\max}$.

4. The PROMETHEE multicriteria method

In the PROMETHEE multicriteria method the starting point is an evaluation table (see Table 1), that contains for each alternative (solution) a_k its score on each criterion (objective) f_j.

Table 1: Evaluation table with scores of the alternatives on the different criteria

	$f_1(\cdot)$	$\dots$	$f_m(\cdot)$
a_1	$f_1(a_1)$	$\dots$	$f_m(a_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_K	$f_1(a_K)$	$\dots$	$f_m(a_K)$

These evaluations are used in the PROMETHEE method to calculate all pairwise differences between any two alternatives k and l on each criterion:

$$d_j(a_k, a_l) = f_j(a_k) - f_j(a_l),\ \forall j = 1, \dots, m, \forall k, l = 1, \dots, K \tag{4}$$

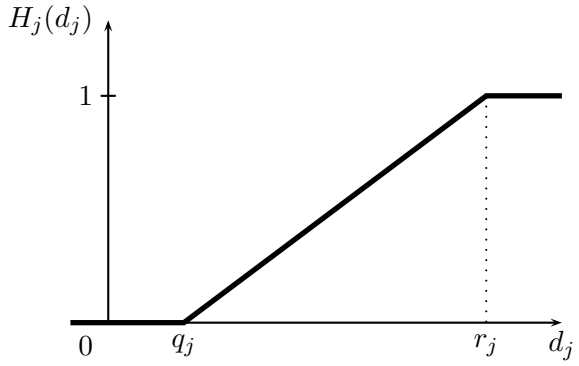
Using these pairwise differences, a *degree of preference* of one alternative over the other on a specific criterion is determined by means of so-called generalized preference functions: $H_j(d_j)$. These are monotonic nondecreasing or nonincreasing functions respectively for criteria that need to be maximized or minimized and may be different from one criterion to another. They are used to transform the pairwise differences, which may be expressed in any unit, into a normalized number between 0 and 1. These functions are used to add *intra-criterion preference information*, as they model the way a decision maker might perceive differences in scores of different alternatives with respect to a single criterion.

In the experiments performed in this paper we use a *linear generalized preference function with indifference region* (i.e., type 5 [3], see appendix B for details). This function has two parameters: q_j being the minimal pairwise difference in scores on a criterion for which the decision maker perceives a difference and r_j the so-called strict preference threshold. If the difference in scores of two alternatives on a criterion is less than q_j, the decision maker is indifferent and the degree of preference is 0. In case this difference is larger than r_j, the degree of preference is 1. The linear generalized preference function with indifference region, including its mathematical formulation, is depicted in Figure 2.

The degrees of preference are aggregated into so-called aggregated preference indices, by adding some *inter-criterion preference information*, in the form of a vector of *criterion weights*, $\boldsymbol{w}$. This weight vector is used to determine the relative importance of each criterion, and is assumed to be normalized, i.e. $\sum_{j=1}^{m} w_j = 1$.

Using the criterion weights, the so-called *aggregated preference index* is calculated. This index is a measure of the decision maker's preference of one alternative over all the others.

$$\Pi(a_k, a_l) = \sum_{j=1}^{m} w_j H_j(d_j(a_k, a_l)) = \sum_{j=1}^{m} w_j H_j(f_j(a_k) - f_j(a_l)) \tag{5}$$



$$H_j(d_j) = \begin{cases} 0 & d_j \leqslant q_j \\ \dfrac{d_j - q_j}{r_j - q_j} & q_j < d_j < r_j \\ 1 & d_j \geqslant r_j \end{cases}$$

Figure 2: Linear generalized preference function with indifference region

The most commonly used PROMETHEE methods are the PROMETHEE I and II methods [4], the former being based on the so-called positive and negative flows: ϕ^+ and ϕ^-, while the latter combines these two flows into a single *net flow*: $\phi = \phi^+ - \phi^-$. The positive flow of an alternative is a measure of how much this alternative is preferred over the others, while the negative flow is a measure of how much other alternatives are preferred over this one. They are defined in the following manner:

$$\phi^+(a_k) = \frac{1}{K-1} \sum_{l=1}^{K} \Pi(a_k, a_l) = \frac{1}{K-1} \sum_{j=1}^{m} \sum_{l=1}^{K} w_j H_j(f_j(a_k) - f_j(a_l)) \tag{6}$$

$$\phi^-(a_k) = \frac{1}{K-1} \sum_{l=1}^{K} \Pi(a_l, a_k) = \frac{1}{K-1} \sum_{j=1}^{m} \sum_{l=1}^{K} w_j H_j(f_j(a_l) - f_j(a_k)) \tag{7}$$

and consequently

$$\phi(a_k) = \frac{1}{K-1} \sum_{j=1}^{m} \sum_{l=1}^{K} w_j \left[H_j(f_j(a_k) - f_j(a_l)) - H_j(f_j(a_l) - f_j(a_k)) \right] \tag{8}$$

In this paper, we will only use the PROMETHEE II method, mainly because the PROMETHEE I method may result in incomparability between a pair of alternatives. In case of the PROMETHEE II method only the net flow ϕ is used. The outranking relation constructed in this method only involves preference and indifference relations, which are defined as:

- Alternative a_k is said to be preferred above a_l (denoted as $a_k \; P^{II} \; a_l$) if

$$\phi(a_k) > \phi(a_l) \tag{9}$$

- Alternatives a_k and a_l are considered to be indifferent (denoted as $a_k \; I^{II} \; a_l$) if

$$\phi(a_k) = \phi(a_l) \tag{10}$$

Unlike the PROMETHEE I method, the PROMETHEE II method ensures a complete ranking of all alternatives. Hence, in the remainder of this paper, when referring to the term PROMETHEE we restrict it to signify PROMETHEE II.

5. Integration of PROMETHEE and tabu search for the multi-objective knapsack problem

In this section, we develop three methods to combine the MOMH and the MCDM phase. These methods, of which the details are explained, are:

1. The *sequential* method follows the traditional MOMH paradigm and first generates an approximation of the Pareto frontier and then applies the PROMETHEE method to this set of solutions. The solution reported is the best one reported by the PROMETHEE method.
2. The MAUT method does not really integrate the PROMETHEE method, but essentially transforms the problem into a single-objective one, by aggregating the objectives in a weighted sum using the criterion weights of the PROMETHEE method. The solution reported is the one with the highest weighted objective function value.
3. The *simultaneous* method integrates the PROMETHEE method into the tabu search method by maintaining and updating a (small) archive of solutions throughout the search and determining the quality of a solution by comparing it to the solutions in the archive. The solution reported is the best one in the final (small) archive according to PROMETHEE.

We implicitly make the assumption that the preferences of the decision maker can be measured before the optimization starts. We further assume that only one solution is needed and that the solution reported should be as good as possible compared to other solutions. The method for comparison we use is the PROMETHEE method.

Combining the tabu search and the PROMETHEE method can be done in several ways, many of which are far from trivial. This is largely due to the fact that while the MCDM method is used to compare several solutions, the tabu search approach keeps only a single solution in memory at any given moment. In this respect, it is important to mention that the choice of a MCDM method does not render the problem a single-objective one, even not when all inter-criterion weights and intra-criterion information have been determined. This stands in contrast to the so-called MAUT methods that do transform the problem into a single-objective one by determining a new objective function as a weighted average of the other objective functions. However, an MCDM method, can only be used to *compare* solutions. It does not give any insight into the quality of a single solution, and requires at least two solutions to compare.

It is important to mention that our choice to use a local search metaheuristic — as opposed to one of the much more popular evolutionary MOMH— is not incidental. The main reason for this choice is that the archive of our simultaneous method has a very different purpose than the population of solutions in an evolutionary algorithm. Note that the purpose of

our exercise is to find one solution that compares as well as possible to other solutions to the same multi-objective problem, not to actually locate and report these other solutions. The archive of the simultaneous method is there to be able to compare solutions using the PROMETHEE method, whereas the population of an evolutionary MOMH is used to store solutions on the Pareto frontier. We felt that having two similar but different archiving mechanisms might obfuscate their use. Of course, none of this is to say that an archiving mechanism like the one presented here cannot be used with an evolutionary algorithm. On the contrary: the method presented here can be used just as well within an evolutionary framework.

5.1. Sequential method

The sequential method first finds an approximation of the Pareto frontier (called the *Pareto archive*) and then applies the PROMETHEE method to find the best solution. Searching an approximation of the Pareto frontier is done by iteratively applying the tabu search method, searching in a different search direction (using a different $\boldsymbol{\alpha}$-vector) each time. α_j is allowed to vary between 0 and 1 in s steps, i.e. α_j can take on values of 0, $1/(s-1)$, $2/(s-1)$, ..., 1. Each possible $\boldsymbol{\alpha}$-vector, for which $\sum_{j=1}^{m} \alpha_j = 1$ is generated. The $\boldsymbol{\alpha}$ vector is used in the tabu search method in the way described in the previous paragraph. In this way, the search is directed in every possible direction in the objective function space.

At the end of a single tabu search run, the best solution encountered is added to the Pareto archive, if it is not dominated by any solution in this archive. Also, all solutions in the archive that are dominated by the current solution, are removed.

After a tabu search, the search continues with a different $\boldsymbol{\alpha}$ -vector. In an iterated local search fashion, the search continuous from the solution produced by the previous tabu search, but this solution is first perturbed by the perturbation move described earlier. The new $\boldsymbol{\alpha}$ -vector implies that the value of each item changes and that the item value list needs to be updated. To save time, our procedure always keeps a sorted item value list by applying the quicksort algorithm to the previous list, with updated values. Usually, the number of items that are out of order between two runs of the tabu search procedure is small, which allows the quicksort algorithm to update this list quickly.

At the end of the multi-objective tabu search procedure, the Pareto archive is subjected to the PROMETHEE method as described above and the solution with the largest net flow is reported.

5.2. MAUT method

The MAUT method combines multi-objective optimization and MCDM in the simplest possible way. It should be stressed that this method does not integrate the PROMETHEE method into the tabu search. By using the same weights at each iteration of the tabu search, the method effectively transforms the problem into a single-objective one. The MAUT method always searches in a given direction in the objective function space by always using the same $\boldsymbol{\alpha}$-vector. This vector consists of the objective weights of the PROMETHEE method, i.e. for

this method $\boldsymbol{\alpha} \equiv \boldsymbol{w}$. After a tabu search run, the current solution is perturbed by the perturbation move and a new tabu search is started with the same $\boldsymbol{\alpha}$-vector. The method ends when a given number of restarts has been performed.

The solution reported at the end of the procedure is the one with the best weighted objective function.

5.3. Simultaneous method

The simultaneous method actually integrates the PROMETHEE method into the tabu search method. At each point in the optimization process, this method maintains a small archive of solutions. The size of the archive A remains fixed throughout the search and its primary purpose is to allow for a multicriteria evaluation of the current solution.

The initial archive is built as follows. First a set of A random $\boldsymbol{\alpha}$-vectors is generated. Then the initial archive is filled with A solutions by performing a tabu search (starting from a random initial solution) using one of the $\boldsymbol{\alpha}$-vectors as direction. The $\boldsymbol{\alpha}$-vector which was used to find the i-th solution in the archive is stored as $\boldsymbol{\alpha}_i$.

The simultaneous method then iterates the following steps:

1. The PROMETHEE method is run on the archive to determine the net flows (ϕ_i) of all solutions in the archive.
2. The solution with the lowest net flow (the "worst" solution according to the PROMETHEE method, i.e., solution $s = \arg\min_i \phi_i$) is removed from the archive.
3. A new solution is found by tabu searching using the vector $\boldsymbol{\alpha}_{\text{new}}$. This vector is calculated as a weighted combination of the other $\boldsymbol{\alpha}$-vectors, using the net flow of each solution they correspond to as weight, i.e., $\boldsymbol{\alpha}_{\text{new}} = \sum_{i=1}^{A-1} \phi_i \boldsymbol{\alpha}_i$.
4. The new solution is added to the archive if it is not a copy of a solution in the archive, otherwise it is discarded. This simple archive management system avoids premature convergence. All net flows of the solutions in the archive are updated efficiently, by the shortcut calculations outlined in appendix A.

The algorithm stops after a fixed number of restarts r. The solution returned is the one with the highest net flow in the final archive, according to the PROMETHEE method.

6. Experiments and results

To ensure a fair comparison of the three methods, they are constructed using the same building blocks as much as possible. In this section, we describe the experimental test sets, how the parameters of the methods have been set and how the methods have been compared.

Experiment data sets are randomly generated for knapsack problems with 10, 100 and 1000 items and 5, 10 and 20 objectives. Both the objectives (profits) and the costs of the items

are random integers between 0 and 50. Knapsack capacity is equal to 200 (10 items), 1000 (100 items) and 5000 (1000 items).

The parameters of the PROMETHEE method are set as follows. The weights of the objectives are all equal, i.e. $w_j = 1/m$. The parameters of the generalized linear preference function with indifference region are determined by setting q_j equal to zero, and p_j equal to the largest difference in the score on this objective between any two alternatives.

Table 2: Parameters of the different methods

Sequential	**MAUT**	**Simultaneous**
Tabu tenure t		
Perturbation probability P_{perturb}		
Maximum non-improving iterations $N_{\max}$		
Step size s	Restarts r	Restarts r
		Archive size A

For each of the methods, we first determine which of the parameters (see Table 2) have an important effect on the performance, considering both run time and solution quality. In general, the exact size of the tabu tenure and perturbation size have very little effect on the outcome of the method. The maximum number of non-improving tabu search iterations generally shows the largest positive effect, followed by the step size (sequential method) and the number of repeats (MAUT and simultaneous methods). It should be noted that the step size determines the number of times that the tabu search method is restarted in the sequential method, just like the number of repeats in the other two methods. For given values of the parameters of the tabu search method, the step size and the number of repeats therefore determine the length of the optimization run. Based on this study, we determine a reasonable default for each of the parameters of the tabu search method in such a way that they all would deliver their best performance for a given computing time.

Keeping the parameters of the tabu search method constant over the three methods, the step size or the number of repeats is set in such a way that each method is allocated the same computing time. This CPU time is determined beforehand as a time that we consider to be "reasonable".

To compare the performance of the three methods, we use a *repeated competition* strategy. In each competition, each of the three methods generates a solution. These solutions are then compared using the PROMETHEE method, using the same parameters as those that are used in the three methods. The method that finds the best solution (the one with the largest net flow) "wins" the competition. We repeat the competition 1000 times and record the number of wins for each method.

Results are reported in Tables 3, 4, and 5 for respectively 5, 10 and 20 objectives. Note that the total number of wins does not need to be 1000, as in a single competition more than one method may find the same solution or a solution with the same net flow.

Table 3: Results for 5 objectives instances

10 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.01	99	502	**766**	1367
0.1	71	480	**798**	1349
1	63	396	**788**	1247
100 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.01	121	621	613	1355
0.1	98	455	**640**	1193
1	44	377	**701**	1122
1000 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.01	52	512	558	1122
0.1	73	513	553	1139
1	31	283	**706**	1020

Table 4: Results for 10 objectives instances

10 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.02	40	538	534	1112
0.2	25	506	**659**	1190
2	6	343	**704**	1053
100 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.02	39	**602**	518	1159
0.2	45	415	**668**	1128
2	5	210	**832**	1047
1000 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.02	38	432	**751**	1221
0.2	32	481	**548**	1061
2	13	134	**854**	1001

Table 5: Results for 20 objectives instances

10 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.05	52	488	489	1029
0.5	69	413	**524**	1006
5	14	208	**789**	1011
100 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.05	6	402	**601**	1009
0.5	12	158	**848**	1018
5	0	45	**959**	1004
1000 items				
CPU (s)	**Sequential**	**MAUT**	**Simultaneous**	**Total**
0.05	1	**568**	437	1006
0.5	3	423	**574**	1000
5	1	211	**790**	1002

The most notable conclusion that can be drawn from these tables is that the sequential method is strongly outperformed by the other two methods. Clearly, generating an approximation of the Pareto frontier and then choosing the best solution using the PROMETHEE method is — in this specific experimental setting — an inferior strategy. Although this observation should be further investigated, a runtime analysis of the algorithm shows that the sequential method requires a prohibitively large amount of time to make the final comparison between all non-dominated solutions generated. This leaves little time to actually search for solutions in the allocated time. This conclusion is valid across all run time levels and across all problem sizes, although the situation does get worse for short running times and large problem sizes. In this case the number of solutions examined by the sequential method is far lower than those of the two simultaneous methods, because the sequential method needs far more time to do the final comparison of the solutions and may spend up to 80% of its time in the comparison phase, whereas the other methods are able to spend most of their time in the optimization phase. This also means that it is not very likely that the performance of the method can be improved by using a more elaborate MOMH than the simple tabu search heuristic employed here. On the contrary, a better MOMH will probably generate an even larger number of non-dominated solutions, forcing the PROMETHEE method to take up more of the alloted time to determine the best solution.

The difference in performance between the MAUT method and the sequential method is tested statistically using a binomial test, using a null hypothesis that both methods have a 50% probability to win the competition and an alternative hypothesis that the method with the highest number of wins performs better than the other one. Numbers in bold in Tables 3 to 5 indicate that the null hypothesis can be rejected at a 95% confidence level using a one-sided binomial test. All statistical tests are performed in R [22].

The numbers in bold show that the simultaneous method outperforms the MAUT method in most cases, indicating that there is merit to combining MCDM and MOMH methods in a more advanced way than just aggregating all objectives into one. Further analysis shows that the MAUT method performs approximately equally well for very short computing times. This may indicate that the simultaneous method, which is more sophisticated than the MAUT method, needs some time to fully unfold its potential. Given slightly larger computing times however, the simultaneous method clearly outperforms the MAUT method.

We could not find a statistically meaningful relationship between the performance of the different methods and the size of the problem, both in terms of number of objectives and in terms of number of items, other than the fact that the performance of the sequential method decreases as the problem size increases. This is probably due to the fact that the Pareto set approximation contains more and more solutions as the problem size increases, requiring more time to find them and more time to allow PROMETHEE to make a choice between them.

7. Conclusions and future research

The multi-objective optimization paradigm suggests that multi-objective optimization problems should be solved in two sequential steps. In a first step, an approximation of the Pareto frontier should be found, which is the main focus of most multi-objective algorithms. In a second step, the best solution on the Pareto frontier should be selected, taking into account the preferences of the decision maker, using a MCDM method. The fact that multi-objective algorithms may generate a very large number of non-dominated solutions and that MCDM methods are generally not conceived with this fact in mind raises some doubts about the soundness of this approach.

The aim of this paper was to experimentally test whether there is evidence to suggest that integrating the MCDM method into the multi-objective optimization method (and thus using both methods *simultaneously*) may in fact be preferable to the sequential option. For the multi-objective knapsack problem, we therefore developed a simple multi-objective tabu search method and combined this with the PROMETHEE MCDM method in three ways, one sequential and two simultaneous. The two simultaneous methods tested were a simple integration in which only the weights of the multicriteria method are used to guide the search (the MAUT method) and a more elaborate integration, in which the PROMETHEE method was truly integrated into the multi-objective tabu search. To ensure an honest comparison, we made sure to reuse the same components as much as possible in all methods.

To determine the best way to combine our multi-objective optimization method with the PROMETHEE method, all three combinations were run on a series of artificially generated instances of varying sizes of the multi-objective knapsack problem. Results were statistically compared to determine the best method as a function of the size of the problem instance.

Our results clearly show that — for this specific experimental setting — integrating the MCDM method into the MOMH strongly outperforms applying both methods sequentially. Both the simple and the elaborate simultaneous methods produce far better results on

instances of all sizes. A strong conclusion of this research is therefore that — if there is any way to measure the decision maker's preferences and to decide upon the method that will be used to select the best solution *prior* to the optimization process — it could be beneficial to integrate the multicriteria decision phase into the multi-objective optimization method. Further, we found that for larger computing times the more advanced integration method outperformed the simple MAUT method.

An alternative approach to the integration of MCDM and MOMH is to limit the number of solutions that the MOMH produces. This should drastically reduce the computing time necessary to do the final comparison between solutions by the MCDM method. In the literature, several methods to restrict the number of solutions reported by an MOMH have been reported [e.g., 18, 20]. Comparing a sequential approach that limits the number of solutions produced during the search with our simultaneous MOMH and MCDM approach is left for future research.

A. Impact of the replacing an alternative on the PROMETHEE methods

Suppose we replace alternative a_l by a new alternative $\tilde{a}_l$. This substitution will have an impact on the final ranking. If we denote the new situation by a ˜ one may conclude from expression (4) that

$$\tilde{d}_j(a_k, a_{k'}) = d_j(a_k, a_{k'}),\ \forall k, k' = 1, \ldots, K \text{ with } k \neq l \neq k', \forall j = 1, \ldots, m \tag{11}$$

and that only the values $\tilde{d}_j(a_k, \tilde{a}_l)$ and $\tilde{d}_j(\tilde{a}_l, a_k)$ might change $\forall k = 1, \ldots, K$ and $\forall j = 1, \ldots, m$.

Hence, only the corresponding preference degrees $\tilde{H}_j(a_k, \tilde{a}_l)$ and $\tilde{H}_k(\tilde{a}_l, a_k)$ $\forall k = 1, \ldots, K$ and $\forall j = 1, \ldots, m$ could have changed. Consequently, only the aggregated preference indices $\tilde{\Pi}(a_k, \tilde{a}_l)$ and $\tilde{\Pi}(\tilde{a}_l, a_k)$, $\forall k = 1, \ldots, K$ will have to be replaced.

To conclude the impact at the level of the flows can easily be expressed as follows in case $k \neq l$:

$$\tilde{\phi}^+(a_k) = \phi^+(a_k) - \frac{1}{K-1}\left[\Pi(a_k, a_l) - \tilde{\Pi}(a_k, \tilde{a}_l)\right] \tag{12}$$

$$\tilde{\phi}^-(a_k) = \phi^-(a_k) - \frac{1}{K-1}\left[\Pi(a_l, a_k) - \tilde{\Pi}(\tilde{a}_l, a_k)\right] \tag{13}$$

$$\tilde{\phi}(a_k) = \phi(a_k) - \frac{1}{K-1}\left[\Pi(a_k, a_l) - \Pi(a_l, a_k) - \tilde{\Pi}(a_k, \tilde{a}_l) + \tilde{\Pi}(\tilde{a}_l, a_k)\right] \tag{14}$$

while in case $k = l$ one uses definitions (6) and (7) to determine respectively $\tilde{\phi}^+(\tilde{a}_l)$ and $\tilde{\phi}^-(\tilde{a}_l)$. An alternative way to calculate $\tilde{\phi}(\tilde{a}_l)$ uses the property:

$$\sum_{k=1}^{K} \phi(a_k) = 0 \tag{15}$$

hence

$$\tilde{\phi}(\tilde{a}_l) = -\sum_{\substack{k=1 \\ k \neq l}}^{m} \tilde{\phi}(a_k) \tag{16}$$

B. Some specificities on the generalized preference functions

In our approach we will only use, for sake of simplicity, generalized preference functions of the so-called type 5, i.e. the V-shaped criterion with indifference threshold [3]:

$$H_j(d_j(a_k, a_l)) = \begin{cases} 0 & \text{if } d_j(a_k, a_l) < q_j \\ \dfrac{d_j(a_k, a_l) - q_j}{r_j - q_j} & \text{if } d_j(a_k, a_l) \in [q_j, r_j] \\ 1 & \text{if } d_j(a_k, a_l) > r_j \end{cases} \tag{17}$$

In the PROMETHEE II method the basic entity is the difference:

$$H_j(d_j(a_k, a_l)) - H_j(d_j(a_l, a_k)) = H_j(d_j(a_k, a_l)) - H_j(-d_j(a_k, a_l)) = \Delta_{kl}^{(j)} \tag{18}$$

Taking into account relation (17) one may rewrite the function $\Delta_{kl}^{(j)}$ as:

$$\Delta_{kl}^{(j)} = \begin{cases} -1 & \text{if } d_j(a_k, a_l) < -r_j \\ \dfrac{-d_j(a_k, a_l) - q_j}{r_j - q_j} & \text{if } d_j(a_k, a_l) \in [-r_j, -q_j] \\ 0 & \text{if } d_j(a_k, a_l) \in]-q_j, q_j[\\ \dfrac{d_j(a_k, a_l) - q_j}{r_j - q_j} & \text{if } d_j(a_k, a_l) \in [q_j, r_j] \\ 1 & \text{if } d_j(a_k, a_l) > r_j \end{cases} \tag{19}$$

or equivalently

$$\Delta_{kl}^{(j)} = \begin{cases} 0 & \text{if } |d_j(a_k, a_l)| < q_j \\ \dfrac{\text{sgn}(d_j(a_k, a_l)) d_j(a_k, a_l) - q_j}{r_j - q_j} & \text{if } |d_j(a_k, a_l)| \in [q_j, r_j] \\ \text{sgn}(d_j(a_k, a_l)) & \text{if } |d_j(a_k, a_l)| > r_j \end{cases} \tag{20}$$

where $\mathrm{sgn}(x)$ stands for the sign of x.

Hence, when using the PROMETHEE II method it suffices to determine only the triangular part of the matrices $(d_j(a_k, a_l))$, $\forall j = 1, \ldots, m$: the elements with indices $k = 1, \ldots, K$ and $l = k, \ldots, K$ in order to be able to determine the matrices $(\Delta_{kl}^{(j)})$, $\forall j = 1, \ldots, m$. The net flow is determined by

$$\phi(a_k) = \frac{1}{K-1} \sum_{j=1}^{m} \sum_{l=1}^{K} w_j \Delta_{kl}^{(j)} \tag{21}$$

References

[1] J. Branke, T. Kauler, and H. Schmeck. Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32:499–507, 2001.

[2] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. In *The Eighth Conference on Parallel Problem Solving from Nature*, pages 722–731, 2004.

[3] J.-P. Brans and B. Mareschal. PROMETHEE methods. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple criteria decion analysis – State of the art surveys*, pages 163–195, 2005.

[4] J.-P. Brans and P. Vincke. A preference ranking organisation method : The PROMETHEE method for MCDM. *Management Science*, 31:647–656, 1985.

[5] R. F. Coelho, H. Bersini, and Ph. Bouillard. Parametrical mechanical design with constraints and preferences: application to a purge valve. *Computational Methods Appl. Mechanical Engineering*, 192:4355–4378, 2003.

[6] C.A.C. Coello. Handling preferences in evolutionary multiobjective optimization: A survey. In *Proceedings of the 2000 Congress on Evolutionary Computation*. IEEE Computer Society, 2000.

[7] D. Cvetkovic and I.C. Parmee. Genetic algorithm based multi-objective optimization and conceptual engineering design. In *Congress on Evolutionary Computation CEC99*, pages 29–36. IEEE, 1999.

[8] D. Cvetkovic and I.C. Parmee. Designers preferences and multi-objective preliminary design processes. In I.C. Parmee, editor, *Proceedings of Fourth International Conference on Adaptive Computing in Desingn and Manufacture (ACDM 2000)*, pages 249–260. Springer London, 2000.

[9] D. Cvetkovic and I.C. Parmee. Use of preferences for gabased multiobjective optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999.

[10] K. Deb. Solving goal programming problems using multi-objective genetic algorithms. In *Congress on Evolutionary Computation*, pages 77–84. IEEE, 1999.

[11] K. Deb and H. Gupta. Searching for robust pareto-optimal solutions in multi-objective optimization. In *Evolutionary Multi-Criterion Optimization*, volume 3410, pages 150–164. Springer Berlin / Heidelberg, 2005.

[12] C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S.Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423. Morgan Kauffman Publishers, 1993.

[13] C.M. Fonseca and P.J. Fleming. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28:26–37, 1998.

[14] G.W. Greenwood, X.S. Hu, and J.G. DAmbrosio. Fitness functions for multiple objective optimization problems: Combining preferences with pareto rankings. In R.K. Belew and M.D. Vose, editors, *Foundations of Genetic Algorithms 4*, pages 437–455. Morgan Kaufmann, 1997.

[15] A. Jaszkiewicz. Evaluation of multiobjective metaheuristics. In X. Gandibleux, M. Sevaux, K. Srensen, and V. T'kindt, editors, *Metaheuristics for multiobjective optimization*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 65–90, Berlin, 2004. Springer.

[16] Y. Jin and B. Sendhoff. Incorporation of fuzzy preferences into evolutionary multiobjective optimization. In *Proceedings of the 4th Asia Pacific Conference on Simulated Evolution and Learning*, pages 26–30, 2002.

[17] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105. IEEE Press, 1999.

[18] J. Knowles and D. Corne. Bounded Pareto archiving: Theory and practice. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'Kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 39–64. Springer, 2004.

[19] S. Massebeuf, C. Fonteix, L.N. Kiss, I. Marc, F. Pla, and K. Zaras. Multicriteria optimization and decision engineering of an extrusion process aided by a diploid genetic algorithm. In *Congress on Evolutionary Computation*, pages 14–21. IEEE Service Center, 1999.

[20] C.A. Mattson, A.A. Mullur, and A. Messac. Smart Pareto filter: Obtaining a minimal representation of multiobjective design space. *Engineering Optimization*, 36(6):721–740, 2004.

[21] T. Meyarivan, K. Deb, A. Pratap, and S. Agarwal. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[22] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

[23] L. Rachmawati and D. Srinivasan. Preference incorporation in multi-objective evolutionary algorithms: A survey. In *IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 3385–3391. IEEE Press, 2006.

[24] B. Rekiek, P. De Lit, F. Pellichero, T. LEglise, E. Falkenauer, and A. Delchamber. Dealing with userss preferences in hybrid assembly lines design. In *Proceedings of the MCPL2000 Conference*, 2000.

[25] B. Roy. The outranking approach and the foundation of ELECTRE methods. *Theory and Decision*, 31:49–73, 1991.

[26] T.L. Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1(1):83–98, 2008.

[27] D.A. Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[28] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In X. Gandibleux, M. Sevaux, K. Srensen, and V. T'kindt, editors, *Metaheuristics for multiobjective optimization*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–38, Berlin, 2004. Springer.