International Journal of Information Technology & Decision Making
Vol. 13, No. 5 (2014) 917–936
© World Scientific Publishing Company
DOI: 10.1142/S0219622014500308

Progressive Multi-Objective Optimization

Kenneth Sörensen* and Johan Springael
University of Antwerp, Faculty of Applied Economics
ANT/OR – University of Antwerp Operations Research Group
Prinsstraat 13, B–2000 Antwerp, Belgium
**kenneth.sorensen@ua.ac.be*

Published 23 December 2013

This paper introduces *progressive multi-objective optimization* (PMOO), a novel technique to include the decision maker's preferences into the multi-objective optimization process. PMOO integrates a well-known method for multi-criteria decision making (PROMETHEE) into a simple multi-objective metaheuristic by maintaining and updating a small reference archive of nondominated solutions throughout the search. By applying this novel technique to a set of instances of the multi-objective knapsack problem, the superiority of PMOO over the commonly accepted sequential approach of generating a Pareto set approximation first and selecting a single solution afterwards is demonstrated.

Keywords: Multi-objective optimization; multi-criteria decision-making; user preferences.

1. Multi-Objective Optimization and Multi-Criteria 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, the number of vehicles used, and (to make sure that all routes are approximately of equal length) the difference between the duration of the longest and the shortest trip. In such a *multi-objective* optimization problem, the notion of optimality needs to be abandoned in favor of the concept of *domination*. A solution x is said to dominate a solution y if x is at least as good as y with respect to all objectives and better with respect to at least one. The aim of multi-objective optimization is therefore not to find an optimal solution, but rather to find the set of nondominated solutions, called the *Pareto frontier*.

Given the fact that most multi-objective optimization problems are NP-hard, (meta)heuristics often present the only viable techniques to solve them. Such *multi-objective metaheuristics* (MOMH) generally generate a (potentially very large) set of (mutually nondominated) solutions, from which the decision maker is left to choose one. This set is called the Pareto frontier approximation. Figure 1 illustrates these concepts.

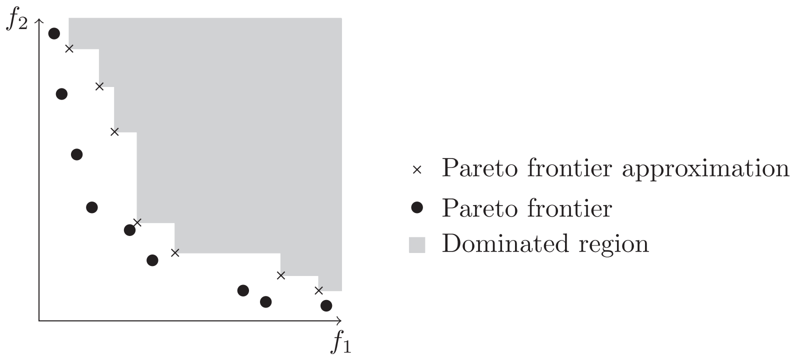

Fig. 1. Multi-objective optimization — the aim of the optimization procedure is to approximate the Pareto frontier.

Most papers on MOMH do not consider the problem of selecting the final solution from the Pareto frontier approximation, and (implictly) assume that the decision maker will use a *multi-criteria decision-making* (MCDM) method for this purpose. In order to do this, the MCDM method must model the decision maker's preferences and use them to determine the best solution. However, rather than using *a posteriori* method (that finds a Pareto frontier approximation first and only then selects a solution), it often makes sense to *incorporate the preferences of the user directly into the* MOMH. At the very least, using *a priori* approach solves the problem that most MCDM methods are not designed with the output of a typical MOMH in mind. MOMH may generate thousands of solutions, whereas MCDM methods were designed to select between a set of solutions that is far smaller than that. Notwithstanding the fact that some methods have been proposed to limit the number of solutions reported by the MOMH (e.g., see Refs. 1 and 2), an integration of preferences into a MOMH may prevent the MCDM method from having to choose between a very large number of solutions.

However, few papers, most of which have been surveyed in Refs. 3 and 4, describe methods in which the user's preferences are integrated into a MOMH. In the literature, examples can be found of preferences being represented as solution attributes,[5,6] trade-offs between objectives,[7] optimization goals,[8–10] outranking relations,[11–13] fuzzy preferences[14–17] or utility functions.[18] A recent approach is *g*-dominance,[19] that uses the idea of reference points[20] that are essentially projections of the user's preferences and searches for solutions that are close to these reference points.

Perhaps surprisingly, the literature seems devoid of any attempt to integrate an actual MCDM method into a MOMH. Although several methods have been developed that integrate the decision maker's preferences into the multi-objective optimization process, these methods generally do not use the theory, concepts and techniques developed in the field of MCDM, but define an ad hoc way to do this. In this paper, a new method for the integration of MOMH and MCDM is developed that does truly integrate a MCDM method into a MOMH. In this way, the advantages of using a widely accepted MCDM technique (e.g., respect for scales, no need to use a utility function) are taken along. This new method, called *progressive multi-objective optimization* (PMOO), maintains a (small) reference archive of nondominated solutions throughout the search. It uses a multi-criteria method (PROMETHEE, explained in Sec. 2) to determine whether incumbent solutions are allowed into the archive by comparing them to the solutions already in the archive. The use of PROMETHEE is not compulsory for the ideas presented in this paper and the use of a different MCDM should only result in a technical difference in the final implementation of PMOO. A local search metaheuristic is used (tabu search in this paper) to search for these incumbent solutions. The metaheuristic uses information from the multi-criteria evaluation of the solutions in the archive to guide the search towards promising regions of the search space.

Progressive multi-objective optimization is *a priori* in nature since the preference modeling takes place before the optimization process. This, of course, assumes that such modeling can be done and that the goal of the method is to produce a single solution. There are many situations imaginable in which the decision maker would actually prefer to obtain several structurally different solutions, that can be used to structure and can even change his/her preferences in the course of the decision-making process. We argue however, that if the decision maker's preferences are measurable and relatively stable, and it is desired of the decision-making method to produce a single solution rather than a Pareto set of solutions, the PMOO method offers a compelling alternative over other comparable methods.

PMOO is one of the first attempts to truly integrate an MCDM within a MOMH. This paper investigates whether this new approach outperforms the *a posteriori* approach implicitly assumed in the MOMH-literature. In a controlled lab experiment, the performance of both methods is compared by applying them to a variety of instances of the *multi-objective knapsack problem*, a generalization of the single-objective knapsack problem in which each item has several profit values. To this end, a simple *tabu search* MOMH is developed (see Sec. 3) and combined with the PROMETHEE MCDM method (see Sec. 4).

The progressive method is compared to a traditional "MCDM-follows-MOMH" method, 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 in Sec. 5. In Sec. 6, some conclusions are drawn and pointers for future research are given.

2. The PROMETHEE Multicriteria Method

The PROMETHEE multicriteria method, first introduced by Brans,[21] is used to compare a set of alternatives (solutions) that have all been ranked on a number of criteria (objectives). For a complete overview of the PROMETHEE method and its variants, see Ref. 22. The starting point for this method is an evaluation table

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	$f_1(a_K)$	$\dots$	$f_m(a_K)$

(see Table 1), that contains for each alternative a_k its score on each criterion f_j. In the language of multi-objective optimization, alternatives correspond to solutions that need to be compared, and the criteria are the objective function values on the basis of which this comparison is to be made.

From the evaluation table, the PROMETHEE method calculates all pairwise differences between any two alternatives k and l on each criterion:

$$d_j(a_k, a_l) = f_j(a_k) - f_j(a_l), \quad \forall j = 1, \dots, m, \ \forall k, l = 1, \dots, K. \tag{2.1}$$

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.

The experiments performed in this paper use a *linear generalized preference function with indifference region* (i.e., type 5, Ref. 22). 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 p_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 p_j, the degree of preference is 1. The linear generalized preference function with indifference region, including its mathematical formulation, is depicted in Fig. 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*, $\mathbf{w}$. This weight vector is assumed to be normalized, i.e., $\sum_{j=1}^{m} w_j = 1$. The aggregated preference index is a measure of the decision maker's preference of an alternative a_k over an alternative a_l.

$$\Pi(a_k, a_l) = \sum_{j=1}^{m} w_j H_j(d_j(a_k, a_l)) = \sum_{j=1}^{m} w_j H_j(f_j(a_k) - f_j(a_l)). \tag{2.2}$$

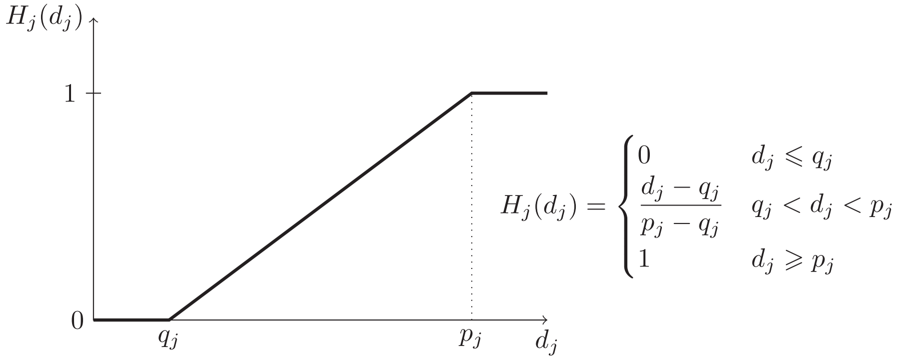

Fig. 2. Linear generalized preference function with indifference region (for maximization of f_j).

The most commonly used PROMETHEE methods are the PROMETHEE I and II methods,[23] 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:

$$\phi^+(a_k) = \frac{1}{K-1}\sum_{l=1}^{K}\Pi(a_k, a_l) = \frac{1}{K-1}\sum_{j=1}^{m}\sum_{l=1}^{K} w_j H_j(f_j(a_k) - f_j(a_l)), \quad (2.3)$$

$$\phi^-(a_k) = \frac{1}{K-1}\sum_{l=1}^{K}\Pi(a_l, a_k) = \frac{1}{K-1}\sum_{j=1}^{m}\sum_{l=1}^{K} w_j H_j(f_j(a_l) - f_j(a_k)) \quad (2.4)$$

and consequently

$$\phi(a_k) = \frac{1}{K-1}\sum_{j=1}^{m}\sum_{l=1}^{K} w_j[H_j(f_j(a_k) - f_j(a_l)) - H_j(f_j(a_l) - f_j(a_k))]. \quad (2.5)$$

The PMOO method developed in this paper uses the PROMETHEE II method, mainly because the PROMETHEE I method may result in incomparability between a pair of alternatives, whereas the PROMETHEE II method is always able to rank any pair of solutions. 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

$$\phi(a_k) > \phi(a_l). \quad (2.6)$$

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

$$\phi(a_k) = \phi(a_l). \quad (2.7)$$

Unlike the PROMETHEE I method, the PROMETHEE II method ensures a complete ranking of all alternatives. This is necessary because PMOO works with a small reference archive of nondominated solutions, and some steps require the identification of the "best" or the "worst" solution in this archive. If incomparability between solutions is allowed, the concept of "best" or "worst" solution becomes problematic. A more involved version of PMOO could allow incomparability, using modified definitions of the "best" and the "worst" solution. However, the PMOO algorithm developed in this paper only uses PROMETHEE II. In the remainder of this paper, the term PROMETHEE is therefore used to signify only PROMETHEE II.

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, and has many variants.[24] In its simplest form, this problem can be described as follows. Given n items, each with a certain cost c_i, $i \in [1, n]$ and a certain value or profit g_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[25] is a simple extension of the single-objective knapsack problem, in which each item has m profits, and the objective is to maximize all m objectives simultaneously. In this problem, g_{ij} represents the profit (gain) of item i in objective function j ($j = \{1, \dots, m\}$), and $f_j(\mathbf{x})$ represents the score of solution $\mathbf{x}$ on the jth objective function value. Note that the term "maximize" is put between quotes in this formulation, because of the multi-objective nature of the problem, in which a true maximum does not exist.

$$\underset{j=\{1,\dots,m\}}{\text{``maximize''}} f_j(\mathbf{x}) = \sum_{i=1}^{n} g_{ij} x_i \tag{3.1}$$

$$\text{s.t.} \sum_{i=1}^{n} c_i x_i \leq C \tag{3.2}$$

$$x_i \in \{0, 1\}. \tag{3.3}$$

The PMOO method described in Sec. 4 embeds the PROMETHEE method into a simple tabu search metaheuristic. Tabu search is a metaheuristic framework introduced by Glover[26] that uses memory structures to guide the search through the solution space. The tabu search procedure iteratively improves the quality of a solution by adding or removing one item at a time. It should be noted that the aim of this tabu search procedure is to provide a simple foundation upon which the principles of the progressive multi-objective optimization framework can be built. It is therefore not expected to be competitive with state-of-the-art approaches for the multi-objective knapsack problem, such as the one by Gandibleux and Freville.[27]

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 $\alpha = \{\alpha_1, \ldots, \alpha_m\}$ determines the weight of each profit function and determines the search direction in the objective function space. In the PMOO method, the α-vector is determined by examining the solutions in its archive, and is the main vehicle through which the PMOO method steers the search in a promising direction.

Using the current weight vector, the value of item i is calculated as $v_i = \sum_{j=1}^{m} g_{ij}\alpha_j / c_i$. 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}. The only restriction is that items that are absent in the solution are only added 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. Progressive MOMH: Integration of PROMETHEE and Tabu Search for the Multi-Objective Knapsack Problem

The progressive MOMH method integrates the PROMETHEE method into the tabu search method by maintaining and updating a (small) reference archive of (high-quality) solutions throughout the search. The PROMETHEE method treats these solutions as alternatives that need to be compared against each other. Similarly, the objective functions of the multi-objective knapsack problem are the criteria that the MCDM uses to base its comparison on. The corresponding terms in the PROMETHEE method and the tabu search method are explained in Table 2.

The main purpose of the PMOO archive is twofold. First, it is used to determine the quality of an incumbent solution by comparing it to the solutions in the archive using the PROMETHEE method. Second, the output of the PROMETHEE method comparing the solutions in the archive is used to determine the value of the current α-vector of the tabu search method and hence steer the search towards promising

Table 2. Corresponding terms in MCDM and MOMH.

MCDM	MOMH
Criterion (f_j)	Objective function (f_j)
Alternative (a_k)	Solution ($\mathbf{x}$)

solutions. The solution reported is the best one in the final archive according to PROMETHEE.

The size A of the archive remains fixed throughout the search. The initial archive is built as follows. First a set of A random α-vectors is generated. Then the initial archive is filled with A solutions by performing a tabu search (starting from a random initial solution) using each of the α-vectors. The α-vector which was used to find the ith solution in the archive is stored as α_i.

The progressive MOMH method then iterates the following steps (a worked-out example of these steps can be found in Appendix B):

(1) The PROMETHEE method is run on the archive to determine the net flows (ϕ_i) of all solutions in the archive. These net flows are calculated based on the scores of each solution in the archive on each of the different objectives.

(2) A new solution is found by tabu searching using the vector α_{new}. This vector is calculated as a weighted combination of the α-vectors of all solutions except the one with the lowest net flow. The net flow of each solution is used as a weight for the α_i-vectors, i.e.,

$$\alpha_{\text{new}} = \frac{\sum_{i=1}^{A-1} \phi_i \alpha_i}{\sum_{i=1}^{A-1} \phi_i}. \tag{4.1}$$

The tabu search starts from the previous solution (i.e., the solution that was returned by the latest run of the tabu search procedure), that is first perturbed by the perturbation move described earlier.

The new α-vector implies that the value of each item changes and that the item value list needs to be updated. To save time, the 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.

(3) 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.

(4) 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.

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.

Unlike other methods for multi-objective optimization the PMOO method never aggregates the objective function values of a solution in a methodologically incorrect way. Rather, the quality of a solution (i.e., its *net flow*) is always determined with respect to the other solutions in the archive. Like in most MCDM methods and unlike in aggregation methods, the comparison of two solutions does not only depend on the objective function values of those solutions, but also on the values of the solutions in the reference archive. Because of this, PMOO can be said to truly integrate an MCDM method in a MOMH.

5. Experiments and Results

We compare the method to two other methods. The first method is called the sequential method, and implements the common MOMH paradigm of first generating an approximation of the Pareto frontier and then selecting a solution from this approximation. As will become clear from the results in this section, the sequential method is strongly outperformed by the progressive method. To exclude the possibility that this is *only* due to the fact that the progressive method integrates the decision maker's preferences, and that it does not matter in which way these preferences are integrated, it is also compared to a method that integrates the preferences in the simplest possible way. The resulting method uses the preferences as weights in a scalar objective function.

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 α-vector) each time.

The list of α-vectors to use is generated as follows. First, a step size s is determined. The individual elements of each α-vector, $\alpha_j(j \in \{1, \dots, m\})$ are 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 α-vector, for which $\sum_{j=1}^{m} \alpha_j = 1$ is generated. The α-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, in an "evenly spaced" way.

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 already in this archive. Also, all solutions in the archive that are dominated by the new solution, are removed.

After a tabu search, the search continues with a different α-vector. In an iterated local search fashion, the search continues from the solution produced by the previous tabu search, but this solution is first perturbed by the perturbation move described earlier.

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. *Utility-based method*

The utility-based method combines multi-objective optimization and MCDM in the simplest possible way. It should be stressed that this method effectively changes the multi-objective problem into a single-objective one and can therefore not be considered a "true" multi-objective method, nor does it really integrate the PROMETHEE method into the tabu search. The utility-based method works by repeatedly searching in the same direction in the objective function space by always using the same α-vector. This vector consists of the objective weights of the PROMETHEE method, i.e., for this method $\alpha \equiv \mathbf{w}$. After a tabu search run, the current solution is perturbed by the perturbation move and a new tabu search is started with the same α-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. *General experiment setup*

To ensure a fair comparison of the three methods, they are constructed using the same building blocks, described in previous sections. This section describes 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. These values are used to set the PROMETHEE method in both the PMOO and the sequential method. For the utility-based method, the same weight values are used.

For each of the methods, it is first determined which of the parameters (see Table 3) 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

Table 3. Parameters of the different methods.

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

nonimproving tabu search iterations generally shows the largest positive effect, followed by the step size (sequential method) and the number of repeats (utility-based and PMOO 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, a reasonable default is determined 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 is considered "reasonable".

Comparing the performance of the three methods is done using a *repeated competition* strategy. In each competition, each of the three methods generates one solution. These three solutions are then compared using the PROMETHEE method, using the same parameters (w_j, q_j, and p_j) as those that are used in the tested methods. The method that finds the best solution (the one with the largest net flow) "wins" the competition. The competition is repeated 1000 times and record the number of wins for each method.

Results are reported in Tables 4–6 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.

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 nondominated 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

Table 4. Results for five objectives instances.

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

method is far lower than those of the two other 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 allocated time to determine the best solution.

Table 5. Results for 10 objectives instances.

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

Table 6. Results for 20 objectives instances.

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

The difference in performance between the utility-based method and the PMOO 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 4 to 6 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.[28]

The numbers in bold show that the PMOO method outperforms the utility-based 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 utility-based method performs approximately equally well for very short computing times. This may indicate that the PMOO method, which is more sophisticated than the utility-based method, needs some time to fully unfold its potential. Given slightly larger computing times however, the PMOO method clearly outperforms the utility-based method.

A statistically meaningful relationship between the performance of the different methods and the size of the problem could not be found, 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.

The experimental study in this paper, like every experimental study of a new method, has its limitations and further study is necessary to establish in which way the results can be transferred to other problems. The multi-objective knapsack problem is a very simple problem and testing the algorithm on more complex multi-objective optimization problems, consisting, e.g., of different types of objective functions, is necessary. Second, using a more powerful multi-objective optimization method instead of the simple tabu search method developed in this paper, would be interesting. Finally, it would be interesting to compare the method to other methods from the literature. This would require, however, that other methods for multi-objective optimization would exist that are able to determine a single solution as part of their functioning.

6. Conclusion 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 nondominated solutions and that MCDM methods are generally not conceived with this fact in mind raises some doubts about the soundness of this approach.

Several attempts can be found in the literature to integrate the decision maker's preferences into a MOMH. However, very few (if any) of these methods are based on an actual MCDM method, which is surprising given that comparing alternatives on different criteria is exactly the aim of MCDM methods. The aim of this paper was to experimentally test whether there is evidence to suggest that integrating an MCDM method into a multi-objective optimization method (and thus using both methods *simultaneously*) may in fact be preferable to the sequential option. To this end, a novel method for multi-objective optimization was developed, called *progressive* multi-objective optimization or PMOO. PMOO integrates the PROMETHEE method into a simple tabu search metaheuristic and maintains a small reference archive of nondominated solutions to perform the coordination between the MOMH and the MCDM method.

For the multi-objective knapsack problem, the PMOO method was compared to two other methods: the sequential and the utility-based method. The sequential method adopted the traditional optimize-first, select-second paradigm typical of the MOMH literature. The utility-based method implemented a simple weighted-objective strategy. To ensure an honest comparison, the same components were reused as much as possible in all methods. The methods 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 — the PMOO method strongly outperforms the sequential method. More surprisingly, the utility-based method does too. 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, it was established that for larger computing times the more advanced PMOO method outperformed the simple utility-based method.

A second result of this research was the fact that the PMOO method outperformed the utility-based method for larger computing times, indicating that a more elaborate integration of preferences into a MOMH is beneficial when more CPU time is available.

This paper has presented a first description of the PMOO method, in which a simple tabu search method was used as the MOMH. The choice for this simple method was made in order not to distract the attention away from the core of the PMOO method to the intricacies of the MOMH. However, it is clear that other MOMH can be used as the basis on which to build a PMOO method. Given that a large majority of MOMH are of the evolutionary type, research on how to integrate the PROMETHEE method (or another MCDM) in an evolutionary MOMH is warranted. Essentially, the PMOO method relies on the MCDM method to guide the search in the MOMH method. The MCDM method should therefore be integrated in that component of the MOMH that is responsible for leading the search towards promising regions of the search space. In an evolutionary algorithm, it is usually the selection component that performs this function. An evolutionary PMOO would therefore use the MCDM method to select solutions from the evolutionary archive for reproduction and possible also for population update (to select bad solutions from the population to replace with good solutions). How this should be done most efficiently is a subject for future research.

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., see Refs. 1 and 2). Comparing a sequential approach that limits the number of solutions produced during the search with the PMOO approach is left for future research.

Appendix A. Impact of Replacing an Alternative on the PROMETHEE Methods

Suppose alternative a_l is replaced by a new alternative $\tilde{a}_l$. This substitution will have an impact on the final ranking. If the new situation is denoted by $\tilde{a}$ one may conclude from expression (2.1) that

$$\tilde{d_j}(a_k, a_{k'}) = d_j(a_k, a_{k'}), \quad \forall\, k, k' = 1, \ldots, K \text{ with } k \neq l \neq k', \quad \forall\, j = 1, \ldots, m \quad \text{(A.1)}$$

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 $k \neq l$:

$$\tilde{\phi}^+(a_k) = \phi^+(a_k) - \frac{1}{K-1}\left[\Pi(a_k, a_l) - \tilde{\Pi}(a_k, \tilde{a}_l)\right], \tag{A.2}$$

$$\tilde{\phi}^-(a_k) = \phi^-(a_k) - \frac{1}{K-1}\left[\Pi(a_l, a_k) - \tilde{\Pi}(\tilde{a}_l, a_k)\right], \tag{A.3}$$

$$\tilde{\phi}(a_k) = \phi(a_k) - \frac{1}{K-1}\left[\Pi(a_k, a_l) - \Pi(a_l, a_k) - \tilde{\Pi}(a_k, \tilde{a}_l) + \tilde{\Pi}(\tilde{a}_l, a_k)\right] \tag{A.4}$$

while in case $k = l$ one uses definitions (2.3) and (2.4) 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:

$$\sum_{k=1}^{K} \phi(a_k) = 0 \tag{A.5}$$

hence

$$\tilde{\phi}(\tilde{a}_l) = -\sum_{\substack{k=1 \\ k \neq l}}^{m} \tilde{\phi}(a_k). \tag{A.6}$$

Appendix B. Example

Let us for sake of clarity consider the small example of a bi-objective knapsack problem involving three items and a reference archive consisting of four possible solutions. The objective functions, which also serve as criteria in the multicriteria method (i.e., PROMETHEE in this case) are given by

$$f_1(\mathbf{x}) = 4x_1 + 7x_2 + 5x_3 \quad \text{and} \quad f_2(\mathbf{x}) = 5x_1 + 5x_2 + 6x_3. \tag{B.1}$$

We do not consider any item weights in this example, since they are not relevant to show the working of the PMOO method. Additionally, due to the problem size of this instructive example, the archive contains some dominated solutions which would not occur in a realistic setting.

Suppose the starting reference archive consists of the following solutions a_i together with their scores on the criteria:

	x_1	x_2	x_3	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$
a_1	0	0	1	5	6
a_2	1	1	0	11	10
a_3	1	0	1	9	11
a_4	0	1	0	7	5

Implementing the PROMETHEE MCDM as mentioned in Sec. 4 with $q_1 = q_2 = 0$ and $p_1 = p_2 = 16$, the following pairwise comparisons are obtained:

$H_1(d_1)$	a_1	a_2	a_3	a_4
a_1	0	0	0	0
a_2	$\frac{3}{8}$	0	$\frac{1}{8}$	$\frac{1}{4}$
a_3	$\frac{1}{4}$	0	0	$\frac{1}{8}$
a_4	$\frac{1}{8}$	0	0	0

and

$H_2(d_2)$	a_1	a_2	a_3	a_4
a_1	0	0	0	$\frac{1}{16}$
a_2	$\frac{1}{4}$	0	0	$\frac{5}{16}$
a_3	$\frac{5}{16}$	$\frac{1}{16}$	0	$\frac{3}{8}$
a_4	0	0	0	0

Using equal and normalized weights $w_1 = w_2 = \frac{1}{2}$, the generalized preference indices and PROMETHEE flows are calculated:

$\Pi(a_k, a_l)$	a_1	a_2	a_3	a_4	$\phi^+(a_k)$	$\phi(a_k)$
a_1	0	0	0	$\frac{1}{32}$	$\frac{1}{96}$	$-\frac{1}{4}$
a_2	$\frac{7}{16}$	0	$\frac{1}{16}$	$\frac{9}{32}$	$\frac{25}{96}$	$\frac{1}{4}$
a_3	$\frac{9}{32}$	$\frac{1}{32}$	0	$\frac{1}{4}$	$\frac{3}{16}$	$\frac{1}{6}$
a_4	$\frac{1}{16}$	0	0	0	$\frac{1}{48}$	$-\frac{1}{6}$
$\phi^-(a_l)$	$\frac{25}{96}$	$\frac{1}{96}$	$\frac{1}{48}$	$\frac{3}{16}$		

Since $\phi(a_1)$ is the smallest a_1 is removed from the reference archive. Suppose that the α_i for the remaining alternatives are given by (i.e., these solutions have been found by applying tabu search with the corresponding α-vector):

$$\alpha_2 = (1, 0), \quad \alpha_3 = (0, 1), \quad \alpha_4 = \left(\frac{1}{2}, \frac{1}{2}\right).$$

According to formula (4.1) the new α-vector used in the tabu search is:

$$\alpha_{\text{new}} = \alpha_2 + \frac{2}{3}\alpha_3 - \frac{2}{3}\alpha_4 = \left(\frac{2}{3}, \frac{1}{3}\right). \tag{B.2}$$

The tabu search will now look for a solution using the function

$$f_{\text{TS}}(\mathbf{x}) = \frac{2}{3}f_1(\mathbf{x}) + \frac{1}{3}f_2(\mathbf{x}) = \frac{13}{3}x_1 + \frac{19}{3}x_2 + \frac{16}{3}x_3 \tag{B.3}$$

as its objective function.

Suppose this leads to the solution $x_{\text{new}} = (0, 1, 1)$. This solution, called $\tilde{a}_1$, then replaces a_1 in the reference archive:

	x_1	x_2	x_3	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$
$\tilde{\mathbf{a}}_\mathbf{1}$	**0**	**1**	**1**	**12**	**11**
a_2	1	1	0	11	10
a_3	1	0	1	9	11
a_4	0	1	0	7	5

where the changes were indicated in boldface.

Again performing a PROMETHEE MCDM on the archive, results into

$H_1(d_1)$	$\tilde{\mathbf{a}}_\mathbf{1}$	a_2	a_3	a_4
$\tilde{\mathbf{a}}_\mathbf{1}$	0	$\mathbf{\frac{1}{16}}$	$\mathbf{\frac{3}{16}}$	$\mathbf{\frac{5}{16}}$
a_2	**0**	0	$\frac{1}{8}$	$\frac{1}{4}$
a_3	**0**	0	0	$\frac{1}{8}$
a_4	**0**	0	0	0

and

$H_2(d_2)$	$\tilde{\mathbf{a}}_\mathbf{1}$	a_2	a_3	a_4
$\tilde{\mathbf{a}}_\mathbf{1}$	0	$\mathbf{\frac{1}{16}}$	0	$\mathbf{\frac{3}{8}}$
a_2	**0**	0	0	$\frac{5}{16}$
a_3	**0**	$\frac{1}{16}$	0	$\frac{3}{8}$
a_4	0	0	0	0

Hence, the following generalized preference indices and PROMETHEE flows are obtained:

$\Pi(a_k, a_l)$	$\tilde{\mathbf{a}}_\mathbf{1}$	a_2	a_3	a_4	$\phi^+(a_k)$	$\phi(a_k)$
$\tilde{\mathbf{a}}_\mathbf{1}$	0	$\mathbf{\frac{1}{16}}$	$\mathbf{\frac{3}{32}}$	$\mathbf{\frac{11}{32}}$	$\mathbf{\frac{1}{6}}$	$\mathbf{\frac{1}{6}}$
a_2	**0**	0	$\frac{1}{16}$	$\frac{9}{32}$	$\mathbf{\frac{11}{96}}$	$\mathbf{\frac{1}{12}}$
a_3	**0**	$\frac{1}{32}$	0	$\frac{1}{4}$	$\mathbf{\frac{3}{32}}$	$\mathbf{\frac{1}{24}}$
a_4	**0**	0	0	0	**0**	$-\mathbf{\frac{7}{24}}$
$\phi^-(a_l)$	**0**	$\mathbf{\frac{1}{32}}$	$\mathbf{\frac{5}{96}}$	$\mathbf{\frac{7}{24}}$		

leading to the conclusion that a_4 should be removed from the archive.

The new α-vector is calculated with $\tilde{\alpha}_1$, being given by (B.2):

$$\alpha_{\text{new}} = \frac{4}{7}\tilde{\alpha}_1 + \frac{2}{7}\alpha_2 + \frac{1}{7}\alpha_3 = \left(\frac{2}{3}, \frac{1}{3}\right) \tag{B.4}$$

which leads in this specific example to a convergence of the α-vector and therefore to the same $f_{\text{TS}}(\mathbf{x})$. The procedure then continues with a tabu search, and is repeated until the stopping criterion is met.

It should be remarked that this convergence of the α-vector is mainly due to the small size of the problem. In case this would happen in a realistic problem, the tabu search procedure would give rise to a different solution by exclusion of the previous one.

References

1. J. Knowles and D. Corne, Bounded Pareto archiving: Theory and practice, in eds. X. Gandibleux, M. Sevaux, K. Sörensen and V. T'Kindt, *Metaheuristics for Multiobjective Optimisation*, Lecture Notes in Economics and Mathematical Systems, Vol. 535 (Springer, 2004), pp. 39–64.
2. C. A. Mattson, A. A. Mullur and A. Messac, Smart Pareto filter: Obtaining a minimal representation of multiobjective design space, *Engineering Optimization* **36**(6) (2004) 721–740.
3. C. A. Coello Coello, Handling preferences in evolutionary multiobjective optimization: A survey, in *Proc. 2000 Congress on Evolutionary Computation, 2000*, Vol. 1 (IEEE, 2000), pp. 30–37.
4. C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd edn. (Springer, New York, 2007).
5. K. Deb and H. Gupta, Searching for robust pareto-optimal solutions in multi-objective optimization, in *Evolutionary Multi-Criterion Optimization*, Vol. 3410 (Springer, Berlin/ Heidelberg, 2005), pp. 150–164.
6. J. Branke, K. Deb, H. Dierolf and M. Osswald, Finding knees in multi-objective optimization, *The Eighth Conf. Parallel Problem Solving from Nature* (2004), pp. 722–731.
7. J. Branke, T. Kaußler and H. Schmeck, Guidance in evolutionary multi-objective optimization, *Advances in Engineering Software* **32** (2000) 499–507.
8. C. M. Fonseca and P. J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, in ed. *Proc. Fifth Int. Conf. Genetic Algorithms*, S. Forrest (Morgan Kauffman Publishers, 1993), pp. 416–423.
9. K. Deb, Solving goal programming problems using multi-objective genetic algorithms, *Congress on Evolutionary Computation* (IEEE, 1999), pp. 77–84.
10. C. M. Fonseca and P. J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms part I: A unified formulation, *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* **28** (1998) 26–37.
11. B. Rekiek, P. De Lit and A. Delchambre, Hybrid assembly line design and user's preferences, *Int. J. Prod. Research* **40**(5) (2000) 1095–1111.
12. R. F. Coelho, H. Bersini and Ph. Bouillard, Parametrical mechanical design with constraints and preferences: Application to a purge valve, *Computer Methods in Applied Mechanics and Engineering* **192** (2003) 4355–4378.
13. 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, *Congress on Evolutionary Computation* (IEEE Service Center, 1999), pp. 14–21.
14. D. Cvetković and I. C. Parmee, Genetic algorithm based multi-objective optimization and conceptual engineering design, *Congress on Evolutionary Computation CEC99* (IEEE, 1999), pp. 29–36.
15. D. Cvetković and I. C. Parmee, Designer's preferences and multi-objective preliminary design processes, in *Proc. Fourth Int. Conf. Adaptive Computing in Desingn and Manufacture (ACDM '2000)*, ed. I. C. Parmee (Springer London, 2000), pp. 249–260.

16. Y. Jin and B. Sendhoff, Incorporation of fuzzy preferences into evolutionary multi-objective optimization, in *Proc. 4th Asia Pacific Conf. Simulated Evolution and Learning* (2002), pp. 26–30.
17. D. Cvetković and I. C. Parmee, Use of preferences for GA-based Multi-objective Optimisation, in *Proc. Genetic and Evolutionary Computation Conf.*, Orlando, Florida, US (1999), pp. 13–14.
18. G. W. Greenwood, X. S. Hu and J. G. D'Ambrosio, Fitness functions for multiple objective optimization problems: Combining preferences with pareto rankings, in eds. *Foundations of Genetic Algorithms 4*, R. K. Belew and M. D. Vose, (Morgan Kaufmann 1997), pp. 437–455.
19. J. Molina, L. V. Santana, A. G. Hernández-Díaz, C. A. Coello Coello and R. Caballero, g-dominance: Reference point based dominance for multiobjective metaheuristics, *European Journal of Operational Research* **197**(2) (2009) 685–692.
20. A. P. Wierzbicki, The use of reference objectives in multiobjective optimization, in *Multiple Criteria Decision Making Theory and Application*, eds. G. Fandel and T. Gal, (Springer-Verlag, New York, 1980), pp. 469–486.
21. J. P. Brans, L'ingénierie de la décision, elaboration d'instruments d'aide à la décision. Méthode PROMETHEE, in *L'aide a la Décision: Nature, Instruments et Perspectives d'avenir*, eds. R. Nadeau and M. Landry, Presses de l'Université Laval, Québec, Canada, 1982, pp. 183–214 (in french).
22. J.-P. Brans and B. Mareschal, PROMETHEE methods, in eds. *Multiple Criteria Decision Analysis — State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott (Kluwer, Boston, 2005), pp. 163–195.
23. J.-P. Brans and P. Vincke, A preference ranking organisation method: The PROMETHEE method for MCDM, *Management Science* **31** (1985) 647–656.
24. C. Wilbaut, S. Hanafi and S. Salhi, A survey of effective heuristics and their application to a variety of knapsack problems, *IMA Journal of Management Mathematics* **19**(3) (2008) 227.
25. C. Bazgan, H. Hugot and D. Vanderpooten, Solving efficiently the 0-1 multi-objective knapsack problem, *Computers & Operations Research* **36**(1) (2009) 260–279.
26. F. Glover, Tabu search–part I, *ORSA Journal on Computing* **50**(3) (1989) 190–206.
27. X. Gandibleux and A. Freville Tabu search based procedure for solving the 0-1 multi-objective knapsack problem: The two objectives case, *Journal of Heuristics* **6**(3) (2000) 361–383.
28. R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria (2009), Available at http://www.R-project.org.