Pareto Simulated Annealing—A Metaheuristic Technique for Multiple-Objective Combinatorial Optimization

PIOTR CZYŻAK and ANDRZEJ JASZKIEWICZ
Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, Poland

ABSTRACT

This paper presents a multiple-objective metaheuristic procedure—Pareto simulated annealing. The goal of the procedure is to find in a relatively short time a good approximation of the set of efficient solutions of a multiple-objective combinatorial optimization problem. The procedure uses a sample, of so-called generating solutions. Each solution explores its neighbourhood in a way similar to that of classical simulated annealing. Weights of the objectives, used for their local aggregation, are tuned in each iteration in order to assure a tendency for approaching the efficient solutions set while maintaining a uniform distribution of the generating solutions over this set. A computational experiment shows that the method is a better tool for approximating the efficient set than some previous proposals. © 1998 John Wiley & Sons, Ltd.

J. Multi-Crit. Decis. Anal., **7**: 34–47 (1998)

KEY WORDS: multiple-objective combinatorial optimization; metaheuristic procedures

1. INTRODUCTION

Many real-life problems are combinatorial, i.e. they concern a choice of the best solution from a finite but large set of feasible solutions (Nemhauser and Wolsey, 1988). It is also a well-known fact that the solutions of real-life problems are often evaluated from several points of view which may be described by different objectives (Steuer, 1986; Roy and Bouyssou, 1993). These facts were reasons for significant research efforts in the fields of *combinatorial optimization* (CO) and *multiple-objective decision making* (MODM). They resulted in many practical applications of methods developed in each of the two fields. Nevertheless, surprisingly few theoretical works concern *multiple-objective combinatorial optimization* (MOCO) problems; see e.g. the survey by Ulungu and Teghem (1994). Some exceptions are *multiple-objective shortest path* problems (Ulungu and Teghem, 1991; Current and Marsh, 1993) and *multiple-objective project-scheduling* problems (Słowiński, 1981, 1989). As a consequence, very few practical applications of MODM methods for combinatorial problems are reported in the OR literature, see e.g. White (1990) and Ulungu and Teghem (1994) for lists of references.

In our opinion the relatively small number of applications of MOCO is not due to the fact that combinatorial problems rarely require multiple objectives but due to the notable difficulty of such problems. Indeed, different objectives are often used in particular classes of combinatorial problems. For example, in vehicle routing the typically used objectives are total cost, distance, number of vehicles and travel time (Assed, 1988) and in project scheduling the typical objectives are net present value, project completion time, mean weighted delay, number of delayed tasks, mean weighted flow time and total resource utilization (Słowiński, 1981, 1989). The objectives, however, are usually used separately or they are combined into a single objective.

The difficulty of MOCO problems results from the following two factors.

- Solving an MOCO problem requires intensive co-operation with the decision maker (DM); this results in especially high requirements for effective tools used to generate efficient solutions.
- Many combinatorial problems are hard even in single-objective versions; their multiple-objective versions are frequently more difficult.

An MODM problem is ill-posed from the mathematical point of view because, except in trivial cases, it has no optimal solution. The goal of MODM methods is to find a solution most consistent with the DM's preferences, i.e. the *best compromise*. Under very weak assumptions about

CCC 1057–9214/98/010034–14$17.50
© 1998 John Wiley & Sons, Ltd

Received 1 November 1995
Accepted 21 February 1997

the DM's preferences the best compromise solution belongs to the set of *efficient* solutions (Rosenthal, 1985). Interactive procedures, which have become especially popular in recent years, generate efficient solutions in computational phases alternating with phases of decision. Such procedures, however, can only be used if sufficiently effective tools for finding efficient solutions are available.

Furthermore, many single-objective combinatorial problems belong to the class of NP-hard problems. Generation of efficient solutions in an MOCO problem is of course not easier than finding solutions optimizing particular objectives and in many cases is even harder. For example, the single-objective shortest path problem is one of the simplest combinatorial problems, while the corresponding multiple-objective problem is NP-hard (Garey and Johnson, 1979). Thus even problems for which relatively efficient single-objective exact methods are known may be difficult if multiple objectives have to be considered.

Tools used for generation of efficient solutions in MOCO, like single-objective optimization methods, may be classified into one of the following categories:

- exact procedures;
- specialized heuristic procedures;
- metaheuristic procedures.

The main disadvantage of exact algorithms lies in their high computational complexity. As a result, only limited classes of real-life MOCO problems may be solved exactly.

The main weak point of specialized procedures, both heuristic and exact, is their inflexibility. This factor seems to be especially important in the case of MOCO. For example, many MODM methods start by presenting to the DM the ideal point, which is found by independent optimization of particular objectives. If the objective functions have different mathematical forms, their optimization may require different specialized procedures. Furthermore, MODM methods use various tools for generating efficient solutions. Some of them apply Geoffrion's theorem (e.g. Steuer, 1986), others use penalty functions (e.g. Charnes and Cooper, 1977), utility functions (Keeney and Raiffa, 1976), ε-constraints (e.g. Steuer, 1986) or achievement-scalarizing functions (Wierzbicki, 1986). Of course, each of the tools may require individual specialized procedures. As a result, solving a given MOCO problem may involve several specialized procedures and it may be difficult to change the formulation of the model, the family of objectives or even the MODM method used to solve the problem.

Consequently, it is not only the opinion of the authors (cf. e.g. Ulungu and Teghem, 1994) that the most promising practical approach to MOCO consists of generating efficient solutions with metaheuristic procedures.

In recent years several metaheuristic methods for single-objective combinatorial problems have been proposed (e.g. Pirlot, 1993). The methods are called metaheuristics because they define only a 'skeleton' of the optimization procedure that has to be customized for particular applications. This class of methods includes simulated annealing (SA) (Cerny, 1985; Kirkpatrick *et al.*, 1983; Laarhoven and Aarts, 1987), tabu search (Glover, 1989) and genetic algorithms (Goldberg, 1988). They allow the finding of nearly optimal solutions for a wide class of combinatorial problems in a relatively short time. The methods have been successfully applied to many problems that cannot be solved exactly in polynomial time.

As most MODM methods use single-objective optimization as a tool for generating efficient solutions, it seems rational to apply the classical single-objective metaheuristic procedures in MOCO. Yet such procedures may appear too inefficient for practical applications.

Single-objective metaheuristic procedures are relatively easy to apply if the DM expresses his/her preferences *a priori* and the information about the DM's preferences is used to build a functional preference model, e.g. a utility function (Keeney and Raiffa, 1976). A metaheuristic procedure can be used in order to optimize this function on the set of feasible solutions. There is, however, both theoretical (Roy and Bouyssou, 1993) and empirical (Corner and Buchanan, 1994) evidence that methods with *a priori* expression of preferences perform relatively poorly in the case of multiple-objective mathematical programming.

In general the single-objective metaheuristic procedures may also be used if the DM is interested in interactive analysis of the set of solutions. Interactive methods consist of computational phases alternating with phases of decision. In each computational phase a solution or a sample of solutions, usually efficient, is generated. The solutions are usually generated by solving substitute single-objective problems. Optimal solutions of the substitute problems are efficient in the original multiple-objective problem (Wierzbicki, 1986). Single-objective metaheuristic procedures can be used in order to solve the substitute problems. Interactive methods, however, may only be used if the time needed for the computational phase is acceptable to the DM. Although metaheuristic procedures are relatively effective tools for combinatorial optimization, they might not be effective enough for use in interactive procedures, especially in the case of methods generating samples of solutions.

This paper proposes a metaheuristic procedure, called Pareto simulated annealing (PSA), for MOCO problems. The goal of the procedure is to find a set of solutions that is a good approximation of the efficient solution set. The set of solutions obtained by the procedure may then be presented to the DM. If the set is small enough, the DM may select the best compromise from it. Otherwise, the DM may explore the set of solutions guided by an interactive procedure for problems with explicitly given sets of alternatives.

The paper is organized in the following way. In Section 2 a formal statement of the MOCO problem is presented. Some previous proposals of multiple-objective metaheuristic procedures and their weaknesses are described in Section 3. In Section 4 the PSA procedure is presented. Evaluation of multiple-objective metaheuristic procedures is discussed in Section 5. In Section 6 a number of computational experiments are reported. Finally, the main features of the procedure are summarized and some possible directions of further research are described in Section 7.

2. PROBLEM STATEMENT

The general MOCO problem is formulated as

$$\max\{f_1(\mathbf{x}) = z_1, \ldots, f_J(\mathbf{x}) = z_J\}$$
$$\text{s.t.} \quad \mathbf{x} \in D$$

where the *solution* $\mathbf{x} = [x_1, \ldots, x_I]$ is a vector of discrete *decision variables* and D is the set of feasible solutions.

A solution $\mathbf{x} \in D$ is *efficient* (*Pareto-optimal*) if there is no $\mathbf{x}' \in D$ such that $\forall j\ f_j(\mathbf{x}') \geqslant f_j(\mathbf{x})$ and $f_j(\mathbf{x}') > f_j(\mathbf{x})$ for at least one j. The set of all efficient solutions is denoted by N.

An efficient solution $\mathbf{x}$ is *supported* if there exists a vector of non-negative weights $\boldsymbol{\Lambda} = [\lambda_1, \ldots, \lambda_J]$ such that $\mathbf{x}$ is the unique global optimum of the problem

$$\max \sum_{j=1}^{J} \lambda_j f_j(\mathbf{x})$$
$$\text{s.t.} \quad \mathbf{x} \in D$$

3. REVIEW OF EXISTING MULTIPLE-OBJECTIVE METAHEURISTIC PROCEDURES

Several multiple-objective metaheuristic procedures have already been proposed. The goal of the procedures is to find a sample of feasible solutions that is a good approximation to the efficient solutions set. Such procedures may be used in the first phase of methods with *a posteriori* articulation of preferences or in computational phases of interactive procedures generating samples of solutions.

Several multiple-objective methods based on genetic algorithms have already been proposed; see e.g. Schaffer (1984), Srinivas and Deb (1994) and Fonseca and Fleming (1995) for reviews. According to our knowledge, the methods have been tested, however, on continuous or very small discrete problems only. Thus the usefulness of these procedures in the case of MOCO is yet to be tested.

Serafini (1992) and Fortemps *et al.* (1994) have independently proposed very similar algorithms based on simulated annealing and suggested their use in MOCO. The outcome of their algorithms is not a single solution but a set M of *potentially efficient solutions*, i.e. the set composed of solutions efficient with respect to all generated solutions. The set is updated whenever a new solution non-dominated with respect to the current one is generated. Updating set M with a new solution $\mathbf{x}$ consists of

- adding $\mathbf{x}$ to M if there is no other solution $\mathbf{v} \in M$ such that $\mathbf{v}$ dominates $\mathbf{x}$,
- removing from set M all solutions dominated by $\mathbf{x}$.

The algorithms of the two above-mentioned proposals are very close to that of single-objective SA. The general scheme of the procedures is given below.

```
Select a starting solution $\mathbf{x} \in D$
$M := \varnothing$
Update the set $M$ of potentially efficient solutions with $\mathbf{x}$
$T := T_0$
repeat
    Construct $\mathbf{y} \in V(\mathbf{x})$                    (1)
    if $\mathbf{y}$ is not dominated by $\mathbf{x}$ then
        Update the set $M$ of potentially efficient solutions with $\mathbf{y}$
    $\mathbf{x} := \mathbf{y}$ (accept $\mathbf{y}$) with probability $P(\mathbf{x}, \mathbf{y}, T, \mathbf{\Lambda})$
    if the conditions of changing the temperature are fulfilled then
        decrease $T$
until the stop conditions are fulfilled
```

where $V(\mathbf{x}) \subseteq D$ is the neighbourhood of solution $\mathbf{x}$, i.e. the set of feasible solutions that may be reached from $\mathbf{x}$ by making a simple move, T_0 is the initial temperature, T is the actual temperature, $\mathbf{\Lambda} = [\lambda_1, \ldots, \lambda_J]$ is a vector of weights and $P(\mathbf{x}, \mathbf{y}, T, \mathbf{\Lambda})$ is the acceptance probability.

The probability of accepting a new solution is calculated in a different way than in single-objective SA. In the case of single-objective SA a new solution is accepted with probability equal to one if it is not worse than the current solution. Otherwise, it is accepted with probability less than one. In the case of multiple objectives, one of the following three exclusive situations may occur while comparing a new solution $\mathbf{y}$ with the current one $\mathbf{x}$:

- $\mathbf{y}$ dominates or is equal to $\mathbf{x}$;
- $\mathbf{y}$ is dominated by $\mathbf{x}$;
- $\mathbf{y}$ is non-dominated with respect to $\mathbf{x}$.

In the first situation the new solution may be considered as not worse than the current one and accepted with probability equal to one. In the second situation the new solution may be considered as worse than the current one and accepted with probability less than one. Serafini (1992) and Fortemps *et al.* (1994) have proposed several multiple-objective rules for acceptance probability which in different ways treat the third situation.

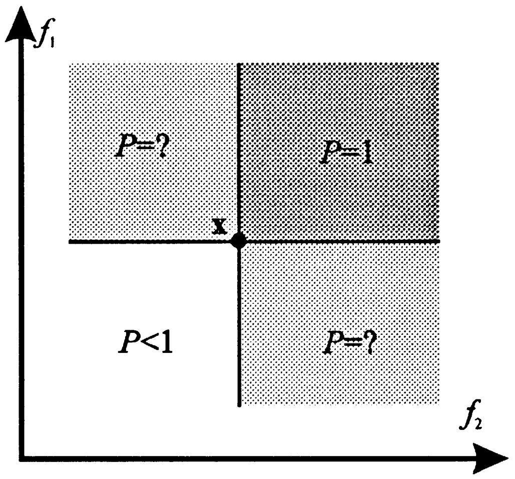


Figure 1. General idea of multiple-objective rules for acceptance probability P

Figure 1 presents the general idea of the multiple-objective rules for acceptance probability in the case of two maximized objectives. Some characteristic rules are described below.

Rule *C* may be seen as a local aggregation of all objectives with an achievement-scalarizing function based on the Chebyshev metric with the reference point $\mathbf{x}$. It is defined by the expression

$$P(\mathbf{x}, \mathbf{y}, T, \mathbf{\Lambda}) = \min\left\{1, \exp\left(\max_j\{\lambda_j(f_j(\mathbf{x}) - f_j(\mathbf{y}))/T\}\right)\right\}$$

A graphical illustration of the rule is given in Figure 2.

Rule *SL* may be seen as a local aggregation of all objectives with a weighted sum of the objectives. It is defined by the expression

$$P(\mathbf{x}, \mathbf{y}, T, \mathbf{\Lambda}) = \min\left\{1, \exp\left(\sum_{j=1}^{J} \lambda_j(f_j(\mathbf{x}) - f_j(\mathbf{y})/T)\right)\right\}$$

A graphical illustration of the rule is given in Figure 3.

Some tests which were performed with procedure (1) resulted in the observation that the procedure works well enough for relatively small problems only. In the case of larger problems the set of potentially efficient solutions may represent a small region of set N. This observation is illustrated by the results of the experiments

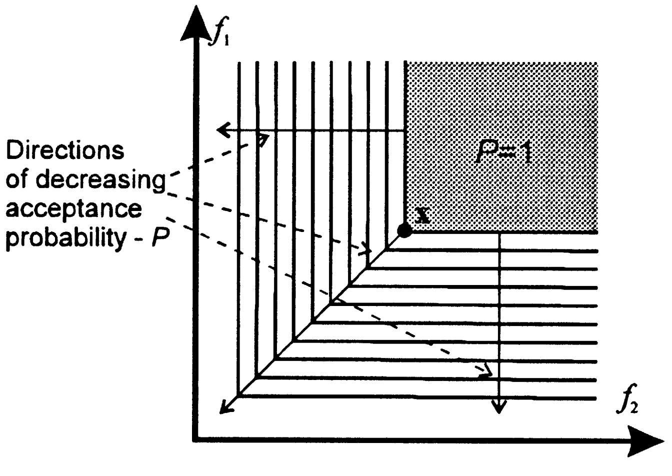


Figure 2. Graphical illustration of rule C

described in Section 6. It was the reason for developing the PSA procedure, which tends to generate a good approximation of the whole set N even for relatively large MOCO problems.

4. DESCRIPTION OF PARETO SIMULATED ANNEALING

4.1. Basic concepts

PSA uses several ideas known from simulated annealing (Kirkpatrick *et al.*, 1983; Laarhoven and Aarts, 1987):

- the concept of neighbourhood
- acceptance of new solutions with some probability;
- dependence of the probability on a parameter called the temperature;
- the scheme of the temperature changes.

PSA, however, uses a sample (population) of interacting solutions at each iteration. The solutions are called *generating solutions*. Among metaheuristic procedures, this concept is used in genetic algorithms (Goldberg, 1988).

Another new idea used in PSA is to control the objective weights used in the multiple-objective

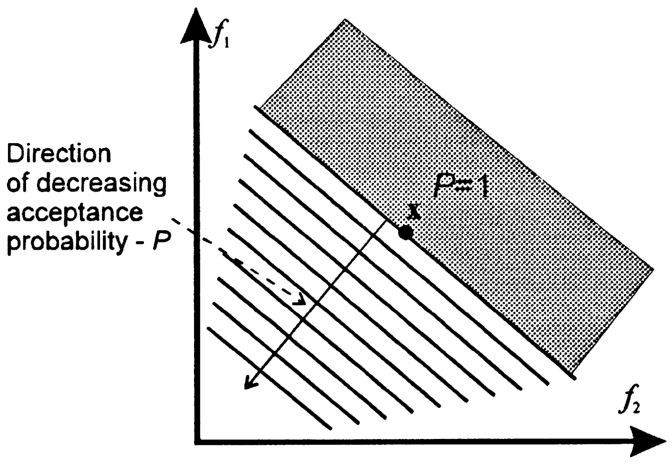


Figure 3. Graphical illustration of rule SL

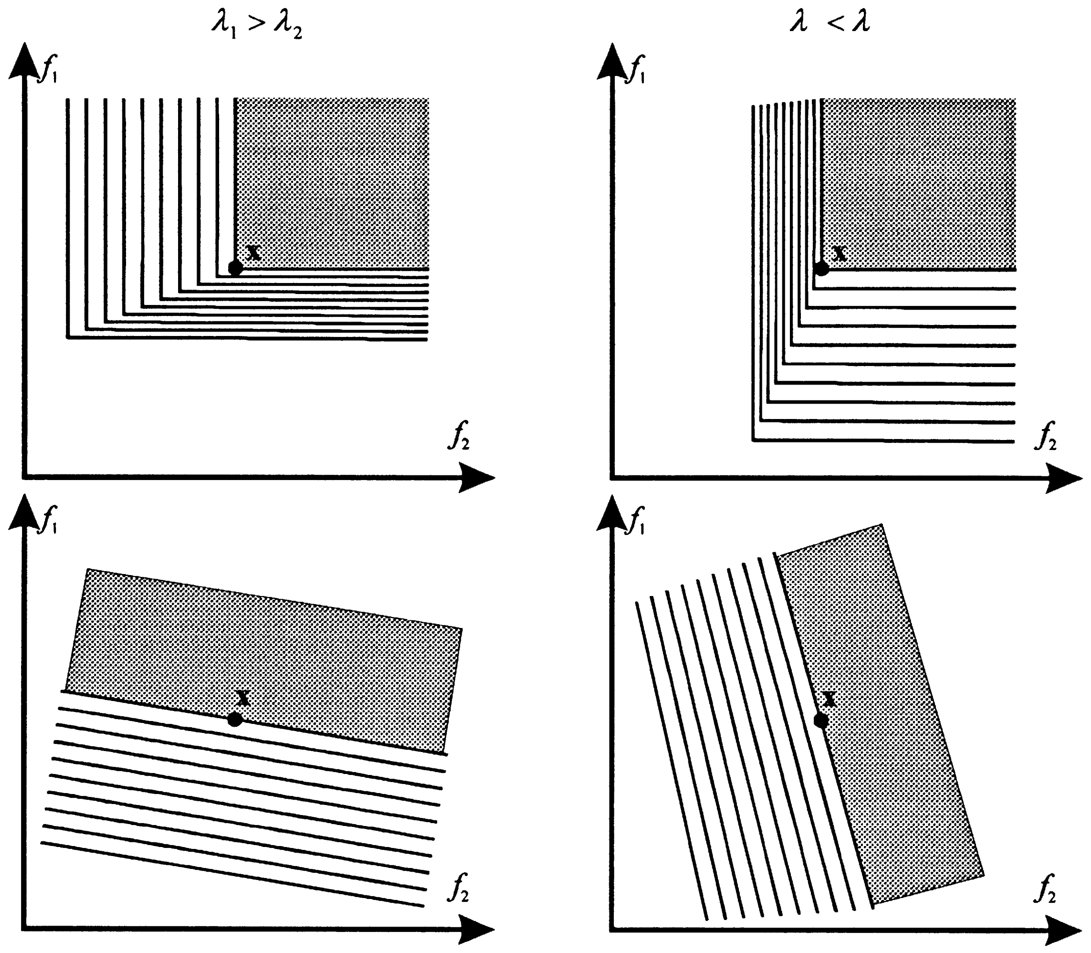


Figure 4. Role of weights in multiple-objective rules for acceptance probability

rules for acceptance probability in order to assure dispersion of the generating solutions over the whole set of efficient solutions. Please note that the higher the weight associated with a given objective, the lower is the probability of accepting moves that decrease the value on this objective and the greater is the probability of improving the value of this objective. Thus, by controlling the weights, one can increase or decrease the probability of improving values of the particular objectives. This fact is illustrated graphically in Figure 4.

4.2. Algorithm of PSA procedure

The general scheme of the PSA procedure may be written as follows.

```
Select a starting sample of generating solutions $S \subset D$
*for each* $\mathbf{x} \in S$ *do*
    Update the set $M$ of potentially efficient solutions with $\mathbf{x}$

$T := T_0$

*repeat*
    *for each* $\mathbf{x} \in S$ *do*
        Construct $\mathbf{y} \in V(\mathbf{x})$
        *if* $\mathbf{y}$ is not dominated by $\mathbf{x}$ *then*
            Update the set $M$ of potentially efficient solutions with $\mathbf{y}$
        Select the solution $\mathbf{x}' \in S$ closest to $\mathbf{x}$ and
        non-dominated with respect to $\mathbf{x}$
            *if* there is no such solution $\mathbf{x}'$ or it is the
            first iteration with $\mathbf{x}$ *then*
                Set random weights such that
                $\forall j\, \lambda_j \geqslant 0$ and $\sum_j \lambda_j = 1$
            *else*
                *for each* objective $f_j$
                    $\lambda_j = \begin{cases} \alpha\lambda_j^{\mathbf{x}} & \text{if } f_j(\mathbf{x}) \geqslant f_j(\mathbf{x}') \\ \lambda_j^{\mathbf{x}}/\alpha & \text{if } f_j(\mathbf{x}) < f_j(\mathbf{x}') \end{cases}$
                normalize the weights such that
                $\Sigma_j\, \lambda_j = 1$
        $\mathbf{x} := \mathbf{y}$ (accept $\mathbf{y}$) with probability
        $P(\mathbf{x}, \mathbf{y}, T, \boldsymbol{\Lambda})$
    *if* the conditions of changing the temperature
    are fulfilled *then*
        decrease $T$
*until* the stop conditions are fulfilled
```

where $\boldsymbol{\Lambda}^{\mathbf{x}} = [\lambda_1^{\mathbf{x}}, \ldots, \lambda_J^{\mathbf{x}}]$ is the weighting vector used in the previous iteration for solution $\mathbf{x}$, $\alpha > 1$ is a constant close to one (e.g. $\alpha = 1.05$) and $P(\mathbf{x}, \mathbf{y}, T, \boldsymbol{\Lambda})$ is one of the multiple-objective rules for acceptance probability described above.

In each iteration of the procedure a sample of solutions, called the *generating sample*, is used. The main idea of PSA is to assure a tendency for approaching the set of efficient solutions as well as an inclination for dispersing the solutions constituting the generating sample over the whole set N. As a result, each solution tends to investigate a specific region of set N.

The tendency for approaching the set of efficient solutions is assured by using one of the above-mentioned multiple-objective rules for acceptance probability. The inclination for dispersing the solutions from the generating sample over the whole set N is obtained by controlling the weights of particular objectives used in these rules. For a given solution $\mathbf{x} \in S$ the weights are changed in order to increase the probability of moving it away from its closest neighbour in S denoted by $\mathbf{x}'$. This is obtained by increasing the weights of the objectives on which $\mathbf{x}$ is better than $\mathbf{x}'$ and decreasing the weights of the objectives on which $\mathbf{x}$ is worse than $\mathbf{x}'$. An intuitive explanation of this approach is given in the Appendix.

Please note that the algorithm of PSA is essentially parallel, because calculations required for each solution from S, i.e. construction of a new solution from its neighbourhood, setting the weights and accepting the new solution, may be done on different processors.

It is also worth mentioning that one of the crucial points of the procedure from the point of view of its effectiveness is the updating of the set of potentially efficient solutions. A data structure called a quad tree allows for very effective implementation of this step (Finkel and Bentley, 1974; Habenicht, 1982).

PSA is not a complete method for solving MOCO problems but just a tool for generating an approximation to the set of efficient solutions. One can use the following two-stage method for finding the best compromise of an MOCO problem.

1. Generation of an approximation M of set N with PSA.
2. Selection of the best solution $\bar{\mathbf{x}}$ from set M with an interactive procedure for problems with explicitly given sets of alternatives.

The second stage requires co-operation with the DM. In this stage the DM learns the possible trade-offs as well as his/her preferences and selects the best compromise. As mentioned before, he/she may require some support in this phase. One of the interactive methods for multiple-criteria problems with a large but finite set of alternatives may be used to support the DM.

5. EVALUATION OF MULTIPLE-OBJECTIVE METAHEURISTIC PROCEDURES

Since the goal of multiple-objective metaheuristic procedures is to find a good approximation of set N, it is important to have some evaluation technique allowing for comparison of different approximations.

A solution obtained by a single-objective optimization method may be evaluated by comparison with randomly generated solutions, with a solution obtained by another method or with some reference solution, e.g. the global optimum or the best solution known so far. Analogously, in the multiple-objective case a set of potentially efficient solutions may be compared with randomly generated solutions, with another solution set obtained by a different method or with some reference set R of solutions (e.g. the set of efficient solutions or the best approximation known so far). The quality metrics described below concern the last case.

It may seem natural to use as a quality metric the percentage of reference solutions found, defined as

$$\frac{\text{card}\{M \cap R\}}{\text{card}\{R\}} \times 100\%$$

This measure has, however, significant disadvantages. In the case of real-size MOCO problems it may be impossible to obtain in a reasonable time a significant percentage of efficient solutions. It is more important to obtain in a short time solutions close to the efficient ones. For example, considering the two approximations presented in Figures 5(a) and 5(b), it is natural to evaluate the first one as better. This quality measure, however, will have a value of zero in both cases.

The other disadvantage may be visualized by considering the approximations presented in Figures 5(a) and 5(c). The second approximation contains some percentage of reference solutions, but all potentially efficient solutions are concentrated in a small region of set R. Many reference solutions have no representation in set M. In the first case for each reference solution there is a relatively close solution in set M. Thus the first approximation gives the DM much better information about the shape of the efficient set and should be evaluated as better than the other one. The percentage of reference solutions found is, however, greater in the second case.

In order to avoid the above-mentioned disadvantages, two quality metrics which tend to measure the closeness of set M to the reference set are proposed. We assume that set M is a good approximation of set R if it gives the DM important information about all regions of set R or, in other words, if for each solution $\mathbf{y} \in R$ there is a close solution $\mathbf{x} \in M$. We propose to measure the closeness of two solutions by the following metric based on the achievement-scalarizing function:

$$c(\mathbf{x}, \mathbf{y}) = \max_{j=1,\ldots,J} \{0, w_j(f_j(\mathbf{y}) - f_j(\mathbf{x}))\}$$

Thus the measure takes the value zero if on all objectives $\mathbf{x}$ reaches the value of solution $\mathbf{y}$. Otherwise, it takes the value of the maximal weighted deviation from $\mathbf{y}$ on particular objectives.

The weights used in the above expression are set as

$$w_j = 1/\Delta_j$$

where Δ_j is the range of objective f_j in the reference set.

The first metric is defined as

$$Dist1 = \frac{1}{\text{card}\{R\}} \sum_{\mathbf{y} \in R} \left\{ \min_{\mathbf{x} \in M} \{c(\mathbf{x}, \mathbf{y})\} \right\}$$

while the second metric is defined as

$$Dist2 = \max_{\mathbf{y} \in R} \left\{ \min_{\mathbf{x} \in M} \{c(\mathbf{x}, \mathbf{y})\} \right\}$$

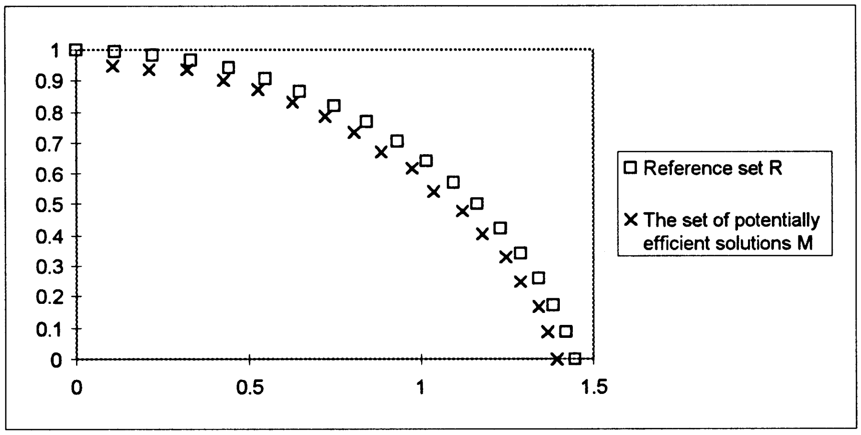
The first metric gives information about the average distance from $\mathbf{y} \in R$ to the closest solution in M, while the second metric gives information about the worst case. The lower the values, the better set M approximates set R. Moreover, the lower the ratio *Dist2*/*Dist1*, the more uniform the distribution of solutions from set M over set R.

6. COMPUTATIONAL EXPERIMENTS

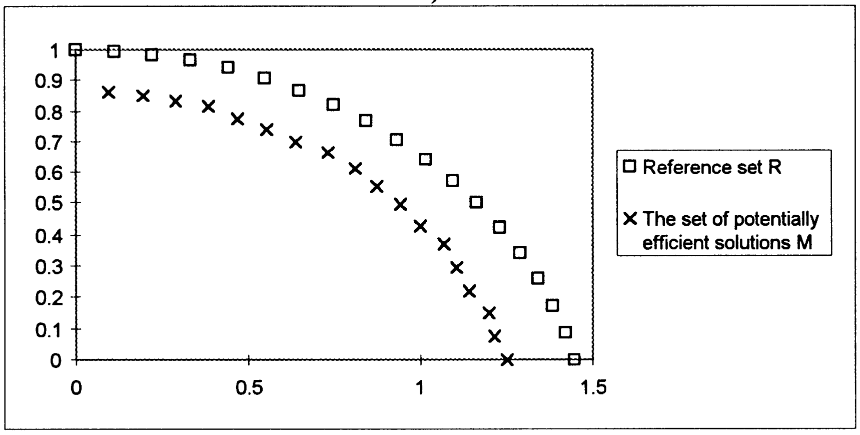
In order to evaluate the PSA method, some computational experiments have been performed. In the experiments the multiple-objective knapsack problem has been used. The multiple-objective knapsack problem seems to be a good test problem because, while being an NP-complete problem, it allows for relatively simple generation of a good approximation of the Pareto solution set in terms of the supported solution set. This set could play the role of the reference set R.

The goals of the experiments were as follows:

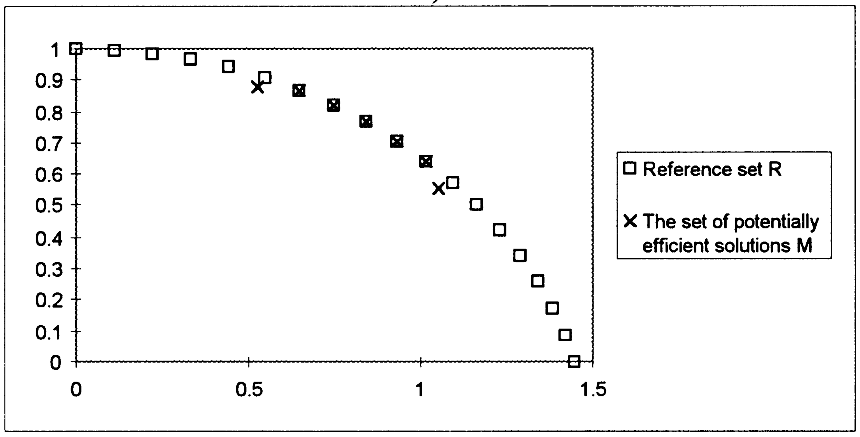
- to compare the effectiveness of the PSA method with the methods of Serafini (1992) and Fortemps *et al.* (1994);
- to find some general pattern of behaviour of the method while changing the size of the generating sample, the number of moves and the type of implementation (sequential or parallel).



a)



b)



c)

Figure 5. Three different approximations of a reference set *R*

6.1. Experiment design

A multiple-objective knapsack (MOK) problem is formulated in the following way. Given a set $E = \{e_1, \ldots, e_L\}$ of elements, a set $G = \{g_1, \ldots, g_L\}$ of element weights and sets $C_1 = \{c_{11}, \ldots, c_{1L}\}, \ldots, C_J = \{c_{J1}, \ldots, c_{JL}\}$ of element values, find a set $E' \subseteq E$ that maximizes the following objectives:

$$\max \sum_{e_i \in E'} c_{1i}$$

$$\vdots$$

$$\max \sum_{e_i \in E'} c_{Ji}$$

$$\text{s.t.} \quad \sum_{e_i \in E'} g_i \leqslant K$$

where K is the knapsack size.

The MOK problem has already been used by Fortemps *et al.* (1994) for evaluation of their procedure. The problem is NP-complete (but not in the strong sense) even in the single-objective case. However, on average it can be solved in a relatively short time. Thus in the multiple-objective case it is relatively simple to obtain supported efficient solutions. Please note that it is not our goal to recommend the use of PSA in order to solve the MOK problem. We have selected this problem for our tests because there are other more efficient methods for finding efficient solutions that allow us to obtain good reference sets even for relatively large MOK problems.

In the bi-objective case the reference sets were composed of all supported efficient solutions generated by applying the first phase of the algorithm proposed by Ulungu and Teghem (1995). In the case of two objectives it is possible to generate all, supported and non-supported, efficient solutions (Ulungu and Teghem, 1995). Computational and memory requirements are, however, very high. We were not able to obtain all non-supported efficient solutions even for problems with 100 elements. In the case of three and more objectives the reference sets were obtained by generating 1000 supported efficient solutions with random weights in the linear combination of objective functions. Redundant solutions were then removed.

The test problems were designed such that the weights and prices of the elements were generated randomly from the range [60, 100]. The knapsack size was set equal to 50% of the total weight of elements.

The tests were done for two-, three- and four-objective problems. The number of elements was set equal to 100, 200, 400 and 800. The number of moves on each temperature level was set equal to 128, 256, 512, 1024 and 2048. For each problem size, 10 different problems were generated. For each particular problem and for each particular setting of the PSA parameters, five runs of the PSA method were made.

6.2. Customization of PSA method to multiple-objective knapsack problem

The following decisions have been made in order to customize the PSA method to this particular problem.

The neighbourhood solutions of solution **x** were generated by the following algorithm:

```
repeat
    remove a randomly selected element from the knapsack
until there is a free space in the knapsack for the
    biggest element that is outside the knapsack
repeat
    insert a randomly selected element into the knapsack
until there is no free space for any element that is
    outside the knapsack.
```

Exactly the same neighbourhood was used in the study of Fortemps *et al.* (1994).

The temperature was decreased after making a given number of moves (accepted or not). The rate of decreasing the temperature was set equal to 0.9. The starting temperature was set equal to 50, which assured that more than 80% of moves were accepted. The procedure was stopped after obtaining the temperature below unity at which less than 5% of moves were accepted.

On a PC with a Pentium 100 CPU a time of about 2 s was necessary to perform 1000 moves for problems with four objectives and 800 elements.

6.3. Results

Because of limited space, only some of the obtained results are presented. All the test problems, reference sets and results are available from the authors upon request.

In order to evaluate the results, we use the *Dist1* and *Dist2* metrics. The percentage of reference solutions found was in most cases equal to zero and was never greater than 5%; 90%–100% (in most cases 100%) of generated solutions were dominated by some reference solutions.

It was observed that the rules *C* and *SL* give similar results. All the results presented below are obtained with rule *SL*.

Table I contains the average results of five runs obtained for 10 three-objective problems with 200 elements for different sizes of the generating sample. On each level of the temperature, 512 moves were made, so the total number of moves was equal to 19,456. A sequential implementation was assumed, i.e. the total number of moves made by all generating solutions was constant and the time of calculation was approximately the same for each size of the generating sample. Thus, the larger the sample, the fewer were the moves made with each single generating solution. For example, when the generating sample was composed of a single solution, 512 moves were made with it. When the generating sample was composed of 16 solutions, 32 moves were made with each of them. Thus two factors influence the values of the quality measures as the size of the generating sample is changed.

- The larger the sample, the larger is the probability that the generating solutions will be dispersed over the different regions of the non-dominated set.
- The larger the sample, the fewer are the steps made with each of the solutions and the larger is the probability that many of them will not reach the non-dominated set.

One can observe that the influence of the two factors gives relatively good results for samples composed of eight and 16 solutions, while for smaller and larger sizes the results are worse. In particular, PSA can give better results than the method of Serafini (size of the generating sample equal to one) even in the case of sequential implementation if the size of the generating sample is appropriately set. One can also observe that the ratio *Dist2*/*Dist1* decreases with increasing size of the generating sample. This means that for larger generating samples a more uniform distribution of potentially efficient solutions over the reference set is obtained.

Similar results were observed for problems with different numbers of objectives and elements. For example, Table II contains results of the same experiment done with four-objective problems with 800 elements. On each level of the temperature, 2048 moves were made, so the total number of moves was equal to 77,824.

Tables III and IV show the influence of the number of moves made on each temperature level when the sequential implementation is assumed. One can see that with an increase in the number of moves an improvement in the results is obtained. In the case of the method of Serafini, which uses a single generating solution, a relatively small improvement is obtained. This suggests that the method does not generate a good representation of all regions of the efficient set and increasing the number of moves does not change this situation

Table I. Results obtained for three-objective problems, 200 elements, sequential implementation

Size of generating sample	*Dist1*	*Dist2*	*Dist2*/*Dist1*
1 (Serafini's algorithm)	0.217	0.481	2.212
2	0.253	0.681	2.690
4	0.206	0.440	2.132
8	0.184	0.397	2.164
16	0.180	0.365	2.026
32	0.214	0.425	1.984
64	0.295	0.518	1.760
128	0.413	0.569	1.377

Table II. Results obtained for four-objective problems, 800 elements, sequential implementation

Size of generating sample	*Dist1*	*Dist2*	*Dist2*/*Dist1*
1 (Serafini's algorithm)	0.417	0.722	1.733
2	0.316	0.705	2.233
4	0.308	0.613	1.988
8	0.274	0.573	2.093
16	0.280	0.509	1.816
32	0.322	0.515	1.599
64	0.406	0.559	1.378
128	0.535	0.631	1.181

Table III. Values of *Dist1* metric obtained for three-objective problems, 200 elements, sequential implementation, different numbers of moves

Size of generating sample	Number of moves					
	128	256	512	1024	2048	4096
1 (Serafini's algorithm)	0.292	0.245	0.218	0.192	0.179	0.172
2	0.284	0.270	0.253	0.241	0.226	0.207
4	0.262	0.233	0.206	0.185	0.167	0.150
8	**0.252**	**0.208**	0.184	0.161	0.139	0.124
16	0.318	0.225	**0.180**	**0.153**	**0.130**	**0.109**
32	0.436	0.300	0.214	0.165	0.137	0.114
64	0.563	0.424	0.295	0.209	0.157	0.130
128	0.663	0.548	0.413	0.288	0.201	0.151

Table IV. Values of *Dist2* metric obtained for three-objective problems, 200 elements, sequential implementation, different numbers of moves

Size of generating sample	Number of moves					
	128	256	512	1024	2048	4096
1 (Serafini's algorithm)	0.572	0.488	0.438	0.373	0.354	0.348
2	0.713	0.673	0.681	0.671	0.646	0.587
4	0.506	0.485	0.440	0.409	0.386	0.343
8	**0.458**	**0.397**	0.397	0.330	0.275	0.240
16	0.508	0.430	**0.365**	**0.326**	**0.262**	**0.219**
32	0.579	0.485	0.425	0.357	0.293	0.251
64	0.667	0.566	0.518	0.445	0.361	0.301
128	0.740	0.649	0.569	0.499	0.437	0.361

significantly. In the tables the best two results for each number of moves are underlined. The best result is additionally written in bold. Please note that the optimal size of the generating sample slowly increases with increasing number of moves. A general conclusion may be drawn from the tables that the larger the number of moves, the bigger is the gain obtained by using a sample of generating solutions in spite of a single solution. For a wide range of number of moves the sample composed of eight or 16 elements gives relatively good results. According to our experience with the MOK and other MOCO problems, eight to 16 generating solutions are usually a good choice.

Table V contains the result of tests when a parallel implementation was simulated, i.e. the number of moves made by each generating solution was constant. Thus the total number of moves increased with increasing size of the generating sample. If, however, calculations required for each of the generating solutions are performed on different processors, the total calculation time will be approximately constant (an additional burden connected with inter-processor communication is ignored). Precisely 128 moves were made by each solution on each temperature level. Of course in this case the results improve with increasing size of the generating sample. One can observe, however, that the improvement is more significant for small sizes of the sample. For relatively large generating samples only a small improvement may be achieved by increasing the sample size.

Table V. Results obtained for three-objective problems, 200 elements, parallel implementation

Size of generating sample	*Dist1*	*Dist2*	*Dist2/Dist1*
1 (Serafini's algorithm)	0.292	0.572	1.959
2	0.270	0.673	2.493
4	0.206	0.440	2.136
8	0.161	0.330	2.050
16	0.130	0.262	2.015
32	0.114	0.251	2.202
64	0.107	0.250	2.336
128	0.102	0.249	2.441

7. SUMMARY AND CONCLUSIONS

A multiple-objective metaheuristic procedure for combinatorial problems has been presented. The procedure tends to generate a good approximation of the efficient solution set in a relatively short time. The main advantages of the procedure are as follows.

- It finds a representation of the whole set of efficient solutions for relatively large problems.
- It naturally allows for a parallel implementation.

The following directions of further research may be considered:

- adaptive setting of the size of the generating sample;
- the use of concepts from other single-objective metaheuristic procedures, e.g. tabu search.

ACKNOWLEDGEMENT

This research was supported by KBN grant 8 S503 016 016.

APPENDIX: INTUITIVE EXPLANATION OF PSA

Paretoland is a country inhabited by shepherds. During the winter they keep their flocks in the valleys and in the summer they move with their flocks to the mountains. The mountains grow up from the south to the north. The Paretolanders are individualists, so they favour pasturing their flocks far away from other shepherds. They prefer the mountain pastures placed on the southern mountainsides. Please note that such pastures may be considered efficient from the point of view of two objectives: elevation and southern latitude. Thus, when a shepherd sees a possibility of moving to a pasture placed higher and further to the south than his current position, he will move his flock to this pasture. However, when he notices another shepherd close to him, he will try to move away from the other shepherd. When he sees another shepherd at a lower elevation and further to the south, he thinks: 'As I am at a higher position than the other guy, I will try to move up'. When he sees another shepherd at a higher elevation and further to the north, he thinks: 'As I am further to the south than the other guy, I will try to move even further in this direction'. Every year this behaviour results in a relatively uniform distribution of flocks on the southern mountain pastures in Paretoland.

REFERENCES

Assad, A. A., 'Modelling and implementation issues in vehicle routing', in Golden, B. L. and Assad, A. A. (eds), *Vehicle Routing: Methods and Studies*, Amsterdam: North-Holland, 1988, pp. 7–46.

Cerny, V., 'A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm', *J. Opt. Theory Appl.*, **45**, 41–51 (1985).

Charnes, A. and Cooper, W. W., 'Goal programming and multiple objective optimizations, Part 1', *Eur. J. Oper. Res.*, **1**, 39–54 (1977).

Corner, J. L. and Buchanan, J. T., 'Experimental consideration of references in decision making under certainty: further evidence', *Proc. XIth Int. Conf. on Multiple Criteria Decision Making*, University of Coimbra, Portugal, August 1994.

Current, J. and Marsh, M., 'Multiobjective transportation network design and routing problems: taxonomy and annotation', *Eur. J. Oper. Res.*, **65**, 1–15 (1993).

Finkel, R. A. and Bentley, J. L., 'Quad trees, a data structure for retrieval on composite keys', *Acta Info.*, **4**, 1–9 (1974).

Fonseca, C. M. and Fleming, P. J., 'An overview of evolutionary algorithms in multiobjective optimization', *Evolut. Comput.*, **3**, 1–16 (1995).

Fortemps, P., Teghem, J. and Ulungu, B., 'Heuristics for multiobjective combinatorial optimization by simulated annealing', *Proc. XIth Int. Conf. on Multiple Criteria Decision Making*, University of Coimbra, Portugal, August 1994.

Garey, M. and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: Freeman, 1979.

Glover, F., 'Tabu search—Part I', *ORSA J. Comput.*, **1**, 190–206 (1989).

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley, 1988.

Habenicht, W., 'Quad trees, a datastructure for discrete vector optimization problems', *Lect. Notes Econ. Math. Syst.*, **209**, 136–145 (1982).

Keeney, R. L. and Raiffa, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, New York: Wiley, 1976.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., 'Optimization by simulated annealing', *Science*, **220**, 671–680 (1983).

Laarhoven, P. J. M. and Aarts, E. H. L., *Simulated Annealing: Theory and Applications*, Dordrecht: Reidel, 1987.

Nemhauser, D. L. and Wolsey, L. H., *Integer and Combinatorial Optimization*, Chichester: Wiley, 1988.

Pirlot, M., 'General local search heuristics in combinatorial optimization: a tutorial', *Belg. J. Oper. Res., Stat. Comput. Sci.*, **32**, 7–67 (1993).

Rosenthal, R. E., 'Principles of multiobjective optimization', *Decis. Sci.*, **16**, 133–152 (1985).

Roy, B. and Bouyssou, D., *Aide Multicritère à la Décision: Méthodes et Cas*, Paris: Economica, 1993.

Schaffer, J. D., 'Some experiments in machine learning using vector evaluated genetic algorithms', *Doctoral Dissertation*, Vanderbilt University, 1984.

Serafini, P., 'Simulated annealing for multiple objective optimization problems', in Tzeng, G. H., Wang, H. F., Wen, V. P. and Yu, P. L. (eds), *Multiple Criteria Decision Making. Expand and Enrich the Domains of Thinking and Application*, New York: Springer, 1992, pp. 283–292.

Slowiński, R., 'Multiobjective network scheduling with efficient use of renewable and non renewable resources', *Eur. J. Oper. Res.*, **7**, 265–273 (1981).

Słowiński, R., 'Multiobjective project scheduling under multiple-category resource constraint', in Słowiński, R. and Węglarz, J. (eds), *Advances in Project Scheduling*, Amsterdam: Elsevier, 1989, pp. 151–167.

Srinivas, N. and Deb, K., 'Multiobjective optimization using nondominated sorting in genetic algorithms', *Evolut. Comput.*, **2**, 221–248 (1994).

Steuer, R. E., *Multiple Criteria Optimization—Theory, Computation and Application*, New York: Wiley, 1986.

Ulungu, E. L. and Teghem, J., 'The multi-objective shortest path problem: a survey', in Černý, M., Gluckaufová, D. and Loula, D. (eds), *Proc. Int. Workshop 'Multicriteria Decision Making Methods–Algorithms–Applications'*, Liblice, 1991, pp. 176–188.

Ulungu, E. L. and Teghem, J., 'Multi-objective combinatorial optimization problems: a survey', *J. Multi-Crit. Decis. Anal.*, **3**, 83–101 (1994).

Ulungu, E. L. and Teghem, J., 'The two phases method: an efficient procedure to solve bi-objective combinatorial optimization problems', *Found. Comput. Decis. Sci.*, **20**, 149–165 (1995).

White, D. J., 'A bibliography on the applications of mathematical programming multiple-objective methods', *J. Oper. Res. Soc.*, **41**, 669–691 (1990).

Wierzbicki, A. P., 'On the completeness and constructiveness of parametric characterization to vector optimization problems', *OR Spectrum*, **8**, 73–87 (1986).