Control and Cybernetics

vol. 29 (2000) No. 3

Tabu search for multiobjective combinatorial optimization: TAMOCO

by

Michael Pilegaard Hansen

Pilegaard Planning
Olof Palmes Gade 3, 5 tv., 2100 Copenhagen, Denmark
E-mail: mph@imm.dtu.dk

Abstract: This paper presents the multiobjective tabu search method, TAMOCO. Being an adaptation of the well-known tabu search, it can be used to generate approximations to the non-dominated solutions of multiobjective combinatorial optimization problems. TAMOCO works with a set of current solutions which, through manipulation of weights, are optimized towards the non-dominated frontier while at the same time seek to disperse over the frontier. The general procedure and some extensions to it are presented, as well as suggestions of usage in interactive procedures. A computational experiment is also presented.

Keywords: multiobjective combinatorial optimization (MOCO), tabu search, metaheuristics.

Introduction

The general multiobjective combinatorial optimization (MOCO) problem can be stated as follows:

"maximize" $f(x)$
subject to $x \in S$

where S is a discrete, finite set of feasible solutions to the problem (the decision space) and f is the n-dimensional objective function to be "maximized", $f(x) = \{f^1(x), f^2(x), \ldots, f^n(x)\}$, $f : S \to \Re^n$.

The term maximization is traditionally written within quotation marks since there generally does not exist any single solution, which can simultaneously provide the optimal value on all objectives. Instead, one may seek to generate a set of non-dominated solutions, where each solution is not worse than the other

optimal set of non-dominated solutions to the MOCO problem is found, when all other solutions in S are dominated by a solution in the set.

While many single objective combinatorial optimization problems are impossible to solve to optimality in polynomial time, even more MOCO problems will be so. For instance, simple single objective problems like the shortest path problem and the assignment problem are NP-complete in a corresponding multiobjective formulation (see Serafini, 1987, and also Garey and Johnson, 1979), even with only two objectives.

This suggests that, in general, we will have to settle for approximations to the non-dominated set. Methods for constructing such approximations have been developed for a selection of particular problems (see, e.g., a survey by Ulungu and Teghem, 1994) using a variety of techniques. Adaptations of general applicable heuristics, or metaheuristics, to multiple objectives have been suggested within the scope of simulated annealing (Serafini, 1992, Ulungu, Teghem and Fortemps, 1995, and Czyżak and Jaszkiewicz, 1996, 1997) and of genetic algorithms (Schaffer, 1985, Horn and Nafpliotis, 1993, Fonseca and Fleming, 1993 and 1995, Horn, Nafpliotis and Goldberg, 1994, and Srinivas and Deb, 1995). Also tabu search (Gandibleux, Mezdaoui and Fréville, 1997) and hybrid methods (Ben Abdelaziz, Chaouachi and Krichen, 1999) have been suggested. In the opinion of Ulungu and Teghem (1994) "the adaptation of these metaheuristics to a multiobjective environment is certainly one of the more promising research directions". See also Borges (1998) and Hansen (1998) for more on metaheuristics for multiobjective combinatorial optimization.

This paper is organized as follows. After introducing some basic notations and general applicable methods in Section 1, we present the basic multiobjective tabu search procedure TAMOCO in Section 2, followed by some useful extensions in Section 3. Section 4 gives a few general comments about using tabu search in the multiobjective context and Section 5 suggests ways of using TAMOCO in some families of interactive methods. Section 6 contains a computational example and the paper closes with some final remarks.

1. Notation and general methods

1.1. Non-dominance

For the sake of shorter notation, we will often refer to an objective function vector as a *point* z. The image of solution x_1 is $z_1 = f(x_1)$, and could also be represented as $\{z_1^1, z_1^2, \ldots z_1^n\} = \{f^1(x_1), f^2(x_1), \ldots f^n(x_1)\}$.[1] Throughout this paper, objectives' indices are written in superscript. Let us also define dominance and superiority:

[1]We not only assume that on each objective "more is better" but also that a solution can be evaluated by a combination of marginal attributes, although this viewpoint has been disputed for more than two millennia dating back to Aristotle who wrote in *Metaphysica* that

The point z_1 *dominates* the point z_2 if and only if $z_1 \geq z_2$ and $z_1 \neq z_2$ (i.e. if $z_1^k \geq z_2^k$ for all the objectives k and $z_1^k > z_2^k$ for at least one objective k).[2] If a point is not dominated by any other point, it is called a non-dominated point.

Solution x_1 is *superior* to solution x_2 if the point $f(x_1)$ dominates the point $f(x_2)$. Solution x_1 is *inferior* to solution x_2 if the point $f(x_2)$ dominates the point $f(x_1)$. If the point $f(x_1)$ is non-dominated, then x_1 is *non-inferior*.

The set of all non-inferior solutions is sometimes referred to as the Pareto optimal set or the efficient set. The set of all non-dominated points is referred to as the non-dominated set.

Also, we use the λ-vector space Λ, defined as: $\Lambda = \{\lambda \in \Re^n \mid \lambda^k \in [0;1] \wedge \sum_{i=1}^n \lambda^i = 1\}$.

A point z is then *supported* if and only if there exists a weight-vector $\lambda \in \Lambda$ so that the scalar value $z^T\lambda$ is the maximum value of $f(x)^T\lambda$ over all x in S. While supported points always are non-dominated, the set of non-dominated points consists of both supported and non-supported points.

1.2. Scaling, metric and measure

Range equalization factors are used to equalize the ranges of the objectives, see Steuer (1986, section 8.4.2), and can be calculated as:

$$\pi^k = \frac{1}{Range^k}\left[\sum_{i=1}^{n} \frac{1}{Range^i}\right]$$

where $Range^k$ is the range width of objective k, given a set of points. The last factor, which can be omitted, is a normalization factor ensuring that $\pi \in \Lambda$.

The λ-weighted Tchebycheff metric defines a distance between two points z_0 and z_1 as:

$$||z_0 - z_1||_\lambda = \max_{k=1..n} \{\lambda^k |z_0^k - z_1^k|\}$$

and an augmented λ-weighted Tchebycheff metric can be defined as:

$$|||z_0 - z_1|||^\lambda = ||z_0 - z_1||^\lambda + \rho \sum_{k=1}^{n} \lambda^k |z_0^k - z_1^k|$$

where ρ is a (often very small) positive augmentation factor.

When measuring the distance from a given point z_0 to a set of points $Z^* = (z_1, z_2, \ldots, z_m)$ we can use the minimum value of an achievement scalarizing function (Wierzbicki, 1986) over the points of Z^*:

$$s^*(z, Z^*, \lambda) = \min_{i=1..m} \{ \max_{k=1..n} \{\lambda^k (z_0^k - z_i^k)\}\}$$

[2]Strictly speaking, this is the definition of a *weak* domination. Solution x *strongly* dominates solution y if and only if $z_1 > z_2$ (i.e. if $z_1^k > z_2^k$ for all objectives k), see Steuer (1986).

This measure can be seen as the λ-weighted Tchebycheff distance from the point z_0 to its projection on the set Z^*, and is intended for usage when z_0 is non-dominated with respect to all points in Z^*.

1.3. Quad tree

"Quad tree" is an important data structure for working with multi-dimensional functions. Originated by Finkel and Bentley (1974), it has found many applications in e.g. databases and image processing. It allows for indexing in multiple dimensions and the general idea is to store all points as nodes in a tree structure, so that the point of each node (as well as all the points under the node) holds a certain relation to the point of the parent node on all dimensions (combinations of "smaller than" and "not smaller than" on each dimension). Although the name *Quad* tree stems from the first formulation in two dimensions, the principle is easily generalized.

Quad tree was first applied to multiobjective optimization by Habenicht (1982), but as a domination-free quad tree (i.e. a quad tree of points where no point dominates any other point) and with computationally very efficient methods to determine whether a given new point is dominated by points in the tree and to retrieve points in the tree, which it dominates.

Sun and Steuer (1996) have presented a method to find the point in a quad tree that is closest to a given new point according to a λ-weighted Tchebycheff metric. Since such a method exists, we will in the following not be intimidated from using distances based on a λ-weighted Tchebycheff metric or using achievement scalarization functions based on this metric.

2. The basic TAMOCO procedure

Tabu search is an optimization procedure which repeatedly moves from a current solution to the best in a list of neighboring solutions while seeking to avoid being trapped in local optima by keeping a tabu-list of forbidden moves. Tabu search in its current form was first suggested by Glover (1989, 1990) and has since then received much success in combinatorial optimization for its fast and aggressive search and very often excellent solution quality. Besides the fundamental tabu-list, many generally applicable extensions have been suggested in the literature. Excellent introductory material can be found in Glover and Laguna (1995 and 1997) and Glover, Taillard and de Werra (1993).

The multiobjective tabu search procedure, TAMOCO, works with a set of current solutions, which simultaneously are optimized towards the non-dominated frontier. The points of the current solutions are sought to cover the whole frontier. Repeatedly for each solution, an optimization direction is made so that the point tends to move away from the other points while moving towards the non-dominated frontier. The solutions take turns in applying one move

We will in the following comment on Fig. 1, which in a Pascal-like fashion outlines the basic TAMOCO procedure.

```
    Procedure basic TAMOCO (maximization of all n objectives):
1.  for each solution x_i in X do set x_i to a random feasible solution
      and set TL_i = {}
2.  set ND = ∅ and set count = 1 and set π^k = 1/n for all objectives k
3.  repeat
4.    for each solution x_i in X do
5.      set λ = 0
6.      for each solution x_j in X where f(x_j) is not dominated by f(x_i)
            and f(x_i) ≠ f(x_j) do
7.        set w = g(d(f(x_i), f(x_j), π))
8         for all objectives k where f^k(x_i) > f^k(x_j) do set λ^k = λ^k + π^k w
9.      end
10.     if λ = 0 then set λ to a randomly chosen vector from Λ
11.     normalize(λ)
12.     find the solution y_i which maximizes λ · f(x_i) where y_i ∈ N(x_i)
          and A(x_i, y_i) ∉ TL_i
13.     if TL_i is full then remove oldest element from TL_i
14.     add A(y_i, x_i) to TL_i as the newest element
15.     set x_i = y_i
16.     if f(y_i) is not dominated by any point in ND then implement the
          point f(y_i) into ND and update π
17.     if DRIFT-criterion is reached then set one randomly selected
          solution from X equal to another randomly selected solution
          from X
18.     set count = count + 1
19.   end
20. until STOP-criterion is met
```

Figure 1: The basic TAMOCO procedure

In line 1, each current solution is set to a random feasible starting solution and its tabu-list (TL) is emptied. In line 2, the current set of non-dominated points (ND) is emptied, an iteration counter is reset and the range equalization factors (π) are set to a unit vector. We then begin the loop which continuously lets each current solution make one move to a neighbor solution until some STOP-criterion is met.

In lines 5-11, the weight vector (λ) for the point is determined. This vector belongs to Λ and will thus ensure optimization towards the non-dominated frontier. We want to set the weights so that the point moves away from the other points, ideally having the points uniformly spread over the frontier. Therefore each element in the weight vector is set according to the proximity of other points for that objective. However, we only compare a point with those points

another point is, the more it should influence the weight vector. The closeness is measured by a distance function (d) based on some metric in the objective function space and using the range equalization weights. The influence is given by a decreasing, positive valued proximity function (g) on the distance. In practice, the proximity function $g(d) = 1/d$ has shown to work well, as well as the Manhattan distance norm (used on the objectives scaled by the range equalization factors, i.e. $d(z_i, z_j, \pi) = \sum \pi^k |z_i^k - z_j^k|$ summed over all objectives k) is a general suggestion yielding a slight focus on the resulting trade-off knee.

In lines 12-15, the standard tabu search procedure is used to replace a current solution with the best feasible neighbor solution (generated by the neighborhood function N) which can be reached from the solution without violating the tabu-list. How to establish a tabu-list is, as always, an open question; in general, the tabu element is some sort of attribute (A) on the from- and to-solutions in order to prevent backward moves. The best neighbor is determined by the scalar product between the weight vector and the vector objective function.

In line 16, the new point is inserted into the ND-set if it is non-dominated with respect to this set. The points already in the ND-set, which thereby become dominated, if any, are removed. If we wish, we can also record here the solution itself. The range equalization factors are set according to the ranges of the points in the ND-set and may therefore need to be updated (obviously, they can only be calculated, if we have at least two points defining a positive range in each objective). Using the ranges of the points in the ND-set is a general suggestion in cases when no other knowledge of the ranges is available.

In line 17, we replace one randomly selected solution by another randomly selected solution whenever a DRIFT-criterion is satisfied, which for instance can be after a fixed number of increments on the count-variable. This is done in order to assure a drift in the movement of the points over the non-dominated frontier, thus exploring the whole frontier, and helps in locating non-supported non-dominated points. The next section will show better the ways of ensuring this drift and at the same time dynamically adjust the number of current solutions in X.

Finally, in line 18, the iteration counter is incremented by 1, and we are ready to continue with the next of the current solutions.

3. Extensions to TAMOCO

3.1. Ensuring a drift and adjusting the number of current solutions

What is likely to happen in the basic TAMOCO is that each of the current solutions converges to its own region in the objective space, explores it carefully using the tabu search metaheuristic, but does not deviate much from the center of its region. While this can be advantageously exploited in some interactive methods, it is in general a bad behavior, since we wish to approximate the

to *drift* over this frontier, thereby changing the centers of their regions. This can be done as suggested in the basic TAMOCO, but it can be more desirable to connect it with an adjustment in the number of current solutions.

Setting the number of current solutions in the basic TAMOCO can be difficult. There should be as many as needed for a reasonable covering of the non-dominated frontier. Too many current solutions on the other hand will give an overlap resulting in too high interference among solutions and an uncontrolled oscillation of the points. Besides, it will demand many computations when determining the weights in lines 6–9 just like it will slow down the convergence of the points on their path towards the non-dominated frontier. These considerations depend on the shape of the non-dominated frontier and of the neighborhood function used. Either way, it can be difficult to predict in advance, and we therefore suggest a dynamic scheme for adjusting this size.

The underlying idea is that the more the search domains of the current solutions overlap, the more often will they dominate each other. Therefore we will in line 17 calculate the domination degree of solution x_i as a count of the number of other current solutions, which dominate it.[3,4] When the DRIFT-criterion is met, we calculate the average of the domination degrees (ADD) of the current solutions and upon this decide whether to 1) increase the number of current solutions, 2) decrease the number of current solutions or 3) leave the number of current solutions unchanged.

If we increase the number of current solutions, we do so by duplicating solutions with the lowest domination degree. If we decrease the number of current solutions, we delete solutions with high domination degree. If we want to keep the number of current solutions constant, we both duplicate and delete solutions according to the highest/lowest domination degree. In all cases, the replaced point will move from "more dominated" to "less dominated" areas on the frontier.

The decision as to what to do can for instance be made using a target interval on ADD. In this implementation, we increase when ADD is lower than 0.03 and decrease when ADD is higher than 0.06, and otherwise leave the number of current solutions unchanged. The actual values of such an interval depend on the neighborhood function used.

The number of current solutions that should be replaced is another design parameter. We suggest doing so with a percentage of the number of current solutions, e.g. 2%, but always for at least one solution. Upon changing the number of current solutions, we may want to use a higher percentage in order to have a faster convergence of the number of current solutions to "what the

[3]This is not unlike Fonseca and Fleming (1993) who used the same measure (called the rank) as the fitness function ($\approx$ objective function) for a genetic algorithm. It is unlikely that their approach can be taken much further in our context, since the vast majority of the current solutions will be non-dominated by other current solutions (be of rank 1).

[4]In practice, we may choose to use a moving average over a number of iterations to describe

frontier can bear using the given neighborhood function". This can be done by e.g. increasing the number of current solutions by 10% when increasing, and decreasing the number of current solutions by 5% when decreasing.

The drift-criterion should be designed so that the new set of current solutions has time to settle over the non-dominated frontier and to explore. This depends on the shape of the non-dominated frontier and the steps in the objective function value space, which the neighborhood function provides, and can typically be after a number of iterations proportional to the number of current solutions or a slower increasing function of this.

For computational efficiency, however, we may initially want to over-ride the drift-criterion and control the growth in the number of current solutions so that these have time to spread away from their origin and constitute a "well-spaced" distribution over the non-dominated frontier before we increase. In the case of a 2-dimensional non-dominated frontier (3 objectives), we can visualize this as an imitation of the geometry of the stem of a tree leading out to larger branches, then to smaller branches and farther out to twigs, thus contributing a new off-spring to the trend of nature inspired heuristics.

Also for computationally efficiency, we will usually set a maximum limit on the number of current solutions, especially when dealing with many objectives.

Concluding this we say that the replacement of current solutions ensures a drift of the current points over the non-dominated frontier. When a solution is removed, it leaves a "hole" in the non-dominated frontier to where other solutions will seek. Likewise will a duplicated solution "push" the solution being duplicated and the other solutions that are close to the two.

3.2. Saturation and minimum levels

In some cases it will be possible, for one or more objectives, to give minimum acceptance levels which must be met in order for a solution to be worth considering. Likewise, it may be possible to give saturation levels on some objectives indicating that whenever these levels are reached, for whatever reason, no further optimization is needed. These levels may be known a priori or obtained after a study of the feasible possibilities.[5]

While such levels can be coded into the objective functions, we also have the possibility of implementing them in the basic TAMOCO by adding 3 lines:

9.1 for all objectives k where $f^k(x_i) > \max^k$ do set $\lambda^k = (1-\alpha^k)\lambda^k$
9.2 if $f^{k'}(x_i) < \min^{k'}$ for any objective k' then
9.3 for all objectives k do set $\lambda^k = (1-\beta^k)\lambda^k + \beta^k\pi^k \text{Max}\{0, (\min^k - f^k(x_i))\}$

[5]For instance, a client may say that: "I prefer to improve our quality but my CEO is dying for a net profit of 15%; less can do, but if it is reached, I am content" or with his knowledge of the problem, be able to state that: "we *can* keep a time-window of 2 hours on delivery; we

where for each objective k, $\max^k$ and $\min^k$ are the saturation level and the minimum level, respectively, and the α^k and β^k factors indicate the degree to which these levels should be pursued (all α^k and β^k values in $[0;1]$).

Normal settings are $\alpha = \beta = \underline{1}$, and the procedure will then work as follows: If one or more of the minimum levels are not reached, the weights are set so that the optimization direction for the point is directly towards the nearest point in the range scaled objective space where all minimum levels are reached. Otherwise, the basic weights are used except that we set the weight to 0 on objectives, for which the saturation level has already been reached, thus indicating that we need not a further optimization on these objectives. Whenever α^k and β^k are lower than 1, some linear combination with the original weights is used.

As mentioned in the example, minimum levels can sometimes be used as a soft way of implementing restrictions, thus relaxing the problem. Also, the minimum and saturation levels need not be parallel to the objective axes but can be general expressions of the objectives. Lines 9.1–9.3 should then be modified correspondingly to work on the contour lines of these expressions.

The idea of saturation and minimum levels is, at this point, not directly aimed for usage in interactive procedures (although they can be a natural part of these). Whether or not one wishes to record non-dominated solutions in the ND-set in line 16 when they do not meet the minimum or saturation levels, must depend on the intention behind each level and the problem being solved.

3.3. STOP-criterion

It has been said that "the wise knows when to stop" and indeed, for some problems, deciding when to stop can be difficult, because the TAMOCO procedure continues to improve the ND-set. Naturally, one may stop the procedure after a given number of iterations, after a given amount of CPU-time, after a given number of iterations without improving the ND-set, and so on. But before suggesting a more advanced stopping criterion, we first take a look at what a good approximation to the non-dominated set is.

In single-objective optimization, we may compare the best-found solution with the optimal solution. If the optimal solution is not known, we can compare with a bound on the optimal solution or with the best solution found so far by anyone. When measuring the quality of an approximation to the non-dominated set of a MOCO problem, a reasonable approach seems to be using a reference set. The reference set may consist either of the optimal non-dominated set (which, as explained in the introduction, is normally both difficult to find and very large) or an approximation to the optimal non-dominated set. The reference set may in the latter case contain a subset of the optimal non-dominated set (perhaps settling for some of the supported points) or be a pool of the non-dominated points obtained after many runs with a large number of iterations.

Having a reference set, we could then count the number of points, which

two reasons. First, the ND-set may not represent the whole reference set, so, even if a relatively large number of points are located, this may not imply that the ND-set is a good approximation, and vice versa. Secondly, the number of located points will in most difficult or large problems be extremely low.

A better measure is then based on an achievement scalarizing function. The average value over each point in the reference set, r_j, of the function $s^*(r_j$, ND-set, $\lambda)$ seems to be a good way of measuring the ND-set, and is also used by Czyżak and Jaszkiewicz (1997). The λ-vector used will normally be composed of the range equalization factors as based on the reference set. The points of the reference set are here seen as representatives for a final selection. Thus, the points of the reference set should be distributed accordingly.[6] For more on measuring the quality of approximations to the non-dominated set, please refer to Hansen and Jaszkiewicz (1998).

Measuring with respect to a reference set is useful in cases of comparisons of methods, setting of parameters, etc., but it does not give us a stopping criterion. However, preliminary results indicate, as we should expect, that there is in practice an extremely high correlation between the above-mentioned measure of the ND-set with respect to a reference set and the average value to the ND-set of the new neighbors generated. The value of a new point x_i is then measured in line 16 by $s^*(x_i$, ND-set, $\pi)$ before x_i is implemented. Therefore, the average of these values can be used as basis for when the procedure should be stopped.[7]

3.4. Indifference thresholds

In line 16 of the basic TAMOCO, all new non-dominated points are recorded in the ND-set. In many combinatorial problems, however, there exist so many non-dominated points that this, for storage reason, is practically impossible to do.

A general applicable but by no means perfect technique for handling this is only to insert new non-dominated points in the ND-set in the following way: If the new point dominates one or more points in the ND-set, the point is inserted (with removal of other points securing non-increasing the current size of the set). Else we only insert the point if the minimum difference with the other points in the ND-set exceeds a certain limit.

This difference can here be a distance measure based on a level of indifference or can be the value of adding the new point to the ND-set, measured by an achievement scalarization function as described for the STOP-criterion.[8]

Either way, the threshold limits can be set dynamically or by the users.

[6] When used for comparison purposes, the reference set should contain points well distributed over the non-dominated frontier; perhaps through the usage of a group of well-dispersed weight vectors (Steuer, 1986, Section 11.9).

[7] In practice, one may use a moving average of the values over a larger number of iterations.

[8] Instead of basing the achievement scalarization function on a weighted Tchebycheff metric, it can be advantageous to base it on an augmented weighted Tchebycheff metric. The intimidating practical problem of doing this, is that it is computationally more difficult to

Heuristic bounds for the maximum size of the resulting ND-set can in some cases be formulated (for inspiration, see e.g. Fonseca and Fleming, 1994, on a similar problem). Non-dominated points which in this way are not inserted in the ND-set can, for instance, whenever they fulfill a less restrictive difference criterion, be put on a list for later usage. When not inserted in the ND-set, they should still contribute with their values in respect to the STOP-criterion.

3.5. Other general extensions

A possibly efficient improvement of the basic TAMOCO is to repeat lines 12–16 a certain number of times so as to make more moves for each calculation of weights. This is not simply an extension of the neighborhood because we can choose to let the number of repetitions depend on the performance of the neighborhood moves, as discussed for the STOP-criterion. For instance, we may choose to perform A iterations of lines 12–16, and add another B iterations whenever a solution which improves the ND-set (perhaps ignoring indifference thresholds) for a certain achievement value is found. One should be aware, however, that such repeated moves imitate a different neighborhood function, and therefore can lead to a solution which is further away in the objective space, thus giving a higher domination degree (a higher fluctuation of the current solutions) as discussed in the beginning of this section.

Also, depending on the neighborhood function used, it may be desirable to moderate the fluctuation of the points, for instance by using a moving average on the weight vector for each solution. Weights for a solution can be set as $\tau\lambda_{\text{new}} + (1-\tau)\lambda_{\text{old}}$, where τ is a constant in $(0;1]$, λ_{new} being the weight vector as set in lines 5–11, and λ_{old} — the weight vector used previously for the same solution.

Finally, in line 12 we may prefer to use different ways of evaluating the neighbors as found by the single objective tabu search. In theory, the basic version of the procedure may not be good at locating non-supported non-dominated points. This theoretical shortcoming will in many cases not present a problem because also non-supported points are inserted into the ND-set, but the procedure may be more correct when using an (augmented) Tchebycheff metric.

4. Usage of general tabu search techniques

Basically, all tabu search techniques can be used in the TAMOCO procedure. Especially, techniques implementing strategic oscillation and intermediate and long term memory should be considered. However, the natural interaction between the current solutions enforces some tactical oscillation by continuously modifying the weights in the aggregated objective function for each solution. Also the effect of the replacement of points can be regarded as a part of the

Since tabu-lists are kept in order to escape local optima and avoid cycling, these special elements of TAMOCO may give the possibility for shorter tabu-lists, thus giving a more effective search. In particular, the risk of cycling can be reduced, even with relatively short tabu-lists, if the length of the tabu-lists differs between points.

The aspiration criterion of tabu search may also be used. For instance, it can make sense to check more of the neighbors generated in line 12 to see if they can contribute to the ND-set, even if they are the results of tabu moves, instead of only checking the best, non-tabu neighbor. Likewise, neighbors resulting from tabu moves can in some cases be accepted as best neighbors.

As for normal tabu search, in cases where we have a badly connecting neighborhood function or when the neighborhood function induces wide valleys in the objective space, we may need once in a while to sample new solutions in order to be able to search the whole feasible set. This can for instance be done by creating new, randomly generated solutions instead of duplicating existing solutions. But it can (for this particular reason and as in the single objective tabu search) be more advantageous, in a systematic or probabilistic fashion, to use more than one neighborhood function so that these, in combination, connect the whole feasible set. Besides, the problem of wide valleys may present less difficulty in TAMOCO due to the continuous variation of the weights.

With a neighborhood function that contains many neighbors for each solution, it can be more efficient to make moves based on a (probabilistic or systematic) sampling of the neighborhood, or in other ways reduce the neighborhood size. While this can be relevant in single objective tabu search, it may be even more so with multiple objectives because we have an n-dimensional objective frontier to discover and can not remain too long at each locality. With neighborhood functions well suited for tabu search, however, we may be able to locate the best neighbor without explicitly having to generate all the neighbors.

More fundamental is the constant designation of the function leading from one solution to the other ones as a neighborhood function. In single-objective tabu search, it is merely a mapping that assigns a candidate list to each solution. This designation, however linguistic, is done in order to underline the importance of the fact that the neighborhood function provides solutions with somewhat similar points in the objective space. For a given neighborhood function, this ability may depend on the problem size.[9]

Another possibility can be to impose a limit (step-size) to the changes in objective function values, that can result from one move. The limit can be more or less strictly enforced (e.g. by forbidding moves that exceed the limit versus the more soft reduction in the λ-weighted objective function when-

[9]For instance, for the traveling salesman problem, a neighborhood function can be to remove one city from the tour and then insert it at another place in the tour. This move will lead to relatively smaller changes in the objective function values when there are many other

ever the step-size is exceeded) and be dynamically set according to the distance to the nearest current solution in X (where again, proximity is measured in the objective space) used in combination with the weight vector. Usage of such limits in combination with the above mentioned aspiration criteria is obvious.

5. Usage in interactive procedures

It is often argued that interactive procedures, where the solution generation phase alternates with the solution evaluation phase, is the future of multiobjective programming. Moreover, it is argued that the generation phase should consist of a partial generation of the efficient set by e.g. Rosenthal (1985), who considers this approach "to be among the most promising approaches to multiobjective optimization".[10]

One problem in a full generation of the non-dominated set with non-iterative solution generation followed by solution evaluation is that the number of non-dominated solutions normally is large and grows exponentially with the number of objectives. But a more fundamental problem is that the size of the minimal sample of that set which, in order to be within a certain desired distance (for instance measured by an achievement scalarization function) from a given non-dominated point (that could have been the final choice, had it been in the sample), is likewise large and can grow exponentially with the number of objectives or as the desired distance decreases.

For guiding the search, the interactive procedures often use a reference point (e.g. the discrete light beam search of Jaszkiewicz and Slowinski, 1994), a reference weighting vector (e.g. as the search principle of the discrete Pareto race of Korhonen, 1988) or a reduced weighting vector space (e.g. by using intervals on each weight element as in the Tchebycheff method of Steuer and Choo, 1983). These reference directions can be included by combining them with the weight vector as made by TAMOCO itself.

If we have a reference weight vector, r, we can do the following:

11.1 for all objectives k do set $d^k = \pi^k r^k$
11.2 normalize(d)
11.3 for all objectives k do set $\lambda^k = \gamma d^k + (1-\gamma)\lambda^k$

If we have reference weight vectors describing intervals, r and R (containing as elements the minimum and maximum weight levels, respectively), we can do the following:

[10]This is also a critique on e.g. the traditional Goal Programming (GP) approach, raised by for instance Zeleny (1981) and Rosenthal (1985). Ironically, GP has been one of the most used techniques in practical applications (Romero, 1991), also in combinatorial optimization (Ulungu and Teghem, 1994). This author has applied an GP-alike approach to a special and very large practical assignment problem containing no less than 15 objectives (Hansen and Vidal, 1995). In any case, GP in combinatorial optimization can be approached with

```
11.1 for all objectives k do
       if λ^k < π^k r^k then set d^k = π^k r^k
       else if λ^k > π^k R^k then set d^k = π^k R^k
       else set d^k = λ^k
11.2 normalize(d)
11.3 for all objectives k do set λ^k = γ d^k + (1 − γ) λ^k
```

In both cases above, the multiplication by the range equalization factors is only relevant, if the weight vectors do not already contain this scalarization.

With a reference point, p, we first make the direction-vector from $f(x_i)$ to p:

```
11.1  for all objectives k do set d^k = Max{0; π^k (p^k − f^k(x_i))}
11.2  normalize(d) /* notice: if d = 0 then the reference point is equal
to or dominated by x_i */
11.3  for all objectives k do set λ^k = γ d^k + (1 − γ) λ^k
```

In all cases, the γ-parameter $[0;1]$ defines the intensification of the search in the reference direction(-s) (or towards the reference point) on behalf of the diversification of the current solutions of X and therefore in the resulting approximation. The γ-parameter can be a constant or vary; it may for instance make sense to start with a low value and then gradually increase it, as the user gains insight into the necessary tradeoffs in the problem and articulates his preferences. As the insight into the problem is gained, it can also be beneficial to alter the saturation and minimum levels, as described earlier.

This flexibility suggests the usage of TAMOCO in many of the interactive procedures based on a partial generation of the non-dominated set and also in combinations of these methods as for instance in the combined procedure by Steuer, Silverman and Whisman (1993) or even in the unified interactive program of Gardiner and Steuer (1994).

In interactive procedures, we will still have to select some of the points from the ND-set to present to the user (each interactive procedure has its own way of doing this), but now the generation of the alternatives has been intensified in the reference direction(-s) (or towards the reference point), so the ND-set will contain better solutions here.

A special usage of TAMOCO can be the algorithm-led usage of obtaining a set of well-dispersed solutions (either covering the entire non-dominated frontier, covering an area in the frontier by using minimum and aspiration levels, or intensified towards a reference point/in a reference direction(-s), or combinations of all these elements). In this case, we fix the number of current solutions, disregard the drift component and may again wish to use other measurements of distance in line 7 as well as different ways of evaluating neighbors given by the single objective tabu search in line 12. The resulting ND-set needs to be filtered

6. Computational experiment

An experiment has been done using the model and the data of Czyżak and Jaszkiewicz (1996). The problem is to select a set of locations from a given, larger set of 50 potential candidate sites in order to establish a chain of petrol stations under the constraint of a maximum available capital budget. Three objectives are formulated: expected short term profit, expected long term profit and expert evaluation of negative environmental impact. The problem is modeled as a multiobjective capital budgeting problem with additive cost, profits and impact for the selected locations, so the model is equal to a multiobjective knapsack problem and therefore belongs to the class of NP-complete problems. Czyżak and Jaszkiewicz generate an approximation to the whole nondominated set containing 1017 points, using their Pareto Simulated Annealing (PSA) method.

In some knapsack problems, relatively good solutions can be found using a yield per cost ratio. Therefore, we will use here the same neighborhood function as Czyżak and Jaszkiewicz did, namely from a current solution randomly remove a selected location until the most expensive, non-selected location could be accepted, and then randomly insert non-selected locations until there is no room for any other non-selected location. While this neighborhood function is not well suited for tabu search due to randomness and the lack of explicitly exploiting e.g. a yield per cost ratio, it does allow us the chance of underlining certain aspects of TAMOCO and also to compare the results with the PSA method.

In order to find a "best" neighbor, we simply pick the best neighbor generated among a sample of neighbors using the neighborhood function above. This sample size thus determines a steepness of the tabu search. Tabu elements are the locations which were removed in the best solution by the neighborhood function, and these locations are not to be re-inserted in the solution by the neighborhood function until they are no longer in the tabu-list. It should be noted that, in this particular problem, the neighborhood function will normally only remove 1 or 2 stations and will likewise normally only insert 1 or 2 stations.

The sample size takes on the values 1, 2, 5, 10, 20, 50 and 100, and the tabu list length the values 0, 2, 4, 6, 8, 10, a tabu list length of 0 giving an ascent-like procedure with different degrees of steepness. For each of the 42 resulting configurations, TAMOCO generates a fixed number of two million neighbors in each experiment. The configurations with high steepness will therefore suffer from fewer best moves and, as a partial compensation, we examine each generated neighbor solution for possible implementation into the ND-set. The solution generation and drift-scheme is done according to Section 2.1 and with the values suggested there. The ADD is computed using a moving average (where the previous value of the ADD counts half) and the DRIFT-criterion is set after 20 iterations for each of the current solutions. Each configuration is solved 3 times with different start-seeds to the random generator. With the resulting

with the one approximation of the PSA procedure, a reference set is generated as the best of all 127 approximations. Please note that this reference set is not the optimal set of non-dominated solutions and that it is used exclusively to measure the relative quality among the 127 sets.

Table 1 shows the results of the 126 experiments as measured with the s^* function of Section 1.2 when compared with the reference set and using the range equalization factors of the reference set for scalarization. The 126 values have been normalized by dividing each value by the value obtained in the best experiment and the average is then computed over the 3 repetitions. Thus, a value of e.g. 1.4 means that as an average over the 3 repetitions, the reference points were in average 40% further from the approximation (measured by the s^* function) than they were from the best approximation. The reference set contained a total of 2012 non-dominated points. We will briefly comment the results.

Sample size	Length of tabu list					
	0	2	4	6	8	10
1	254.3	239.1	231.0	255.8	243.6	253.9
2	82.6	78.2	81.0	88.7	101.2	112.9
5	21.7	11.9	11.6	14.2	18.4	25.5
10	16.8	3.2	2.8	3.2	3.9	5.0
20	16.7	2.3	1.6	1.3	1.2	1.5
50	14.4	2.7	2.0	1.4	1.3	1.2
100	18.8	3.4	2.7	1.8	1.8	2.0

Table 1. s^* values of the 42 configurations (normalized and averaged over 3 repetitions)

Having a tabu list greatly improves the results. This finding is obviously essential to the proposed multiobjective tabu search method because it indicates how a tabu list indeed helps overcome local minima, even when the search direction is constantly altered.

The absence of a tabu-list does not give very good results. However, the performance here becomes better using a higher steepness. In general, this can imply that random ascent methods in a structured approach improves with the steepness of the ascent (until a level, where they too frequently get trapped in local optima).

The approximation generated by the PSA method had a normalized s^* function value of 155.7, which is clearly inferior to the TAMOCO method. However, we discourage from generalizing based on this case alone. The approximation generated by PSA did not cover the whole non-dominated frontier very well. Large parts of it were left virtually undiscovered, which may be assigned to some error. More generally, it must the stressed, that each problem (including the defining data) is unique and what may prove effective in one instance, may

7. Final remarks

The motivation of this work is that many real world decision problems are combinatorial in their nature and contain a multitude of objectives that can not always be meaningfully modeled in a single objective function. While many methods exist for helping a user to select a solution among a given set of alternatives, few general procedures exist for the generation of these alternatives, especially when dealing with combinatorial problems.

Tabu search has with its aggressive search in many single-objective combinatorial optimization problems shown to give good quality solutions, and doing so very fast. This is an appreciated behavior when optimizing over a multidimensional frontier and tabu search may therefore posses a high potential in multiobjective optimization.

This paper has presented a general adaptation of tabu search for generating an approximation to the non-dominated set for multiobjective optimization problems, primarily aimed at solving problems of combinatorial nature. The basic TAMOCO procedure is here supplied with extensions, which can be seen as examples. Most important, perhaps, is the interactive basis on which the procedure could be used, directing and/or restricting the search to certain areas of the non-dominated frontier. More natural, however, would be the usage as a generator of efficient solutions to established interactive procedures.

The method presented in this paper could benefit from further research. As an example, other ways of setting the weights can be explored. While TAMOCO in preliminary tests has shown to be rather robust with respect to the settings, this will need to be formally investigated. Some settings concern the neighborhood function used on the problem or the shape of the non-dominated frontier and may be adjusted in an adaptive fashion as outlined in the extensions. Finally, more comparisons need to be made with competing methods on other, and more difficult, problems.

References

BEN ABDELAZIZ F., CHAOUACHI J. and KRICHEN S. (1999) A hybrid heuristic for multiobjective knapsack problems. In: Voss, S., Martello, S., Osman, I. and Roucairol, C., eds., *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers, 205-212.

BORGES, P. C. (1998) *Multicriteria planning and optimization — Heuristic approaches.* Ph.D. dissertation, IMM-PHD-1998-50, Institute of Mathematical Modelling, Technical University of Denmark.

CZYŻAK, P. and JASZKIEWICZ, A. (1996) A multiobjective metaheuristic approach to the location of a petrol station by the capital budgeting model.

CZYŻAK, P. and JASZKIEWICZ, A. (1998) Pareto simulated annealing — a metaheuristic technique for multiple objective combinatorial optimization. *Journal of Multiple Criteria Decision Analysis*, **7**, 34-47.

FINKEL, R. A. and BENTLEY, J. L. (1974) Quad trees. A data structure for retrieval on composite keys. *Acta Informatica*, **4**, 1-9.

FONSECA, C. M. and FLEMING, P. J. (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Proceedings of the ICGA '93*. Morgan Kaufmann Publishers, 416-423.

FONSECA, C. M. and FLEMING, P. J. (1995) An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, **3**, 1-16.

GANDIBLEUX, X., MEZDAOUI, N. and FRÉVILLE, A. (1997) A tabu search procedure to solve multiobjective combinatorial optimization problems. In: Caballero, R., Ruiz, F. and Steuer, R. E., eds., *Advances of Multiple Objective and Goal Programming: Proceedings of the Second International Conference on Multi-Objective Programming and Goal Programming (MOPGP '96)*. Springer.

GAREY, M. R. and JOHNSON, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco.

GARDINER, L. R. and STEUER, R. E. (1994) Unified interactive multiple objective programming. *EJOR*, **74**, 391-406.

GLOVER, F. (1989) Tabu Search — Part I. *ORSA Journal on Computing*, **1**, 3, 190-206.

GLOVER, F. (1990) Tabu Search — Part II. *ORSA Journal on Computing*, **2**, 1, 4-32.

GLOVER, F. and LAGUNA, M. (1995) Tabu Search. Chapter 3 in Reeves, C. R., ed., *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill.

GLOVER, F. and LAGUNA, M. (1997) *Tabu Search*. Kluwer.

GLOVER, F., TAILLARD, E. and DE WERRA, D. (1993) A user's guide to tabu search. *Annals of Operations Research*, **41**, 3-28.

HABENICHT, W. (1982) Quad trees. A data structure for discrete vector optimization problems. *Lecture notes in economics and mathematical systems*, **209**, 136-145.

HANSEN, M. P. (1998) *Metaheuristics for multiple objective combinatorial optimization*. Ph.D. dissertation, IMM-PHD-1998-45, Institute of Mathematical Modelling, Technical University of Denmark.

HANSEN, M. P. and JASZKIEWICZ, A. (1998) Evaluating the quality of approximations to the non-dominated set. Submitted for publication.

HANSEN, M. P. and VIDAL, R. V. V. (1995) Planning of high school examinations in Denmark. *European Journal of Operational Research*, **87**, 519-534.

HORN, J. and NAFPLIOTIS, N. (1993) Multiobjective Optimization using the

HORN, J., NAFPLIOTIS, N. and GOLDBERG, D. E. (1994) A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation*, **1**, 82-87.

JASZKIEWICZ, A. and SLOWINSKI, R. (1994) The LBS-discrete interactive procedure for multiple-criteria analysis of decision problems. *Proceedings of the XIth International conference on MCDM*, Coimbra 1-6.08.94.

KORHONEN, P. (1988) A visual reference direction approach to solving discrete multiple criteria problems. *EJOR*, **34**, 152-159.

ROMERO, C. (1991) *Handbook of critical issues in goal programming.* Pergamon Press.

ROSENTHAL, R. E. (1985) Principles of multiobjective optimization. *Decision Sciences*, **16**, 133-152.

SCHAFFER, J. D. (1985) Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the ICGA '85.* Lawrence Erlbaum, 93-100.

SERAFINI, P. (1987) Some considerations about computational complexity for multiobjective combinatorial problems. *Lecture Notes in Economics and Mathematical Systems*, **294**, 222-232.

SERAFINI, P. (1992) Simulated annealing for multi objective optimization problems. *Proceedings of the Xth International conference on MCDM*, Tapei 19-24.07.92, 87-96.

SRINIVAS, N. and DEB, K. (1995) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, **2**, 3, 221-248.

STEUER, R. E. (1986) *Multiple Criteria Optimization: Theory, Computation, and Application.* John Wiley & Sons.

STEUER, R. E., CHOO, E.-U. (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, **26**, 326-344.

STEUER, R. E., SILVERMAN, J., and WHISMAN, A. W. (1993) A combined Tchebycheff/aspiration criterion vector interactive multiobjective programming procedure. *Management Science*, **39**, 1255-1260.

SUN, M. and STEUER, R. E. (1996) InterQuad: An interactive quad tree based procedure for solving the discrete alternative multiple criteria problem. *EJOR.*

ULUNGU, E. L. and TEGHEM, J. (1994) Multi-objective combinatorial optimization problems: a survey. *Journal of Multiple-Criteria Decision Analysis*, **3**, 83-104.

ULUNGU, E. L. and TEGHEM, J. (1995) The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, **20**, 149-165.

ULUNGU, E. L., TEGHEM, J. and FORTEMPS, PH. (1995) Heuristics for multiobjective combinatorial optimization by simulated annealing. *MCDM: theory and applications, Proceedings of the 6th International Conference*

ULUNGU, E. L., TEGHEM, J. and OST, CH. (1998) Efficiency of interactive multi-objective simulated annealing through a case study. *Journal of Operational Research Society*, **49**, 1044-1050.

VISEE, M., TEGHEM, J., PIRLOT, M. and ULUNGU, E. L. (1998) Two-phases method with Branch and Bound procedure to solve bi-objective Knapsack problem. *Journal of Global Optimization*, **12**, 139-155.

ZELENY, M. (1981) The pros and cons of goal programming. *Computers and Operations Research*, **8**, 357-359.