INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

An Evolutionary Metaheuristic for Approximating Preference-Nondominated Solutions

Murat Köksalan, Selcen (Pamuk) Phelps,

To cite this article:
Murat Köksalan, Selcen (Pamuk) Phelps, (2007) An Evolutionary Metaheuristic for Approximating Preference-Nondominated Solutions. INFORMS Journal on Computing 19(2):291-301. https://doi.org/10.1287/ijoc.1050.0170

INFORMS Journal on Computing
Vol. 19, No. 2, Spring 2007, pp. 291–301
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1902 | 0291

DOI 10.1287/ijoc.1050.0170

An Evolutionary Metaheuristic for Approximating Preference-Nondominated Solutions

Murat Köksalan
Department of Industrial Engineering, Middle East Technical University, 06531 Ankara, Turkey,
koksalan@ie.metu.edu.tr

Selcen (Pamuk) Phelps
Department of Accounting, Business, Economics, and Management Information Systems, Westminster College,
Fulton, Missouri 65251, USA, phelpss@westminster-mo.edu

We propose an evolutionary metaheuristic for approximating the preference-nondominated solutions of a decision maker in multiobjective combinatorial problems. The method starts out with some partial preference information provided by the decision maker, and utilizes an individualized fitness function to converge toward a representative set of solutions favored by the information at hand. The breadth of the set depends on the precision of the partial information available on the decision maker's preferences. The algorithm simultaneously evolves the population of solutions out toward the efficient frontier, focuses the population on those segments of the efficient frontier that will appeal to the decision maker, and disperses it over these segments to have an adequate representation. Simulation runs carried out on randomly generated instances of the multiobjective knapsack problem and the multiobjective spanning-tree problem have found the algorithm to yield highly satisfactory results.

Key words: multiple criteria; combinatorial optimization; evolutionary heuristic
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received September 2001; revised March 2002, October 2004, September 2005; accepted November 2005.

1. Introduction

Since many real problems are multiobjective and have combinatorial components, the last decade has witnessed a growing interest in multiobjective combinatorial optimization (MOCO). Most MOCO methods concentrate on obtaining an approximation of the efficient frontier of the problem. Even if a good approximation can be found, the true goal of the decision maker (DM) is still a single solution, or at most a small subset of solutions for detailed consideration. Concluding the exploration with an approximation of the entire efficient frontier hence necessitates a subsequent search, where the DM will first have to eliminate irrelevant solutions, and then choose from among a selection of relevant ones.

We draw on the proven strength of metaheuristics in addressing combinatorial problems, and on work on partial information in the field of multiple criteria decision making (MCDM), to develop an evolutionary metaheuristic to approximate only the relevant segments of the efficient frontier. The method starts out with some partial information on DM preferences, and evolves a population of solutions toward those regions of the objective space that will appeal to the DM. An individualized fitness function allows each solution to represent itself as favorably as possible in light of the known preferences of the DM. By means of this fitness function, the algorithm converges toward, not a single solution, but a set of solutions favored by the information at hand. The breadth of the set depends on the precision of the partial information available. The DM may then either search the final population of the metaheuristic interactively, or draw a sample of diverse solutions that is small enough for his/her detailed consideration. Simulation runs indicate that the algorithm yields highly satisfactory solutions for randomly generated instances of the multiobjective knapsack problem and the multiobjective spanning-tree problem.

The next section overviews the relevant literature and defines terms. Section 3 highlights the significance of the method being proposed and details the algorithm. Section 4 summarizes computational findings for test problems, and Section 5 presents our conclusions and directions for further research.

2. Background Literature and Terminology

MCDM is characterized by the existence of trade-offs between different objectives, which must be evaluated in terms of the preferences of a given DM. In MOCO, this is further complicated by a large and irregular

search space, causing even the single-objective versions of the underlying problems to be NP-hard. A critical survey of the scope and methods of MCDM is given in Stewart (1992), while surveys of MOCO may be found in Ulungu and Teghem (1994) and Ehrgott and Gandibleux (2002, 2004).

In the presence of multiple criteria, the concept of optimality breaks down, to be replaced by that of efficiency. Let X be the set of feasible solutions and $f_k(x)$, $k = 1, \ldots, p$, be the kth objective function evaluated at solution $x \in X$. We say that $y \in X$ is an *efficient solution* to the MOCO problem

$$\text{“}\underset{x \in X}{\text{Max}}\text{”}\{f_1(x), f_2(x), \ldots, f_p(x)\}$$

if there exists no $x \in X$ such that $f_k(x) \geq f_k(y)$ for all $k = 1, \ldots, p$, with at least one inequality strict. If any such x exists, it is said to *dominate* y. Together, all efficient solutions to the problem define the *efficient frontier*. The *ideal point* of the problem, $f^* = (f_1^*, f_2^*, \ldots, f_p^*)$, dominates all feasible solutions and is found by separately maximizing the objectives $f_k^* = \text{Max}_{x \in X}\{f_k(x)\}$.

If the DM's preference structure is known and can be represented by a utility function $U(x) = u(f_1(x), \ldots, f_p(x))$, then the superiority of solutions will be determined by their appeal to the DM and solution x may be said to be *preferred* to solution y if and only if $U(x) \geq U(y)$.

Given the inherent difficulty of expressing and measuring preferences, there will usually be only *partial* or *incomplete information* available on the function U. If we represent all likely utility functions by the set Ω, then we would be able to say that x *is preferred to* y *under* Ω if and only if $U(x) \geq U(y)\ \forall\, U \in \Omega$.

If no such $x \in X$ exists, then solution y is *preference-nondominated under* Ω. Moreover, if a solution y has the highest utility score for at least one likely utility function, i.e., $\exists U \in \Omega \ni U(y) = \text{Max}_{x \in X}\{U(x)\}$, then y is *potentially optimal*.

An overview of issues related to decision making under partial information on U, and of the early literature on the subject, is in Weber (1987). While occasional studies (see, for instance, Hazen 1986) have considered varying families of utility functions, the widespread tendency has been to assume that U is linear, $U(x) = \sum_{k=1}^{p} w_k f_k(x)$, and that partial information pertains to the vector $w = (w_1, \ldots, w_p)$ of *criteria weights*. Under such a utility function, some authors (e.g., Athanassopoulos and Podinovski 1997) have aimed to identify preference-nondominated or potentially optimal solutions.

If the DM is able to formulate linear inequalities constraining the possible values of w, then these inequalities, in conjunction with a normalizing constraint of the form $\sum_{k=1}^{p} w_k = 1$, form a polytope W of all possible weight vectors. Under these circumstances, solution x is at least as preferred as solution y under W if and only if $\sum_{k=1}^{p} w_k(f_k(x) - f_k(y)) \geq 0\ \forall\, w \in W$.

Most authors studying MOCO problems have characterized or approximated the full efficient frontier, although the cardinality of this set may be exponential in problem size. Of these, Hajela and Lin's (1992) genetic algorithm most resembles our method in that each solution is assigned its own vector of criteria weights; however, the weight vectors in that study are evolved along with the other properties encoded in the solutions' genetic representation. In general, Coello Coello (2000) notes that studies approximating the entire efficient frontier "do not provide any insight into the decision making itself." At the other extreme, *a priori* approaches have assumed that the DM could provide precise information on either goal values or criteria weights for an aggregate utility function, in effect reducing the problem to a single objective. As a more insightful alternative, occasional interactive metaheuristics progressively elicit information from an accessible DM and converge toward the unique solution that represents the DM's most favored compromise of objectives (Teghem et al. 2000, Phelps and Köksalan 2003). The partial-information case, where the DM is not accessible beyond providing some limited preference information *a priori*, has to our knowledge not been addressed explicitly.

The success of multiobjective extensions of metaheuristics such as tabu search (e.g., Gandibleux et al. 1996), simulated annealing (e.g., Czyzak and Jaszkiewicz 1998), and evolutionary algorithms (e.g., Zitzler and Thiele 1999) in approximating the efficient frontiers of MOCO problems makes it likely that these methods may be further extended to work in the partial-information case. Deb (2001) and Coello Coello (1999, 2000) provide overviews of the work done using evolutionary algorithms; of these a number of directed search methods could conceivably be applied to the partial-information problem. In Deb's (2001, pp. 379–382) biased sharing approach, a set of criteria weights supplied by the DM is used to bias the sharing function, so that the spread of the final population will favor criteria with higher weights. The Branke et al. guided domination approach and the Parmee et al. weighted domination approach (Deb 2001, pp. 382–386) also direct the search to particular regions of the efficient frontier, this time by generalizing the traditional definition of domination.

3. An Evolutionary Metaheuristic

The significance of the evolutionary metaheuristic for approximating preference-nondominated solutions (EMAPS) proposed here is that all available preference information is used to guide and restrict

the search effort. Instead of spending computational resources securing the entire efficient frontier, some parts of which are of no interest to the DM, the portions that are of interest are searched in greater depth. The subsequent search for a single solution is hence carried out in a narrower and more relevant search space. The more precise the partial information on the DM's preferences, the narrower will be the space in question. The algorithm is hence a flexible tool, and may be used for all cases between the two extremes of perfect information and no information. If the utility function of the DM is known precisely, then the algorithm will approximate the optimum for that function; in the absence of any preference information, the full efficient frontier will be approximated.

Traditionally, the goals of most multiobjective evolutionary metaheuristics are twofold: (1) to push the population of solutions out towards the efficient frontier, and (2) to disperse it over the frontier to have an adequate representation. Dominance-based fitness functions are the most common tool for guiding the population in direction (1), while direction (2) is usually achieved by some form of fitness sharing or niching. In EMAPS, a third goal is added to the other two, as the algorithm simultaneously strives (3) to focus the population on those segments of the efficient frontier that will appeal to the DM, given our partial information on his/her preferences.

The main difference between EMAPS and the other guided-search evolutionary algorithms mentioned in Section 2 lies in the information that the DM is expected to provide. No precise utility weights or dominance-transforming parameters are presupposed; the preference information is instead elicited through qualitative statements that the DM might feel more comfortable making. This information is then used in a special fitness function, to restrict the search directions taken by the algorithm.

EMAPS evolves a population of solutions with the goal of generating a good approximation of the DM's preference-nondominated solution set (X^{PND}) under a constrained weight set W. Although it is assumed that W is defined by linear inequalities and consistent with a linear utility function, the algorithm does not eliminate convex dominated solutions.

A major factor in the design of the algorithm is use of strategies that encourage the population's diversity and hence good coverage of the entire set X^{PND}. For this purpose, duplicate solutions are not permitted to be simultaneously present in the population. The fitness score of each new offspring is calculated by solving an LP model. This allows each member of the population to be evaluated in terms of the weight vector $w \in W$ that puts it at the greatest advantage, and assigns higher scores to those solutions that approximate the portions of X^{PND} that are underrepresented. Finally, new solutions entering the population tend to replace inferior solutions to which they are similar.

3.1. The Restricted Weight Space

For the purposes of this study, we will assume that the DM provides us with some information that can be transformed into linear inequalities on criteria weights, which we use to constrain the weight space W. Statements of the form:

- $f_k \succ f_m$ (the DM finds objective k to be at least as important as objective m)
- $x^i \succ x^j$ (the DM prefers solution x^i over solution x^j)
- $w_k^{LB} \le w_k \le w_k^{UB}$ for a subset of objectives $K \subseteq \{1, \ldots, p\}$

may thus be used to form the set

$$W = \Bigg\{ w = (w_1, w_2, \ldots, w_p) \,\Big|\, w_k \ge w_m \ \forall f_k \succ f_m,$$
$$\sum_{k=1}^{p} w_k (f_k(x^i) - f_k(x^j)) \ge 0 \ \forall x^i \succ x^j,$$
$$w_k^{LB} \le w_k \le w_k^{UB} \ \forall k \in K, \sum_{k=1}^{p} w_k = 1,$$
$$w_k \ge 0 \ \forall k = 1, \ldots, p \Bigg\}.$$

We assume that W is nonempty, i.e., that the information provided by the DM is consistent, both internally and with our assumptions on the form of the utility function. The resolution of inconsistencies during the specification of W remains a problem of some interest, but is not addressed here, since various resolution methods may conceivably be devised without much difficulty.

3.2. Forward Filtering

At several points in the algorithm, a set of p-dimensional (solution or weight) vectors is filtered to obtain a smaller diverse set by an operation known as *forward filtering*. The idea is to choose those vectors that are approximately evenly spaced and collectively cover all parts of the space covered by the entire set. This is achieved by constructing a subset of elements that are as dispersed as possible with respect to a given metric, and is explained in detail in Steuer (1986, pp. 311–321). In EMAPS, Euclidean distances are the metric to be maximized in constructing the forward-filtered subset.

3.3. Initialization and Weight-Space Sampling

The initial population of EMAPS consists of some solutions generated randomly, and some relatively higher-quality solutions used to seed the population. The seed solutions are produced by repeated application of a base heuristic known to work well on

the single-objective version of the underlying MOCO problem; where the single-objective problems to be solved are formed by linearly aggregating the objective functions using different weight vectors obtained through an even sampling of the restricted weight space W.

In the absence of any special structure for W, it is possible to generate a large set R of weights by incrementing each w_k uniformly from 0 to 1 by some step size $1/r$. The set

$$R = \left\{ w=(w_1, w_2, \ldots, w_p) \,\middle|\, w_k \in \left\{0, \frac{1}{r}, \frac{2}{r}, \ldots, \frac{r-1}{r}, 1\right\} \right.$$
$$\left. \forall k = 1, \ldots, p, \sum_{k=1}^{p} w_k = 1 \right\}$$

formed in this way has cardinality

$$|R| = \binom{r+p-1}{p-1}.$$

The weight vectors belonging to the set $W \cap R$ may then be used for linear aggregation of objectives. If the set $W \cap R$ has fewer elements than the desired number of seed solutions, the process may be repeated for a higher value of r. If the set $W \cap R$ has too many elements, it may be forward filtered to the desired cardinality. It should, however, be noted that not all aggregation weights need lead to distinct solutions, so that the number of seed solutions that are actually obtained may be smaller than the number of weight vectors in the filtered set.

3.4. The Fitness Function

The likelihood that a solution will survive and reproduce during evolution is determined by its fitness score, which, in EMAPS, should reflect the solution's contribution to approximating X^{PND}. One way of evaluating this contribution involves allowing each solution to select a favorable search direction in W, and measuring how superior it is to the other solutions in the population with respect to performances along that direction. Such a fitness function evolves the population in the three directions mentioned earlier: (1) A given solution immediately has higher fitness than any solutions it dominates, so evolution is towards the efficient frontier. (2) Solutions approximating relatively underrepresented portions of X^{PND} have higher fitness than those that are clustered together, since the former outperform a higher proportion of the population under their preferred search direction, while the latter have to vie with each other for fitness. This implicit sharing effect spreads the population over X^{PND}. (3) Solutions approximating X^{PND} have higher fitness than solutions approximating other portions of the efficient frontier that are remote from the search directions indicated by W, so evolution "focuses" on the available preference information.

3.4.1. Relative Strength of Solutions. Given the existing population X' at any point in the algorithm, suppose that we would like to compare a new solution x^{new} with the members of X'. Let us denote by $\phi(w, x)$ the linear combination under weight vector w of the criteria values for solution $x \in X$: $\phi(w, x) = \sum_{k=1}^{p} w_k f_k(x)$. If we can somehow compute a favorable weight vector for x^{new}, $w^{\text{new}} \in W$, then a measure of the strength of x^{new} relative to some $x^i \in X'$ would be $\varepsilon^i = \phi(w^{\text{new}}, x^{\text{new}}) - \phi(w^{\text{new}}, x^i)$, indicating the surplus utility the DM might derive from choosing x^{new} over x^i if s/he indeed has a linear utility function defined by the weights w^{new}. The strength of x^{new} relative to the entire population X' may then be measured by either the average of these differences, $\bar{\varepsilon} = \sum_{x^i \in X'} \varepsilon^i / |X'|$, or the difference between x^{new} and its closest contender along the search direction w^{new}, $\varepsilon = \text{Min}_{x^i \in X'}\{\varepsilon^i\}$.

A negative ε value indicates that x^{new} is outperformed by some $x^i \in X'$ even at its favorable weights, so that it cannot be potentially optimal for linear utility functions. However, it is possible for a convex-dominated solution with $\varepsilon < 0$ to be potentially optimal if the underlying utility function is not linear. By itself, the worst-case measure ε may be a poor indicator of the fitness of x^{new}, as it is possible for an x^{new} with a few close contenders to fall in a fairly underrepresented portion of X^{PND}, and hence outperform many other solutions in X' under its favorable weights. Taken together, the two terms ensure that the fitness function forces the population to maximize not only the weighted criterion values $w_k f_k(x)$, but also the spread of solutions across the relevant portion of the efficient frontier. Thus, in EMAPS, the fitness score $\textit{fitness}(x^{\text{new}}) = \alpha\bar{\varepsilon} + (1-\alpha)\varepsilon$ is a convex combination of ε and $\bar{\varepsilon}$ for some $\alpha \in [0, 1]$. Extensive test runs suggest that α levels of 0.00, 0.25, and 0.50 tend to work well. If computational resources can be spared, a conservative approach would be to make several runs of EMAPS under different values of α and combine the results.

3.4.2. Favorable Weight Computation. A straightforward application of the above idea is to solve the following LP to find the *favorable weights* w^{new} of x^{new} as the weights that maximize the fitness of x^{new}:

$$\text{Max} \quad \frac{\alpha}{|X'|} \sum_{x^i \in X'} \varepsilon^i + (1-\alpha)\varepsilon$$
$$\text{st} \quad \sum_{k=1}^{p} w_k^{\text{new}} (f_k(x^i) - f_k(x^{\text{new}})) + \varepsilon^i = 0 \quad \forall x^i \in X'$$
$$\varepsilon - \varepsilon^i \le 0 \quad \forall x^i \in X'$$
$$w^{\text{new}} \in W,$$
$$\varepsilon \text{ and the } \varepsilon^i \text{ unrestricted in sign.}$$

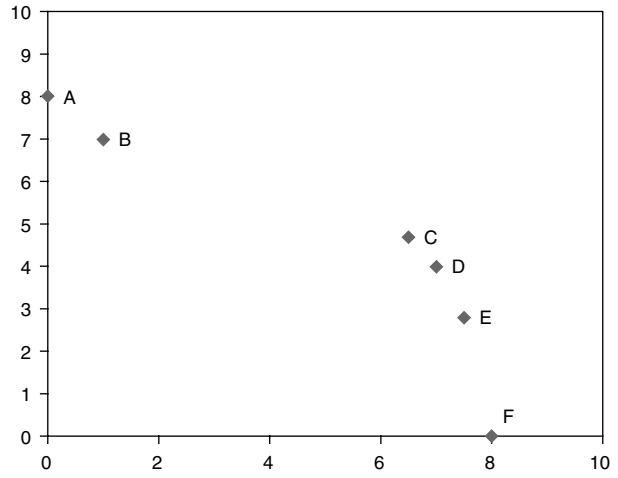

Figure 1 An Example

However, since the number of constraints in the above LP is proportional to the cardinality of the population, and since it has to be solved frequently, the calculation of favorable weights and fitnesses may become too time-consuming for large population sizes. As an alternative, one may store a smaller set X'' of dispersed high-quality solutions and substitute X'' for X' in the LP to be solved for the computation of w^{new}. In our applications of EMAPS, we used as X'' a forward-filtered subset of the seed solutions in the initial population. The smaller LP for each x^{new} can then be solved in less time, resulting in substantial time savings over the course of evolution. The objective function of the smaller LP, however, does not directly yield the fitness of x^{new}. Once the favorable weights w^{new} are found, the score $fitness(x^{new})$ must instead be calculated separately using the entire population X', as explained in the previous section.

To illustrate the weights and the fitness values, consider the two-criteria example in Figure 1. Table 1 shows the favorite weights, the fitness values, and the ranks of the solutions under different α values. All solutions are efficient, but solution B is dominated by convex combinations of some of the other solutions. Though its fitness value is low for small α values, it improves for larger α values. This is an important property, as it is desirable to have such convex dominated solutions to survive in the populations for problems having nonconvex sets of efficient solutions.

3.4.3. Fitness Updates. Since the fitness of solution x^{new} is calculated with respect to the population at the time of entry of x^{new}, fitness values may become outdated as the population evolves. Hence, a solution may retain the high fitness score assigned at the time of its entry, and yet be outperformed by many others that entered the population after it. Conversely, if a certain portion of X^{PND} becomes relatively less represented over time, the fitnesses of the solutions falling in that portion may underestimate their increased contribution to approximating X^{PND}. To prevent such occurrences, the fitness values of all solutions in the population are updated upon every entry into, or exit from, the population. This process is not very time-consuming since, with the use of X'', the favorable weight vectors of solutions do not depend on the members of X' at any given time.

Table 1 Favorite Weights, Fitnesses, and Ranks of the Solutions in the Example

α	Point	Fav. weights	Fitness	Rank
0.00	A	(0.00, 1.00)	1.0000	1
	B	(0.34, 0.66)	−0.3265	6
	C	(0.36, 0.64)	0.2636	3
	D	(0.66, 0.34)	0.0862	5
	E	(0.80, 0.20)	0.1600	4
	F	(1.00, 0.00)	0.5000	2
0.25	A	(0.00, 1.00)	1.8250	1
	B	(0.00, 1.00)	0.0250	6
	C	(0.36, 0.64)	0.4155	4
	D	(0.66, 0.34)	0.4147	5
	E	(0.80, 0.20)	0.6230	3
	F	(1.00, 0.00)	1.2750	2
0.50	A	(0.00, 1.00)	2.6500	1
	B	(0.00, 1.00)	1.0500	4
	C	(0.71, 0.29)	0.6118	6
	D	(0.85, 0.15)	0.8591	5
	E	(1.00, 0.00)	1.2500	3
	F	(1.00, 0.00)	2.0500	2
0.75	A	(0.00, 1.00)	3.4750	1
	B	(0.00, 1.00)	2.0750	4
	C	(0.85, 0.15)	1.0440	6
	D	(1.00, 0.00)	1.5500	5
	E	(1.00, 0.00)	2.1250	3
	F	(1.00, 0.00)	2.8250	2
1.00	A	(0.00, 1.00)	4.3000	1
	B	(0.00, 1.00)	3.1000	3
	C	(1.00, 0.00)	1.8000	6
	D	(1.00, 0.00)	2.4000	5
	E	(1.00, 0.00)	3.0000	4
	F	(1.00, 0.00)	3.6000	2

3.5. Evolution

Evolution involves iteratively choosing members of the population to be recombined to form new solutions, which may then replace relatively inferior members of the existing population. While the genetic representation of solutions and the (crossover, mutation, repair, etc.) operators being employed during evolution are problem-specific, some generic features of the algorithm are detailed below.

3.5.1. The Selection of Parents and Creation of Offspring. The population maintained by EMAPS is always sorted in order of decreasing fitness scores. The first parent for a crossover is then selected in reference to ranks in the sorted population. The rth

ranked member will be chosen as the first parent with probability (Reeves 1993) $P(x^{[r]}) = 2(n - r + 1)/[n(n + 1)]$. After the second parent is selected on a purely random basis, crossover, repair, and mutation operators appropriate for the problem at hand are used to produce one or more offspring.

3.5.2. Sampling of Weights to Guide Evolutionary Operators. It is often possible to design evolutionary (recombination, repair, etc.) operators that exploit a known base heuristic for the single-objective version of the MOCO problem being addressed. If such operators are to be employed by the algorithm, a rule must be specified for selection of the operator-guidance weight vector w^{og} used to aggregate the objectives and create the single-objective problem to be solved. In EMAPS, the existence of favorable weights proves useful in the choice of w^{og}. For instance, in a recombination operator such as the one described below for multiobjective spanning-tree problems, w^{og} may be set equal to the favorable weights of either of the parents, or to some combination of the two. In some cases, time may be gained by storing information on single-objective problems obtained under a predetermined set of weight vectors, W^{pre} forward filtered from a larger set $W \cap R$. The member of W^{pre} to be used as w^{og} in a given operation may then be found by minimizing some distance metric from favorable weights.

3.5.3. Population Entry and Replacement Rules to Encourage Diversity. If a new offspring has a fitness score that is worse than that of the lowest-ranking solution in the population, it is immediately discarded and referred to as a "stillborn" baby. Solutions that are duplicates (in the objective space) of the existing members of the population are also denied entry, although they are not counted among the stillborn. If the new solution is distinct from the existing population members and not stillborn, then it is inserted into the population at the place indicated by its fitness score.

As the new solution x^{new} enters the population, it is compared to each of the existing members $x^i \in X'$ with respect to their own favorable weights. If any x^i performs worse under its own favorable weights w^i than x^{new}, i.e. we have $\zeta^i = \phi(w^i, x^i) - \phi(w^i, x^{\text{new}}) < 0$, then $x^i \in X'$ is a candidate for removal from the population. We set $\zeta^t = \text{Min}_{\zeta^i<0}\{\zeta^i\}$ and remove the solution $x^t \in X'$ with the most negative ζ^i from the population. The fact that x^{new} outperforms x^t under the weights w^t implies that the search directions favoring the two solutions are not very distant, and that x^{new} is more promising than x^t in its ability to approximate X^{PND}. In this sense, the replacement scheme described above tends to replace population members by similar and better-performing ones, and is therefore expected to encourage diversity in the population.

The population cardinality in EMAPS is upward flexible with a predetermined upper limit. As long as the population size is below this limit, we employ only the replacement policy described above, and increase cardinality by one if no solution with negative ζ^i exists. After the upper limit is reached, a second replacement policy is put into effect: In the absence of solutions with negative ζ^i, the entering solution replaces the member of the population with the lowest fitness score to keep the population size stable.

3.6. Termination

The termination of EMAPS may be based on any of the following conditions:

(i) A predetermined number of new offspring are consecutively stillborn, indicating that the population cannot be improved much further.

(ii) The population has converged, as measured by the percentage of genes that have the same value in a predetermined majority of the population.

(iii) The number of crossovers has reached a predetermined value.

Following termination, the population is purged of dominated solutions and forward filtering may be applied to present the DM with a predetermined number of diverse solutions. Alternatively, the DM may choose to have access to the entire final population, e.g. to perform an interactive search on its members.

3.7. Customization for a Specific Application

To customize EMAPS to fit a given MOCO problem, it is necessary to specify a number of aspects. In addition to aspects such as a genetic representation, crossover and mutation operators and probabilities, repair operators or penalty functions to handle infeasibilities, and terminating conditions that are common to most evolutionary algorithms, it is also necessary to specify the following for EMAPS:

(i) A base heuristic for the single-objective version of the problem.

(ii) Initial and maximum population cardinality values.

(iii) Number of desired seeds in the initial population.

(iv) Number of desired weights in W^{pre}.

(v) Number of desired solutions to be used in favorable weight LPs.

(vi) A value for α, determining the proportion of ε to $\bar{\varepsilon}$ in the fitness definition.

(vii) A rule for the selection of a subset of solutions to be presented to the DM or a method for the DM to search the ending population.

4. Computational Findings

The generic metaheuristic described above was implemented on randomly generated instances of two well-known MOCO problems, the multiobjective knapsack problem (MOKP) and the multiobjective spanning-tree problem (MOST). For each of three problem dimensions given by $p = 2$, 3, and 4, ten instances of MOKP on 200 items, ten instances of MOST on 20 nodes, and ten instances of MOST on 50 nodes were used. Experimental conditions were similar to those of Phelps and Köksalan (2003), where MOKP and MOST problems were used to test an interactive algorithm to find the most preferred solution of the DM.

In all simulation runs, the maximum population cardinality was set to 500 and the maximum number of crossovers to 5,000. Further improvements may be possible by enlarging these values for larger problems, with broad search spaces and wide W sets.

Initial population sizes and characteristics were defined with reference to p, since the search space grows exponentially with the number of objectives. Table 2 indicates the step sizes $1/r$ used to define the sets R. The targeted number of weight vectors forward filtered out of $W \cap R$ and used to seed the population equaled two thirds of the initial population cardinality.

The seed solutions were further forward filtered to obtain twenty dispersed high-quality solutions to be used in favorable-weight LPs. In addition, W^{pre} weight sets of cardinality 10 were forward filtered out of $W \cap R$ and the corresponding solutions used to guide the repair operator in MOKP and the crossover operator in MOST. To counteract the high intensification tendencies inherent in these operators, the mutation probability was set to 0.90. Crossover probability, on the other hand, was specified as 1.00. The algorithm was terminated when either the maximum number of crossovers was realized, or the population converged (meaning that 95% or more of the genes had the same value in 95% or more of the population), or 50 consecutive offspring were stillborn.

The α parameter in the fitness function, measuring the weight of the average vs. the minimum utility surplus, was varied as $\alpha = 0.00$, 0.25, 0.50, 0.75, and 1.00, and the performances of each of these versions, as well as the union of their outputs, were evaluated. The union operation involved first combining the final populations and then deleting duplicated or dominated solutions. As a final point, the performances of forward filtered subsets of the union sets were also investigated.

Table 2 Initial Population Characteristics for EMAPS

p	2	3	4
Initial population cardinality	60	90	150
Targeted number of seed weights	40	60	100
Step size $1/r$	1/800	1/145	1/76
Cardinality of set R	801	10,731	79,079

To evaluate EMAPS's performance, it is necessary to recall that the algorithm aspires to provide the DM with a good approximation of his/her preference-nondominated solution set X^{PND} under the constrained weight set W. If there is actually a single element of W defining the DM's preferences, which cannot be more precisely determined *a priori*, then the performance measure must be in reference to this single underlying utility-maximization problem. If, on the other hand, all elements of W are equally likely to define the DM's preferences, or perhaps different weight vectors are valid for different regions of the efficient frontier, then the performance measure must take the entire W into account.

Our simulation runs on MOKP and MOST involved actual criteria weights w_k^{act} randomly generated from a discrete uniform distribution over $[0, 1]$ and then normalized to sum to one. The utility-maximization problem $\text{Max}_{x \in X} U(x)$ was then solved for the utility functions

$$\text{Linear:}\quad U(x) = -\sum_{k=1}^{p} w_k^{\text{act}}(f_k^* - f_k(x))$$

$$\text{Tchebycheff:}\quad U(x) = -\max_{k=1,\ldots,p} \{w_k^{\text{act}}(f_k^* - f_k(x))\},$$

where f^* is the ideal point for the problem. Linear and Tchebycheff utility functions are commonly used for simulating the DM's preferences in the multicriteria literature (see, for example, Köksalan and Sagala 1995). Performance may hence be evaluated by the scaled deviation percentages

$$\delta = 100\frac{U(x^{\text{BENCH}}) - U(x^{\text{BEST}})}{U(x^{\text{BENCH}}) - U(x^{\text{BAD}})},$$

where x^{BENCH} is a good benchmark solution to the utility maximization problem, x^{BAD} is a poor solution, and x^{BEST} is the solution with highest $U(x)$ value from among all the solutions in the final population of EMAPS or a subset thereof presented to the DM.

In addition, W sets of varying sizes were constructed around the normalized weights w^{act} as

$$W = \left\{ w = (w_1, w_2, \ldots, w_p) \;\middle|\; \sum_{k=1}^{p} w_k = 1, \right.$$
$$\left. w_k^{\text{act}} - \omega \le w_k \le w_k^{\text{act}} + \omega \;\forall k = 1, \ldots, p \right\}$$

for the values $\omega = 0.05$, 0.15, and 0.25. The W sets corresponding to these three ω levels were referred to as the cases of $W1$, $W2$, and $W3$, respectively. The bound constraint on w_k was replaced by $0 \le w_k \le 2\omega$

whenever we had $w_k^{act} - \omega < 0$ and by $1 - 2\omega \le w_k \le 1$ whenever we had $w_k^{act} + \omega > 1$.

The weight vectors falling in the $W \cap R$ sets were used (without filtering) with linear utility functions in the generation of approximate preference-nondominated solution sets $\hat{X}^{PND}$. These sets are approximate in the sense that they do not contain any convex dominated solutions and may not contain all supported efficient solutions in the range since not all possible $w \in W$ are used, although the sampling of W is quite dense.

Scaled deviation percentages $\delta|\hat{x}.$ were calculated for each $\hat{x} \in \hat{X}^{PND}$ and the average $\delta_{avg} = \sum_{\hat{x}\in\hat{X}^{PND}}(\delta|\hat{x})/|\hat{X}^{PND}|$ was used to measure how successfully EMAPS approximates X^{PND} on average. Furthermore, the $\delta|\hat{x}$ value for the solution $\hat{x}$ that is approximated with the least success, $\delta_{max} = \text{Max}_{\hat{x}\in\hat{X}^{PND}}\{\delta|\hat{x}\}$, measured representative ability by identifying any gaps that may exist in the population.

4.1. Implementations on MOKP

For the MOKP, the random generation of the problems, genetic representation, and the crossover and mutation operators are as in Phelps and Köksalan (2003). A greedy algorithm, which fills the knapsack in order of decreasing contribution per unit capacity usage ratios c_j/a_j, was used both as a heuristic to seed the initial population and to perform local search. A reverse greedy algorithm was also used with actual weights w^{act} by filling the knapsack in order of increasing ratios to generate x^{BAD}. The x^{BENCH} solutions in reference to the single utility functions defined by w^{act} were found by an optimization package. However, since the combined cardinality of all the R sets was 906,110, solving all problems induced by the weight vectors in the $W \cap R$ sets proved computationally excessive. The $\hat{X}^{PND}$ sets for MOKP were, therefore, formed by solving linear relaxations of the utility-maximization problems, and it was found that even the δ_{avg} and δ_{max} values computed in reference to these "upper bound" solutions were quite excellent.

In order to maintain the feasibility of offspring, a greedy empty and fill repair operation is adapted from Phelps and Köksalan (2003) as follows: At initialization, the vectors $w^{og} \in W^{pre}$ are used in turn to define a composite objective function coefficients $c_j = \sum_{k=1}^{p} w_k^{og} c_j^k$ and the items sorted with respect to their contribution per unit-capacity-usage ratios c_j/a_j. An additional binary variable in uniform crossover determines which offspring will be assigned the favorable weights of which parent. Then, when an offspring assigned the weight vector w^{asg} is to be repaired, the sorted item list corresponding to the $w^{og} \in W^{pre}$ with the shortest Euclidean distance from w^{asg} is used. If a solution violates the knapsack capacity, items are taken out of the knapsack in the list order until feasibility is restored. If a solution (either upon generation or after an empty operation) does not use the full knapsack capacity it could command, then additional items are put into the knapsack in the list order. For alternative ways of handling the constraints, see Deb (2001, pp. 275–298) and Fonseca and Fleming (1998).

4.2. Implementations on MOST

Random generation of the MOST problems, seeding of the population, finding x^{BENCH} and x^{BAD}, and the mutation operator are as in Phelps and Köksalan (2003). The $\hat{X}^{PND}$ sets approximating the DM's preference-nondominated solutions were found by using the greedy algorithm to solve linear utility-maximization problems to optimality. No genetic representations were utilized and the *subgraph-MST crossover* operator we designed dealt directly with the spanning trees. Subgraph-MST crossover is based on the greedy algorithm and finds the single-objective minimum spanning tree of the subgraph obtained by taking only the edges of the parent trees into consideration. At initialization, sorted arc lists are formed and stored for ten different composite arc cost values corresponding to $w^{og} \in W^{pre}$. Then, when parent solutions x^{P1} and x^{P2} are to be recombined, the subgraph-MST crossover operator is applied twice to the subgraph formed by juxtaposing the arcs of x^{P1} and x^{P2}, yielding two offspring solutions x^{C1} and x^{C2}. The composite arc costs used to create x^{C1} are those found for the member of W^{pre} with shortest Euclidean distance from x^{P1}'s preferred weights w^{P1}, while those used to create x^{C2} correspond to the member of W^{pre} with shortest Euclidean distance from w^{P2}.

4.3. Simulation Results

Table 3 below presents the average of the δ_{avg} values and the maximum of the δ_{max} values observed over the ten problem instances of each problem type. In each row, the minimum value for the EMAPS versions corresponding to the α levels of 0.00, 0.25, 0.50, 0.75, and 1.00 is marked with an asterisk (*). Table 3 also includes the results for the union operation and for forward-filtered subsets of the union operation of cardinality 20 (filt20) and 50 (filt50). Space restrictions prevent us from presenting similar tables for running times and δ values for the case of a single underlying utility function. The maximum ranges for δ values under linear utility functions were 0.35, 1.10, and 4.00, and those under Tchebycheff utility functions were 1.20, 7.00, and 12.00, for MOKP, MOST20, and MOST50, respectively.

As might be expected, δ values increase when the form of the DM's utility function is very different from that assumed by the algorithm, when the problem dimension is high, and when the precision of the

Table 3 **Average δ_{avg} (and Maximum δ_{max}) Values**

			α							
Problem	W	p	0	0.25	0.5	0.75	1	Union	filt20	filt50
MOKP	$W1$	2	*0.08 (0.22)	*0.08 (*0.16)	*0.08 (*0.16)	*0.08 (0.22)	0.09 (0.20)	0.07 (0.16)	0.08 (0.16)	0.08 (0.16)
		3	0.11 (0.27)	0.11 (0.29)	*0.10 (0.27)	*0.10 (*0.23)	0.11 (0.27)	0.08 (0.23)	0.11 (0.23)	0.10 (0.24)
		4	0.10 (*0.49)	0.09 (*0.49)	*0.08 (*0.49)	0.09 (*0.49)	0.09 (*0.49)	0.08 (0.49)	0.10 (0.49)	0.09 (0.53)
	$W2$	2	0.10 (0.56)	*0.09 (*0.27)	0.11 (0.39)	0.11 (0.28)	0.11 (0.29)	0.07 (0.18)	0.09 (0.18)	0.11 (0.18)
		3	0.15 (0.59)	0.13 (0.49)	*0.11 (*0.44)	*0.11 (0.46)	0.12 (0.51)	0.09 (0.35)	0.13 (0.35)	0.11 (0.36)
		4	0.22 (1.39)	*0.19 (1.13)	*0.19 (*0.91)	*0.19 (1.65)	0.22 (1.69)	0.14 (0.83)	0.21 (0.83)	0.22 (0.92)
	$W3$	2	0.19 (0.94)	*0.15 (0.91)	*0.15 (*0.53)	0.17 (0.59)	0.17 (0.64)	0.09 (0.35)	0.15 (0.35)	0.15 (0.35)
		3	0.21 (1.47)	*0.16 (*0.71)	0.17 (0.77)	0.20 (0.73)	0.23 (0.89)	0.12 (0.51)	0.17 (0.51)	0.18 (0.60)
		4	0.43 (3.52)	0.35 (*1.73)	*0.34 (1.77)	0.35 (1.81)	0.38 (2.21)	0.25 (1.10)	0.39 (1.10)	0.38 (1.21)
MOST20	$W1$	2	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
		3	*0.00 (0.02)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (0.02)	*0.00 (0.02)	0.00 (0.00)	0.00 (0.22)	0.00 (0.11)
		4	*0.00 (*0.05)	*0.00 (*0.05)	*0.00 (*0.05)	*0.00 (*0.05)	*0.00 (*0.05)	0.00 (0.05)	0.01 (0.30)	0.04 (0.18)
	$W2$	2	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (0.01)	*0.00 (0.29)	0.00 (0.00)	0.00 (0.01)	0.00 (0.00)
		3	*0.00 (0.11)	*0.00 (*0.06)	*0.00 (0.16)	0.01 (0.28)	0.01 (0.36)	0.00 (0.06)	0.04 (0.51)	0.02 (0.23)
		4	*0.04 (*0.65)	0.05 (1.00)	0.09 (1.34)	0.10 (1.53)	0.18 (2.10)	0.02 (0.49)	0.18 (1.90)	0.13 (1.10)
	$W3$	2	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (0.21)	0.01 (0.17)	0.02 (0.64)	0.00 (0.00)	0.01 (0.20)	0.00 (0.00)
		3	*0.01 (*0.26)	0.04 (0.97)	0.08 (1.47)	0.17 (2.19)	0.17 (3.08)	0.01 (0.22)	0.31 (0.82)	0.05 (0.57)
		4	*0.15 (*2.04)	0.67 (5.48)	0.67 (4.53)	0.70 (4.85)	0.73 (5.67)	0.12 (1.43)	1.03 (4.07)	0.34 (2.49)
MOST50	$W1$	2	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	*0.00 (*0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
		3	*0.00 (*0.01)	*0.00 (*0.01)	*0.00 (0.02)	*0.00 (0.02)	*0.00 (0.05)	0.00 (0.01)	0.01 (0.07)	0.00 (0.03)
		4	*0.00 (*0.07)	*0.00 (*0.07)	*0.00 (*0.07)	0.01 (*0.07)	0.01 (*0.07)	0.00 (0.07)	0.04 (0.16)	0.01 (0.07)
	$W2$	2	*0.00 (*0.00)	*0.00 (0.04)	*0.00 (0.12)	*0.00 (0.03)	*0.00 (0.03)	0.00 (0.00)	0.00 (0.01)	0.00 (0.00)
		3	*0.01 (*0.13)	0.03 (0.69)	0.03 (0.69)	0.05 (0.58)	0.07 (0.70)	0.01 (0.11)	0.04 (0.28)	0.04 (0.13)
		4	*0.08 (*0.79)	0.15 (1.63)	0.22 (2.89)	0.34 (3.46)	0.51 (3.22)	0.07 (0.58)	0.28 (1.22)	0.25 (0.94)
	$W3$	2	*0.00 (*0.01)	0.01 (0.24)	*0.00 (0.19)	0.01 (0.33)	0.11 (1.26)	0.00 (0.01)	0.00 (0.03)	0.00 (0.01)
		3	*0.04 (*0.52)	0.30 (3.04)	0.28 (3.34)	0.30 (3.31)	0.51 (4.15)	0.03 (0.30)	0.13 (0.96)	0.29 (0.40)
		4	*0.27 (*4.06)	0.99 (8.20)	1.13 (6.54)	1.10 (6.57)	1.58 (8.85)	0.22 (1.81)	0.74 (3.54)	1.17 (2.22)

partial information on DM preferences is low. High values of the α parameter also seem to weaken the evolutionary forces involved in the algorithm, resulting in high δ levels. Simply setting $\alpha = 0.00$ and working only with the minimum-utility surplus seems to yield quite successful results for MOST problems, while α levels of 0.50 or 0.25 seem to work better for MOKP. Studying the distribution of asterisks in Table 3, we find that once again $\alpha = 0.25$ and 0.50 work well for MOKP, while $\alpha = 0.00$ and 0.25 perform better for MOST problems.

When the union is taken of the different versions, average δ_{avg} values remain below 0.25%, maximum δ_{max} values below 1.81%, and average δ values below 0.32% with a linear underlying utility function and below 2.14% with a Tchebycheff underlying utility function for all problems. Even if the DM does not wish to search the entire set of solutions resulting from the union operation, the results remain encouraging for forward-filtered subsets. When the DM is presented a set of at most 20 solutions, if s/he is able to choose the best therein, average δ and δ_{avg} values are below 1.2% and δ_{max} values are below 4.5% in all cases with linear underlying utility functions. When the sample size increases to 50, the corresponding figures fall to 1.0% and 2.5%. When the underlying utility function of the DM differs sharply from what is assumed by the algorithm, however, average δ values can go up to 7.0% for 20-solution samples and 3.5% for 50-solution samples, so conducting an interactive search over the entire union set may be worthwhile.

In the final populations, we observe that there are many convex dominated solutions in both problems. This supports our previous illustration that convex dominated solutions can survive throughout the evolution process.

Computation durations, like δ values, tend to increase when there are increases in size of the problem, the dimension of the problem, or the size of the restricted weight-space. We find that extreme levels of 0 and 1 for α result in shorter run times, conceivably because they simplify the LPs that must be solved for favorable weight computations. Maximum values for a 233 MHz Pentium II processor were 400 seconds for MOKP and MOST20, and 600 seconds for MOST20. If we ignore the extreme values for $\alpha = 0.75$, average durations are under seven minutes for all cases. Combining and, if necessary, filtering the

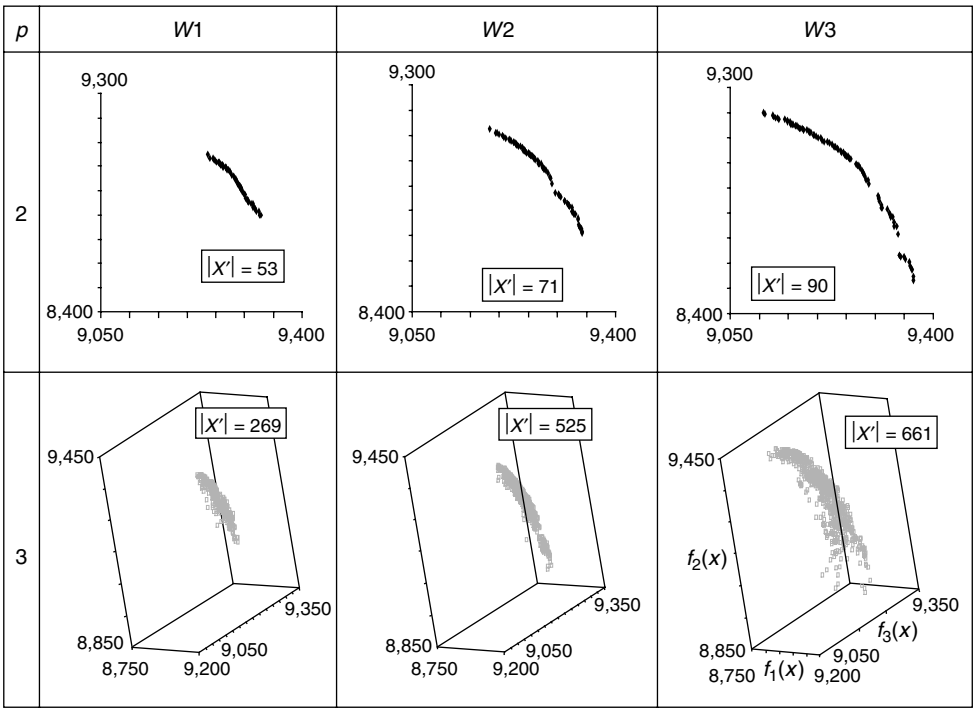

Figure 2 Illustration of the Effect of Precision on EMAPS

results of the five versions takes less than 40 minutes for all problems.

Finally, the simulation results illustrate the focusing capability of the EMAPS fitness function when one compares the union sets obtained for the same problem under different degrees of precision. Figure 2 displays typical results for one 2-dimensional and one 3-dimensional problem. It is quite clear in the figure that, as the size of the restricted weight space W increases, the cardinality and spread of the union set increases to approximate a wider range of preference-nondominated solutions.

5. Conclusions

The simulation runs discussed above indicate that EMAPS succeeds in finding a high-quality approximation of the DM's preference-nondominated set in all cases. The effect of the parameter α, controlling the balance in the fitness function of the minimum and the average utility surplus, seems to follow no fixed pattern. Further research on this effect remains of interest, as does the design of a dynamic metaheuristic where α is progressively adjusted. In our simulations, α values of 0.00, 0.25, and 0.50 appeared to yield good results. The final populations reported under different levels of α were distinct but mostly of equally high quality. When the problem size or dimension is large, or the partial information available on the DM's preferences is not very precise, the set X^{PDM} to be approximated grows quite large and running EMAPS under several different levels of α and combining the results emerges as a good alternative to increasing the maximum number of crossovers or the maximum population cardinality allowed, especially if the latter is not feasible with the given computational resources. When the resulting set from the union operation is filtered to obtain a smaller subset of disperse solutions, a representative sample of high-quality solutions may be obtained.

The fitness function designed for EMAPS makes it possible for the population to move in all three of the desired evolutionary directions simultaneously without requiring niching, fitness sharing, or other mechanisms traditionally employed to guarantee a good spread of the population over the efficient frontier. When a positive value is used for the parameter α, convex-dominated solutions in otherwise underrepresented regions of the set of preference-nondominated solutions may have higher fitness than supported efficient solutions in crowded regions. This may improve the metaheuristic's performance for nonlinear utility functions, especially when a small population is used.

Although the LPs solved to find the favorable weights of solutions are small, they need to be solved frequently and they are the main factors affecting the solution times. Finding a more efficient way of handling this step would improve computational performance.

Comparisons of the algorithm with others in the literature are desirable, and may be carried out for the no-information case where the entire efficient frontier

is being approximated. The method can easily be customized to solve different MOCO problems as well as other multiobjective problems that cannot be solved analytically. The evolution operators and base heuristics required for customization may in many cases be readily adapted from the vast catalog of evolutionary algorithms that have been developed in recent years to address single-objective combinatorial optimization problems.

Acknowledgments

The second author was supported by the Scientific and Technical Research Council of Turkey (TÜBİTAK), and both authors visited the Krannert School of Management, Purdue University, during a part of this research.

References

Athanassopoulos, A. D., V. V. Podinovski. 1997. Dominance and potential optimality in multiple criteria decision analysis with imprecise information. *J. Oper. Res. Soc.* **48** 142–150.

Coello Coello, C. A. 1999. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Inform. Systems* **1** 269–308.

Coello Coello, C. A. 2000. Handling preferences in evolutionary multiobjective optimization: A survey. *2000 Congress Evolutionary Comput.* **1** 30–37.

Czyzak, P., A. Jaszkiewicz. 1998. Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization. *J. Multi-Criteria Decision Anal.* **7** 34–47.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK.

Ehrgott, M., X. Gandibleux. 2002. Multiobjective combinatorial optimization. M. Ehrgott, X. Gandibleux, eds. *Multiple Criteria Optimization—State of the Art Annotated Bibliographic Surveys*, Vol. 52. Kluwer Academic Publishers, Boston, MA, 369–444.

Ehrgott, M., X. Gandibleux. 2004. Approximative solution methods for multiobjective combinatorial optimization. *TOP* **12** 1–89.

Fonseca, C. M., P. J. Fleming. 1998. Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: A unified formulation. *IEEE Trans. SMC-Part A: Systems and Humans* **28** 26–37.

Gandibleux, X., N. Mezdaoui, A. Freville. 1996. A tabu search procedure to solve multiobjective combinatorial optimization problems. R. Caballero, R. Steuer, eds. *Proc. MOPGP'96*. Springer-Verlag, Berlin.

Hajela, P., C. Y. Lin. 1992. Genetic search strategies in multicriteria optimal design. *Structural Optim.* **4** 99–107.

Hazen, G. B. 1986. Partial information, dominance, and potential optimality in multiattribute utility theory. *Oper. Res.* **34** 296–310.

Köksalan, M. M., P. N. S. Sagala. 1995. Interactive approaches for discrete alternative multiple criteria decision making with monotone utility functions. *Management Sci.* **41** 1158–1171.

Phelps, S., M. Köksalan. 2003. An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Management Sci.* **49** 1726–1738.

Reeves, C. R., ed. 1993. *Modern Heuristic Techniques for Combinatorial Problems*. Wiley, New York.

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

Stewart, T. J. 1992. A critical survey on the status of multiple criteria decision making theory and practice. *OMEGA Internat. J. Management Sci.* **20** 569–586.

Teghem, J., D. Tuyttens, E. L. Ulungu. 2000. An interactive heuristic method for multiobjective combinatorial optimization. *Comput. Oper. Res.* **27** 621–634.

Ulungu, E. L., J. Teghem. 1994. Multi-objective combinatorial optimization problems: A survey. *J. Multi-Criteria Decision Anal.* **3** 83–104.

Weber, M. 1987. Decision making with incomplete information. *Eur. J. Oper. Res.* **28** 44–57.

Zitzler, E., L. Thiele. 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Comput.* **3** 257–271.