Annals of Operations Research 131, 79–99, 2004

Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection

KARL DOERNER karl.doerner@univie.ac.at
Department of Management Science, University of Vienna, Bruenner Str. 72, A-1210 Vienna, Austria

WALTER J. GUTJAHR walter.gutjahr@univie.ac.at
Department of Statistics and Decision Support Systems, University of Vienna, Universitaetsstr. 5/3, A-1010 Vienna, Austria

RICHARD F. HARTL, CHRISTINE STRAUSS and CHRISTIAN STUMMER
{richard.hartl, christine.strauss, christian.stummer}@univie.ac.at
Department of Management Science, University of Vienna, Bruenner Str. 72, A-1210 Vienna, Austria

Abstract. Selecting the "best" project portfolio out of a given set of investment proposals is a common and often critical management issue. Decision-makers must regularly consider multiple objectives and often have little a priori preference information available to them. Given these contraints, they can improve their chances of achieving success by following a two-phase procedure that first determines the solution space of all efficient (i.e., Pareto-optimal) portfolios and then allows them to interactively explore that space. However, the task of determining the solution space is not trivial: brute-force complete enumeration only works for small instances and the underlying NP-hard problem becomes increasingly demanding as the number of projects grows. Meta-heuristics provide a useful compromise between the amount of computation time necessary and the quality of the approximated solution space. This paper introduces Pareto Ant Colony Optimization as an especially effective meta-heuristic for solving the portfolio selection problem and compares its performance to other heuristic approaches (i.e., Pareto Simulated Annealing and the Non-Dominated Sorting Genetic Algorithm) by means of computational experiments with random instances. Furthermore, we provide a numerical example based on real world data.

Keywords: ant colony optimization, simulated annealing, genetic algorithms, portfolio selection, multiobjective combinatorial optimization

1. Introduction

In most real-life situations, decisions are made in the presence of multiple objectives that are often conflicting. In addition, many of the problems are combinatorial (Nemhauser and Wolsey, 1988). Consequently, researchers from operational research and management science have constituted scientific communities dedicated to multiobjective decision-making and combinatorial optimization, respectively; these fields have attracted a tremendous amount of activity during the past few decades (cf. Steuer, Gardiner, and Gray, 1996, and Dell'Amico, Maffioli, and Martello, 1997, for bibliographies). Together, they play a decisive role in multiobjective combinatorial optimization

(MOCO; cf. Ehrgott and Gandibleux, 2000, for a survey and White, 1990, and Ulungu and Teghem, 1994, for references to applications), for which the branch of portfolio selection is of particularly high practical relevance. Research and development (R&D) management provides an especially illustrative example for corresponding implications: in the increasingly competitive, global marketplace, innovation is often cited as an important strategy for survival and R&D therefore has a key role to play in a firm's future success. As a consequence, it is imperative for enterprises to determine the "best" subset of R&D projects out of dozens of competing proposals (i.e., to identify that project portfolio which provides the most attractive mix of benefits with respect to given management objectives).

In a multiobjective portfolio selection model, difficulties naturally arise in formulating an appropriate objective function. Basically, two ways of proceeding exist. The first approach involves building a function that aggregates the different attributes (e.g., cash flow, sales or even such intangibles as image) that characterize the attractiveness of any given portfolio and thus, as far as possible, reflects its overall benefit. A major drawback to this approach lies in the fact that it requires extensive a priori preference information (e.g., weights, thresholds, marginal benefits, or guidelines for benefit or resource substitution between different categories). In addition empirical evidence suggests that such an approach actually performs relatively poorly in the case of multiple objective mathematical programming (Corner and Buchanan, 1995). A different approach lies in accepting several criteria within the model and (partially) determining the efficient (i.e., non-dominated or Pareto-optimal) portfolio candidates. After this initial phase, the decision-maker is given an opportunity to explore the solution space on the basis of guidance provided by an interactive procedure involving sets of alternatives that are explicitly given. This exploration continues until a satisfactory portfolio is found. Regularly, this approach can be undertaken without the above mentioned preference data. However, the process involved in identifying the set of efficient portfolios is not trivial. While a brute-force complete enumeration procedure can determine them within acceptable time for comparatively small problems, that task becomes increasingly demanding as the number of projects grows.

When decision-makers are confronted with a large number of competing projects, heuristic approaches provide a tradeoff between the quality of the solution space and the computational effort required to achieve this approximation. Several adaptions of metaheuristic procedures have already been proposed: the most common one being the genetic algorithm (GA). Since the pioneering method by Schaffer (1985), numerous related approaches have been published (see Fonseca and Fleming, 1993; Horn, Nafpliotis, and Goldberg, 1994; Srinivas and Deb, 1994; Murata and Ishibuchi, 1995; Coello and Christiansen, 1998; Zitzler and Thiele, 1999; Hanne, 2000, for examples and Coello, 2000; Deb, 2001, for surveys). A promising alternative known as simulated annealing (SA) was discussed in Serafini (1994) and subsequently refined by Czyzak and Jaszkiewicz (1998), Ulungu, Teghem, and Ost (1998), and Hapke, Jaszkiewicz, and Slowinski (2000). Tabu search (TS) approaches (e.g., Gandibleux, Mezdaoui, and Freville, 1997; Ben Abdelaziz, Chaouachi, and Krichen, 1999; Hansen, 2000; Alves and

Climaco, 2000) form the third major class of heuristic procedures for multiobjective combinatorial optimization (MOCO) problems.

We aim at providing a heuristic approach in the field of multiobjective portfolio selection by introducing Pareto Ant Colony Optimization (P-ACO; cf. Doerner et al., 2001a, 2002b), an extension of the traditional Ant System (AS; cf. Dorigo, 1992, 1996) and Ant Colony Optimization (ACO; cf. Dorigo and Di Caro, 1999), respectively. So far, Gambardella, Taillard, and Agazzi (1999) developed an ant algorithm for a bi-criterion vehicle routing problem, basing their approach on the assumption that the two criteria can be ordered lexicographically. This multi-colony approach uses one ant colony for each objective; its applicability is limited to those problems for which priorities can be defined for the objectives. Further, Iredi, Merkle, and Middendorf (2001) and McMullen (2001) developed an Ant Colony system for optimization problems that consist of two objectives only and applied it to sequencing problems. Our approach differs not only in the problem class but also in the number of optimization criteria, e.g., five to ten objectives in the numerical examples. Iredi, Merkle, and Middendorf (2001) try to cover the whole set of possible weights by using 100 ants, each of which has a different deterministic weighting vector. Those ants are grouped in 10 heterogeneous populations. While this is reasonable for problems with two objectives, this does not necessarily hold in the case of proper multiobjective problems because of the many weighting vectors necessary. Therefore, we use a single population with each ant having different, randomly generated weights (cf. Doerner et al., 2001a). The increased complexity in the problem structure further requires a tool to administrate the numerous (up to several thousands) potentially efficient portfolios in reasonable computation time; for that purpose we designed a generalized quad tree for our ACO implementation. Moreover, we used a different pheromone strategy which conforms better to the significantly higher complexity. It should be noticed that the administration of non-dominated solutions is computationally trivial in the bi-criterion case, whereas it is not in the proper multicriteria case. The reason lies in the number of efficient solutions that usually is significantly higher in the latter case. Moreover, from a computational effort point of view the determination whether a given solution actually is efficient and whether it dominates some other proposed efficient solutions is expensive while it is simple in the bi-criterion case, where the already identified efficient solutions easily can be sorted by one criterion and automatically are sorted (in reversed order) by the other as well. Then one simply has to find the right position for the current solution for one criterion and immediately gets access to all relevant (potentially dominated) solutions because they will be direct neighbours.

In contrast to the adaptive GA, SA, and TS heuristics, P-ACO constructs its portfolios. Thus, it largely avoids infeasible portfolio candidates by explicitly taking into consideration even complex project interactions. The Ant Colony approach imitates the behavior shown by real ants when searching for food. Ants communicate information about food sources via the quantity of an aromatic essence called pheromone, which the ants secrete as they move along. Over time, the short direct paths leading from the nest to a food source are more frequented than longer paths. As a result, the di-

rect paths are marked with more pheromone, which in turn attracts an ever increasing number of ants to follow these shorter routes and make the corresponding pheromone trails grow faster. Artificial ants not only imitate the behavior described, but also apply additional, problem-specific heuristic information. The Ant System has been applied to and provided solutions for various hard combinatorial optimization problems (cf. Dorigo and Gambardella, 1997; Bullnheimer, Hartl, and Strauss, 1999b; Gambardella, Taillard, and Agazzi, 1999; Stuetzle and Dorigo, 1999; Bauer et al., 2000; Doerner, Hartl, and Reimann, 2001b; Doerner et al., 2002a) and a convergence proof for a generalized Ant System algorithm has been established (Gutjahr, 2002). In order to meet multiobjective problem specific requirements, our P-ACO approach implements several pheromone vectors and applies random weights for their use. The lifespan concept and the pheromone decoding scheme are two additional salient features that play an essential role in modelling the portfolio selection process.

2. Problem description

Portfolios may be described as subsets of the set of all N project proposals; they are modeled as vectors $x = (x_1, \ldots, x_N)$, where the binary variables x_i indicate whether project i is included in the portfolio ($x_i = 1$) or not ($x_i = 0$). Our approach aims at determining the efficient project portfolios (i.e., for them no other feasible alternative exists that promises higher values in at least one of the objectives and offers at least the same in all the others). Following the model introduced by Stummer (1998) and Stummer and Heidenberger (2001), respectively, a project i is characterized both by the benefits $b_{i,l,t}$ it provides in the B benefit categories l (e.g., cash flow, sales, and patents) and the T planning periods t (e.g., financial years), as well as by its resource consumption $r_{i,q,t}$ in the R resource categories q (e.g., funds, manpower, and production capacity). The benefit value for a portfolio x is given by

$$b_{l,t}(x) = \sum_{i=1}^{N} b_{i,l,t} \cdot x_i + \sum_{j=1}^{V} v_j(x) \cdot v_{j,l,t} + \sum_{j=1}^{W} w_j(x) \cdot w_{j,l,t} \quad \text{for } l = 1, \ldots, B \text{ and } t = 1, \ldots, T, \tag{1}$$

where the sum of the individual project benefits must be adjusted by the effects of $V + W$ project interactions. The V interactions of the first type refer to subsets $V_j = \{i \in N\colon\ v_{i,j} = 1\}$ (with $v_{i,j} = 1$ if project i is effected by an interaction j and $v_{i,j} = 0$ otherwise) and generate benefits $v_{j,l,t}$ only if at least m_j of these projects are included in the portfolio:

$$v_j(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} x_i \cdot v_{i,j} \geqslant m_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The W additional interactions provide benefits of $w_{j,l,t}$ for portfolios containing no more than a given maximum number of projects out of subsets W_j.

The model similarly determines necessary resources $r_{q,t}(x)$ for resource category q (with $q = 1, \ldots, R$) and planning periods t. Furthermore, it traces both benefit and resource values separately for each period instead of aggregating them to a (discounted) overall value and, thus, provides additional information for the decision maker (see Ringuest and Graves, 1990, for a discussion). As a consequence, a comparatively high number of $K = B \cdot T$ objectives

$$u_k(x) = b_{l,t}(x) \quad \text{for } k = l + (t-1) \cdot B \tag{3}$$

have to be handled just for the benefit categories. Moreover, remaining resources also may be considered as additional $R \cdot T$ objectives

$$u_k(x) = R_{q,t} - r_{q,t}(x) \quad \text{for } k = B \cdot T + q + (t-1) \cdot R \tag{4}$$

where $R_{q,t}$ stands for the corresponding resource limitations.

The above objectives are subject to two groups of constraints. The first group ensures that no more than a given maximum (not less than a given minimum) number of projects may appear in a portfolio in relation to a given subset of projects. This set of constraints guarantees the selection of a minimum number of projects that, for example, deal with emerging technologies, restrict the number of projects based on conventional concepts (even if they seem attractive in a short-time perspective) or deal with balancing aspects (e.g., with respect to new and current projects). These constraints may be expressed as

$$\sum_{i=1}^{N} \tilde{v}_{i,j} \cdot x_i \geqslant \tilde{m}_j \tag{5}$$

where $\tilde{m}_j$ represents a minimum number of certain projects that must be included in a portfolio and $\tilde{v}_{i,j}$ indicates whether project i is in the corresponding subset j of effected projects. Inequalities for a constraint variation demanding a maximum number of projects take a similar form.

The second set of constraints concerns resource limitations $R_{q,t}$ and minimum benefit requirements $B_{l,t}$. They can be written as

$$r_{q,t}(x) \leqslant R_{q,t} \quad \text{for } q = 1, \ldots, R \text{ and } t = 1, \ldots, T, \quad \text{and} \tag{6}$$

$$b_{l,t}(x) \geqslant B_{l,t} \quad \text{for } l = 1, \ldots, B \text{ and } t = 1, \ldots, T. \tag{7}$$

3. Solution procedures

The above problem is characterized by numerous constraints as well as a large number of objectives. The first-mentioned entail a considerable high percentage of infeasible portfolio candidates. Due to the latter substantial computational effort has to be expended

in order to take into account the various project interactions and to accordingly determine all the objective values (i.e., the evaluation of a portfolio is quite "expensive"). Moreover, the investigation whether a current portfolio may be considered as efficient or not becomes time-consuming, too. We therefore use a quad tree data structure for identifying, storing and retrieving non-dominated portfolios.

Such quad trees generalize classic binary trees to K-dimensional space. First introduced by Finkel and Bentley (1974) for data storage and retrieval, they have been applied to discrete vector optimization problems by Habenicht (1983). The project portfolios are stored in the nodes of the tree. Given K objectives, a node is followed by up to $2^K - 2$ subtrees, where all portfolios in such a subtree have the same dominance relation (i.e., for each objective they are all better or all worse, respectively, than the root). With this hierarchical structure, only a small percentage of all possible pairwise comparisons is required for efficiency verification (for a recent discussion cf. Sun and Steuer, 2000).

The remainder of this section decribes in detail the Pareto Ant Colony Optimization (P-ACO) and two solution procedures for benchmarking: Pareto Simulated Annealing (PSA) and the Non-Dominated Sorting Genetic Algorithm (NSGA).

3.1. Pareto Ant Colony Optimization

In the initialization phase, Γ ants are generated, each ant starting with an empty portfolio $x = (0, \ldots, 0)$. The lifespan Ξ and the objective weights (i.e., the ant's individual preferences) $p = (p_1, \ldots, p_K)$ are determined randomly for each ant. Note, that whenever we do not explicitly mention the domain random numbers are chosen equally distributed from the domain [0, 1).

In the construction phase of the algorithm, each ant tries to construct a feasible portfolio x by applying a pseudo-random-proportional rule using heuristic information η_i and pheromone information τ_i. After a portfolio has been constructed, its feasibility and efficiency is determined. If the portfolio under consideration is feasible and efficient it is stored. Global pheromone update is performed by using the best and the second-best portfolio x of the current iteration for each objective k.

The proposed P-ACO algorithm for the problem at hand is the following:

procedure `P-ACO` () {
  Initialization of `P-ACO`; /* create Γ ants, initialize pheromone vectors with τ_0 */
  repeat until termination criterion is **true** {
    for $Ant = 1$ **to** Γ {
      determine the lifespan Ξ of the ant randomly on the interval $[1 \ldots N]$;
      set $x = (0, \ldots, 0)$; /* create empty portfolio */
      determine the objective weight p_k for each objective k randomly;
      $\xi = \Xi$; /* indicates the number of projects to be selected */
      while $\xi > 0$ **and** $\exists\, i$ with $\eta_i(x) > 0$ {
        select a project i using formula (8) below and add it to x;

```
            update local pheromone information;
            decrement ξ;
        }
        check feasibility of portfolio x;
        if portfolio x is feasible {
            check efficiency of portfolio x;
            if portfolio x is efficient {
                store portfolio x and remove dominated ones;
    } } }
    for each objective k {
        determine best and second-best solution u_k for each objective k;
        update global pheromone information using best and second-best solution
        using formula (11)
} } }
```

In this pseudocode and in what follows the term "randomly generated" means generated according to a uniform distribution.

3.1.1. Heuristic information

The heuristic information is based on a quantitative value that measures how well some project candidate fits into a partially constructed portfolio. For each project candidate i an aggregated value of attractiveness $\eta_i(x)$ is computed. This value depends on the (partial) portfolio x. Furthermore, it is based on constraints and targets that can be categorized into four categories: maximum or minimum restrictions (e.g., upper/lower limit for the number of projects of a certain project type in the portfolio), resource restrictions (e.g., maximum available workforce) and benefit restrictions (e.g., minimum profit expectations). If a maximum restriction or a resource restriction is violated, then the attractiveness value is set to zero. If the maximum restriction and the resource restriction is fulfilled, then the attractiveness value corresponds to the degree of fulfillment in the two remaining categories (i.e., minimum restrictions and/or benefit restrictions). A special case occurs when all restrictions are satisfied by including the considered project in the portfolio; in this case, the attractiveness value is set to one.

3.1.2. Pheromone information

For each objective k the pheromone information is stored in a vector τ, with the number of elements corresponding to the number of projects. The value τ_i^k represents the current pheromone information, i.e., the pheromone information with respect to objective k of including project i in a "good" portfolio. Roli, Blum, and Dorigo (2001) proposed three different pheromone decoding schemes for maximal constraint satisfaction problems; two decoding schemes were based on matrices and one was vector-based. Based on previous work by Doerner et al. (2001a, 2002b) further extensive testing on our multiobjective portfolio selection problem showed that the vector-based ACS-comp outperformed the two other variants with regard to solution quality for given computa-

tion times. In light of this finding, we will refer solely to the ACS-comp implementation in the remainder of this paper.

3.1.3. Decision rule

Given the attractiveness, the pheromone information, and the set of all feasible projects $\Omega(x) = \{i \in N:\ \eta_i(x) > 0,\ x_i = 0\}$, a feasible project i is selected to be added to the current portfolio x according to a pseudo-random-proportional rule that can be stated as follows:

$$i = \begin{cases} \arg\max\limits_{i \in \Omega(x)} \left\{ \left[\sum\limits_{k=1}^{K} \left(p_k \cdot \tau_i^k\right) \right]^{\alpha} \cdot \left[\eta_i(x)\right]^{\beta} \right\} & \text{if } q \leqslant q_0, \\ \hat{i} & \text{otherwise,} \end{cases} \tag{8}$$

where q is a random number uniformly distributed in [0..1), q_0 is a parameter ($0 \leqslant q_0 < 1$) to be set by the user representing the probability that the portfolio is chosen which gives the highest aggregate value of pheromone and attractiveness. The random variable $\hat{i}$ is selected according to the probability distribution given:

$$\mathcal{P}_i(x) = \begin{cases} \dfrac{\left[\sum_{k=1}^{K}(p_k \cdot \tau_i^k)\right]^{\alpha} \cdot [\eta_i(x)]^{\beta}}{\sum_{h \in \Omega(x)} \left(\left[\sum_{k=1}^{K}(p_k \cdot \tau_h^k)\right]^{\alpha} \cdot [\eta_h(x)]^{\beta}\right)} & \text{if } i \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

This probability distribution is biased by the parameters α and β, which determine the relative influence of the trails and the visibility, respectively.

3.1.4. Pheromone update

A local pheromone update is performed once an artificial ant has added a project to a portfolio. When an ant selects a project i, the amount of pheromone on the elements τ_i^k of the pheromone vector is decreased for each objective k. The local pheromone update rule for these elements can be stated as follows:

$$\tau_i^k = (1 - \rho) \cdot \tau_i^k + \rho \cdot \tau_0, \tag{10}$$

where τ_0 is the initial value of trails and ρ is the evaporation rate. On account of local updating, ants prefer those combinations of orders that have not yet been chosen. As a result, the diversity of the solutions provided is enhanced.

Global pheromone information is updated once each ant of the population has constructed a solution, and the feasibility and efficiency have been determined. Preliminary tests have shown that a pheromone update procedure suffices in which only the best and the second-best solution provided by an iteration is used for global updating (cf. Dorigo and Gambardella, 1997; Bullnheimer, Hartl, and Strauss, 1999a). The update rule for each objective k is as follows:

$$\tau_i^k = (1 - \rho) \cdot \tau_i^k + \rho \cdot \Delta\tau_i^k, \tag{11}$$

where ρ is the evaporation rate (with $0 \leqslant \rho \leqslant 1$). Pheromone information is increased by a quantity $\Delta\tau_i^k$ if a project i is in a portfolio of a population's best (second-best, respectively) ant according to objective k. This update quantity for the best ant can be represented as

$$\Delta\tau_i^k = \begin{cases} 10 & \text{if } x^k_{\text{best},i} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

After the update with respect to the best ant is performed according to (11) and (12) a similar update (11) is made with respect to the second best ant where the update quantity can be written as

$$\Delta\tau_i^k = \begin{cases} 5 & \text{if } x^k_{\text{second-best},i} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Tests with various pheromone update strategies have shown that using the two best ants with pheromone quantity 10 for the best ant and 5 for the second-best ant leads to good results. Furthermore, the performance is not very sensitive with respect to these parameters.

3.2. Pareto Simulated Annealing

For our simulated annealing implementation, we use a technique by Czyzak and Jaszkiewicz (1998) called "Pareto Simulated Annealing" (PSA). This is an extension of the multiobjective simulated annealing algorithms proposed by Serafini (1994) and by Ulungu, Teghem, and Fortemps (1995). The latter approaches already use a population M of potentially efficient solutions rather than a single current solution, as is the case in classical, single-objective simulated annealing. The new features of Czyzak and Jaszkiewicz's extension are to allow a kind of interaction among the current solutions in the population M (i.e., they are updated in such a way that they evolve as distant from each other as possible), and to iteratively modify weights assigned to the objective criteria.

Our basic implementation of the PSA algorithm for the problem under consideration in this paper is described below in pseudocode. Let us use the following notation (K and N are the number of criteria and the number of projects, respectively, as before):

S: sample set of current (feasible) solutions,
s: number of elements in S (fixed positive integer parameter),
p: probability of choosing a 1-bit when generating an initial solution x in S (fixed parameter between 0 and 1),
M: solution set (i.e., set of all proposed efficient solutions in current iteration),
w_{ik}: weight of criterion k for the ith element of sample set S,
$u_k(x)$: objective value (to be maximized) of criterion k for solution x,
a: weight modification factor (fixed parameter greater than 1),
T: temperature parameter for simulated annealing,

L: number of iterations on each temperature level of the simulated annealing algorithm (fixed positive integer parameter),
b: temperature reduction (annealing) factor (fixed parameter smaller than 1, near 1).

Now, the algorithm is the following:

```
procedure PSA () {
S = ∅;
repeat until S contains s solutions {
   generate a random binary vector of length N by setting a bit to 1 with probability p
        and to 0 otherwise;
   if (x is a feasible solution) add x to sample set S;
}
M = ∅;
for i = 1 to s
   if (ith solution x in S is efficient w.r.t. M)
      add x to solution set M and remove dominated ones;
initialize temperature parameter T;
repeat until termination criterion is true {
   for l = 1 to L
      for i = 1 to s {
         x = ith solution in S;
         construct a random feasible neighbor solution y to x by (repeated) flipping of
              one or more bits in x and checking feasibility;
         if (y is efficient w.r.t. M) add y to solution set M and remove dominated ones;
         x' = element in S non-dominated by x that has minimum Hamming distance
               to x;
         if (first run or no x' found) {
            for k = 1 to K
               draw random weight w_ik;
            normalize weights w_ik to Σ_k w_ik = 1;
         }
         else {
            for k = 1 to K {
               if (x better than x' according to criterion k)
                  w_ik = a w_ik;
               else
                  w_ik = w_ik / a;
            }
            normalize weights w_ik to Σ_k w_ik = 1;
         }
```

accept y (i.e., replace x as the ith solution in S by y) with probability
$\min(1, \exp(\sum_k w_{ik}\,(u_k(x) - u_k(y))/T))$;
} $T = bT$; }}

In order to decrease the run time, we modify this basic algorithm as follows: Instead of computing the non-dominated element x' with minimum Hamming distance to the current x (which takes much computation time in the innermost loop of the algorithm), at each step we select a small random subset J of bit positions and minimize the Hamming distance only on those binary substrings defined by J. This modification results in a considerable improvement of the solution quality obtained after a pre-specified computation time.

3.3. Non-Dominated Sorting Genetic Algorithm

For our genetic algorithm implementation, we have chosen to use a technique by Deb (2001) called "Fast Elitist Non-Dominated Sorting Genetic Algorithm" (NSGA). At each generation, a combined population consisting of the parent and the children population is constructed first. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate, while dominated solutions are assigned a fitness worse than the worst fitness of any non-dominated solution. The assignment of fitness ensures that the search is directed towards the non-dominated front.

Our basic implementation of the NSGA for the problem under consideration in this paper is described below in pseudocode. Let us use the following notation (K and N are the number of criteria and the number of projects, respectively, as before):

E:	sample set of solutions – parent population,
Q:	sample set of solutions – children population,
e:	number of elements in E (fixed positive integer parameter),
q:	number of elements in Q (fixed positive integer parameter),
p:	probability of choosing a 1-bit when generating an initial solution x in E (fixed parameter between 0 and 1),
M:	solution set,
$F = (F_1, F_2, \ldots)$:	set of all non-dominated fronts,
$u_k(x)$:	objective value (to be maximized) of criterion k for solution x,
t:	iteration counter.

The algorithm is now as follows:

procedure NSGA () {
$E = \emptyset$; $Q = \emptyset$;
repeat until E contains e solutions {
generate a random binary vector of length N by setting a bit to 1 with probability p and to 0 otherwise;

```
   if (x is a feasible solution) add x to sample set E;
}
M = ∅; t = 0;
for i = 1 to e
   if (ith solution x in E is efficient w.r.t. M)
      add x to solution set M and remove all dominated ones;
repeat until termination criterion is true {
   R_t = E_t ∪ Q_t;
   F = fast-nondominated-sort(R_t);
   /* compute all non-dominated fronts F = (F_1, F_2, ...) of R_t using objective values
   u_k(x) for each objective k*/
   E_{t+1} = ∅;
        include the non-dominated fronts in the parent population E_{t+1} until the parent
        population contains e portfolios;
   Q_{t+1} = make-new-population(E_{t+1});
   /* apply one-point crossover and mutation to create a new population Q_{t+1} using
   fitness values according to F*/
   for i = 1 to q
     if (ith solution x in Q is efficient w.r.t. M)
        add x to solution set M and remove all dominated ones;
   t = t + 1;
}}
```

The first front F_1 consists of all non-dominated portfolios that are not yet dominated by any other portfolios; the second front F_2 consists of portfolios that are only dominated by those of the first front, and so on. After preliminary test and in order to decrease the run time, we modify this basic algorithm as follows: Instead of computing all non-dominated fronts of R_t, we compute only the first five fronts. It is not necessary to sort more than five fronts because due to the number of (non-linear) constraints a portfolio is either included in "upper" fronts or it is infeasible.

4. Numerical analysis

In the following section we describe computational tests which we performed in order to compare the solution quality and performance of the three approaches described in this paper. We tested the approaches on random problem instances that we generated systematically using a problem instance generator. In addition, we also compared the approaches by applying them to real-world data.

4.1. Random problems

In order to provide a fair comparison of the solution quality and performance of the described approaches we generated heterogeneous random problem instances. The de-

termining factors for portfolio selection problems are the numbers of objectives, the numbers of constraints and the numbers of projects.

4.1.1. Problem instance generator

We generated 18 random problem instances on basis of the following procedure:

A. Determine the number of projects which are the basis for possible portfolios. We generate problem instances of two differing sizes: twelve instances consisting of portfolios based on twenty projects and six instances containing portfolios based on thirty projects.

B. Determine the number of objectives. We generate instances of two objective types: for each of the two sizes, we generate instances with five objectives and instances with ten objectives.

C. Determine the number of constraints. For each size and each objective type, we generate two restriction types: problem instances with few restrictions and problem instances with many restrictions. Problem instances with few (many) restrictions contain three (nine) to six (twelve) interactions to model minimum restrictions, and another three (nine) to six (twelve) interactions to model maximum restrictions. Furthermore, these problem instances have up to three (three to five) interactions to model synergism effects.

D. Determine the resource consumption and the benefits for each objective associated with each project. The amount of resource consumption and the increase in benefit values for each of the objectives are determined randomly. However, they correlate in a way that projects with high resource consumption regularly provide a high level of benefits and vice versa.

E. Determine the quantity of the overall resources available. The quantity of the overall resources available is determined randomly and correlates to the resource consumption of one-third of the projects generated (the one-third used is chosen randomly). Furthermore, that quantity is weighted by a random number between 0.8 and 1.2.

F. Determine the synergies. Pairs of project candidates for synergy effects are selected randomly; the number of synergies depends on the constraint type (cf. step C of this procedure). The additional contribution to the objective values by combining the two selected projects into the same portfolio equals 3% of the overall benefits of all twenty (thirty, respectively) projects generated for the problem instance.

4.1.2. Parameters for the three approaches

The following section provides results for the computational tests, which were performed in order to provide an insight into how the solution quality of Pareto Ant Colony Optimization, Pareto Simulated Annealing and the Non-Dominated Sorting Genetic Algorithm develops when applying them to problem instances generated with the above procedure. In preliminary tests a comparison with a Monte Carlo Simulation and a randomized greedy approach by using the same heuristic information which was integrated

into ACO was performed (cf. Doerner et al., 2001a, 2002b). Tests showed a high superiority of ACO over the random search and provide evidence that the learning feature of ACO contributes essentially (up to 23%) to the solution quality. To provide a yardstick for a comparison of various results, we have chosen the total number PE of proposed efficient portfolios and the number E of proposed portfolios appearing in the efficient set (proven actually efficient through complete enumeration).

The parameter settings of P-ACO chosen for the computational experiments ($\alpha = 1$, $\beta = 1$, $\rho = 0.1$, $\Gamma = 10$) were taken from other applications and were pre-tested for the problem under consideration. As a result of these pre-tests the parameter q_0 is reduced from $q_0 = 0.9$ to $q_0 = 0.4$ because a higher level of diversification is desirable for our application. For the same reason $\tau_0 = 1$ appeared to be superior to the much smaller values suggested by Dorigo and Gambardella (1997) for their setting.

Some parameter settings of PSA were directly applied to our problem instances (e.g., weight modification and annealing factor) whereas others had to be adapted to the problem size (e.g., population size) according to parameter setting rules outlined in Czyzak and Jaszkiewicz (1998). For our problem instances with twenty projects we choose a population size s of 350 feasible portfolios and a size of 2000 for the problems with thirty projects. Further, the probability p of choosing a 1-bit when generating an initial solution x is set to 0.29 (because this setting turned out to be better than the value 0.19 originally suggested by Czyzak and Jaszkiewicz, 1998). The weight modification factor a is set to 1.01, and the initial temperature parameter is $T = 1$. The number of iterations on one temperature level equals 2 for the problems with twenty projects and equals 20 for the larger instances. Finally, the temperature is reduced by a factor $b = 0.9$.

As the parameters of the NSGA must also be adjusted to the problem size, the parameter settings are based on the findings by Deb (2001). A population size s of 200 feasible portfolios is used for the "twenty-project-problems," whereas the population size is set to 400 for the larger instances with thirty project proposals. The probability p of choosing a 1-bit when generating an initial solution x is set to 0.29, like in the PSA approach. The population is categorized into five efficient frontiers by the non-dominated sorting procedure. There, the fitness value of non-dominated solutions of the first front equals 1.0 and is decreased by 0.2 for the following fronts (e.g., the fitness of a solution of the second front is 0.8 while it is 0.2 for a solution of the fifth front which contains all remaining portfolios). The fitness of an infeasible solution generated by using the crossover or mutation operator is set to 0.05.

4.1.3. Numerical results for random problems

We performed all runs on a personal computer with a Pentium III-933 microprocessor, 128 MB RAM, and the operating system Windows ME; all procedures were implemented in C++.

Figure 1 shows the results computed by each approach for the random problem instances with twenty portfolios. To obtain heterogeneous instances, we combine each objective type with each constraint type and generate three problem instances with few

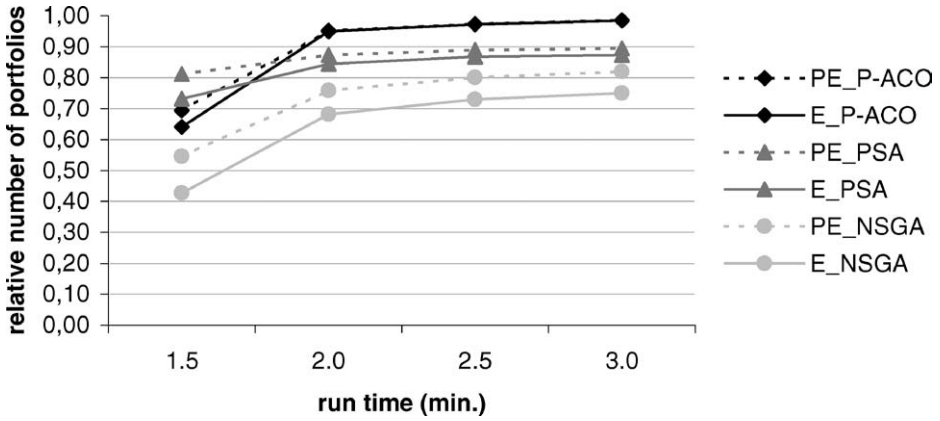

Figure 1. Numerical results of instances with 20 projects.

restrictions and another three with many restrictions for each objective type (i.e., five or ten objectives). The instances with five objectives and few constraints have 92, 58, and 232 efficient portfolios, those with many constraints have 17, 14, and 77 efficient portfolios. The instances with ten objectives and few constraints have 724, 575, and 898 efficient portfolios, while those with many constraints have 73, 169, and 973 efficient portfolios. We present values averaged over these twelve problem instances and over three computational runs. The dashed line indicates the number of proposed efficient portfolios whereas the bold line shows the actually efficient ones found. Run times varied between 1.5, 2.0, 2.5, and 3.0 CPU minutes.

PSA performs better than the other two approaches when run times are low: it suggests 12% more portfolios than P-ACO and 26% more than NSGA. The reason for the good values of PSA after short run times lies in the large initial population. In addition, the hit rate E/PE (it can be interpreted as the probability that the approach proposes a portfolio belonging to the efficient set and indicates the degree of "dilution" of a solution) is better than the hit rate of the other two approaches: 9% better than P-ACO and 30% better than the NSGA. Slightly increased run time makes P-ACO superior to PSA and NSGA. The learning feature of the P-ACO approach quickly leads to roughly 10% better results than PSA and approximately 20% better results than NSGA. Finally, the relatively small gap between the dashed and solid line referring to P-ACO shows an appealing hit rate (i.e., the ratio of potentially efficient portfolios and actually efficient ones) and may be interpreted as a low probability that P-ACO would suggest a dominated portfolio as an efficient one.

Figure 2 shows the results computed by each approach for random problem instances based on thirty portfolios. To obtain heterogeneous instances, we combined each objective type with each constraint type, and generated two instances with five objectives and few restrictions (461 and 1061 efficient portfolios), two instances with five objectives and many constraints (374 and 365 efficient portfolios), and another two instances with ten objectives and few and many restrictions (621 and 2619 efficient portfolios).

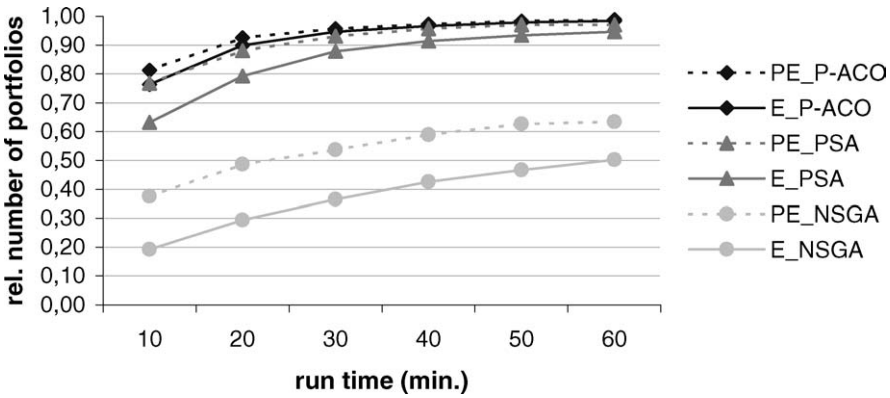

Figure 2. Numerical results of instances with 30 projects.

The run time alternatives were set to 10, 20, 30, 40, 50, and 60 minutes to give an insight into the development of the solution quality of each approach.

P-ACO shows better results than the other approaches in terms of number of proposed efficient portfolios and in terms of number of efficient portfolios found. Compared to PSA, P-ACO identifies both more and "better" portfolios from the very beginning: after 10 minutes of run time it proposes on average 81% (which are 4% more portfolios than for PSA); the ratio of the actually efficient portfolios among the proposed ones is 13% higher than the PSA's ratio after 10 minutes of run time. After 60 minutes of run time, P-ACO proposes 2% more portfolios than PSA and the ratio of actually efficient ones is 3% higher than for PSA. NSGA performs significantly worse than the two other appoaches. Again, P-ACO has the best hit rate.

4.2. Numerical results using real world data

In the following section, we present a numerical study that applies the three approaches described previously by using real world data from an R&D environment. It outlines a rather complex decision-making situation that does not permit any "intuitive" favoring of certain project combinations in advance. Our example considers thirty projects ($N = 30$), three planning periods, and two benefit categories (i.e., $K = 3 \cdot 2 = 6$). Thus, the alternative space includes 2^{30} (i.e., more than 10^9) portfolios. The projects vary substantially in both their potential benefits and the resources they require. Moreover, some projects vary significantly in their benefit values and/or resource consumption while other projects provide average values. In addition to limited resources and minimum benefit requirements, ten supplementary constraints ensure that – to provide examples for a maximum and a minimum restriction – any feasible portfolio includes at most one out of three projects pursuing the same goal, or at least two projects that help to diversify business. Finally, four interactions are used to model synergism or cannibalism between projects. After eight hours of run time complete enumeration shows that this real world problem has 980 efficient portfolios.

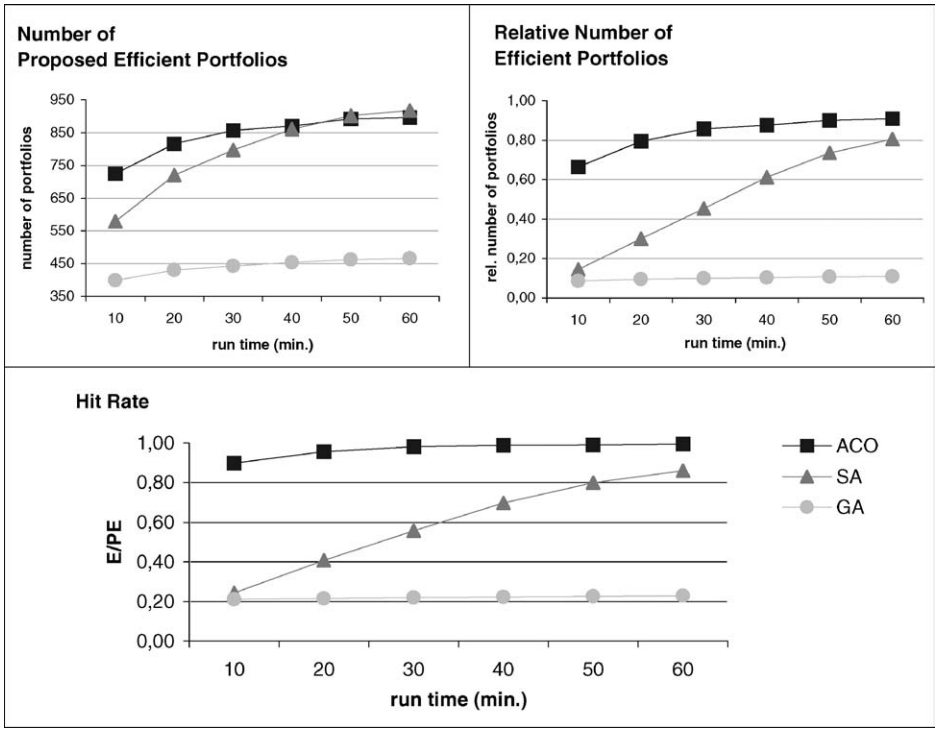

Figure 3. Numerical results with real world data.

Again, we measure the proposed efficient and the efficient portfolios that each approach found after 10, 20, 30, 40, 50, and 60 minutes of run time. Figure 3 gives an overview of several characteristics; the upper left graph shows the absolute number of proposed efficient portfolios generated by the three approaches under consideration. PSA proposes slightly more portfolios after 40 minutes of run time than P-ACO; NSGA proposes relatively few portfolios. The graph in the upper right corner measures the results of the approaches on the relative number of actually efficient portfolios. Although PSA has proposed more portfolios than P-ACO, the hit rate of P-ACO is clearly superior to PSA (see bottom graph). The reason why PSA proposes many erroneous portfolios lies in the large initializing population, which generates many random-driven, feasible solutions. The inferior results of the NSGA are based on the fact that many generated solutions become infeasible due to the large number of constraints.

As only less than 0.1% of the total search space had to be checked (i.e., on average 0.85 million portfolios compared to 1.20 million for the PSA and 0.96 million for the NSGA) to find already 92% of the efficient portfolios (after 60 minutes) this can be interpreted as a promising indicator that P-ACO will generate satisfying solutions in reasonable computation time even for problems that are too large to be enumerated completely. Apparently, SA can generate the largest number of portfolios in the given time frame, while P-ACO has the largest overhead and thus can only generate the smallest

number. Nevertheless, except for very short run times (where SA is best) P-ACO finds most PE solutions of all approaches.

5. Conclusions

Multiobjective combinatorial optimization plays a decisive role in the decision-making process on the strategic management level. Recent research activities focused on heuristic approaches for such NP-hard problems. Our paper introduces Pareto Ant Colony Optimization as a solid method to provide an efficient algorithm for this challenging problem class. We extended and enriched Ant Colony Optimization by defining multiple pheromone vectors (i.e., one pheromon vector for each objective), random objective weights for each objective, and the lifespan concept in order to apply P-ACO to multiobjective problems.

The solution quality of P-ACO is shown by providing benchmarks based on the Pareto Simulated Annealing and the Non-Dominated Sorting Genetic Algorithm approaches. To compare the solution quality and the performance, we applied the three approaches to 18 heterogeneous random problem instances and one instance using real-world data. In our experiments P-ACO turned out as the most efficient one.

Following our results, the application of P-ACO to the project portfolio selection problem under consideration has three advantages: (1) it can handle the (complex) project interactions and constraints better than the other two meta-heuristics, (2) it is robust in that it shows very good results on various problem characteristics (e.g., many or few constraints and/or objectives), and (3) heuristic information can easily be plugged into the algorithm.

Our current experience shows that for some problem instances certain efficient portfolios (found by complete enumeration) are sometimes extremely difficult to find. It could be worthwhile to analyze these "hard-to-find" portfolios in more detail to obtain ideas for further enhancement of our P-ACO approach.

Future research will focus on an enhanced efficiency of the algorithm, e.g., with an analysis of the lifespan concept in order to estimate promising lifespans for the ants. Furthermore, it will focus on real world problems with more than hundred projects. In large problems it will be important to guarantee solutions diversified over the efficient frontier; an initial attempt may consist of integrating the core idea of PSA into P-ACO to keep solutions isolated from each other.

Acknowledgments

The authors are grateful for financial support from the Austrian Science Foundation (FWF) under grant SFB #010 "Adaptive Information Systems in Economics and Management Science;" thanks are due to the anonymous referees for valuable comments on a previous version of this paper.

References

Alves, M. and J. Climaco. (2000). "An Interactive Method for 0–1 Multiobjective Problems Using Simulated Annealing and Tabu Search." *Journal of Heuristics* 6, 385–403.

Bauer, A., B. Bullnheimer, R. Hartl, and C. Strauss. (2000). "Minimizing Total Tardiness on a Single Machine Using Ant Colony Optimization." *Central European Journal of Operations Research* 8, 125–141.

Ben Abdelaziz, F., J. Chaouachi, and S. Krichen. (1999). "A Hybrid Heuristic for Multiobjective Knapsack Problems." In S. Voss, S. Martello, I. Osman, and C. Roucairol (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Boston: Kluwer, pp. 205–212.

Bullnheimer, B., R. Hartl, and C. Strauss. (1999a). "A New Rank Based Version of the Ant System: A Computational Study." *Central European Journal of Operations Research* 7, 25–38.

Bullnheimer, B., R. Hartl, and C. Strauss. (1999b). "An Improved Ant System Algorithm for the Vehicle Routing Problem." *Annals of Operations Research* 89, 319–328.

Coello, C. (2000). "An Updated Survey of GA-Based Multiobjective Optimization Techniques." *ACM Computing Surveys* 32, 109–143.

Coello, C. and A. Christiansen. (1998). "Two New GA-Based Methods for Multiobjective Optimization." *Civil Engineering Systems* 15, 207–243.

Corner, J. and J. Buchanan. (1995). "Experimental Consideration of Preference in Decision Making Under Certainty." *Journal of Multi-Criteria Decision Analysis* 4, 107–121.

Czyzak, P. and A. Jaszkiewicz. (1998). "Pareto Simulated Annealing: A Metaheuristic Technique for Multiple-Objective Combinatorial Optimization." *Journal of Multi-Criteria Decision Analysis* 7, 34–47.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester: Wiley.

Dell'Amico, M., F. Maffioli, and S. Martello. (1997). *Annotated Bibliographies in Combinatorial Optimization*. Chichester: Wiley.

Doerner, K., W.J. Gutjahr, R.F. Hartl, C. Strauss, and C. Stummer. (2001a). "Ant Colony Optimization in Multiobjective Portfolio Selection." In *Proceedings of the 4th Metaheuristics International Conference*, Porto, pp. 243–248.

Doerner, K., R.F. Hartl, and M. Reimann. (2001b). "Cooperative Ant Colonies for Optimizing Resource Allocation in Transportation." In E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, and H. Tijink (eds.), *Applications of Evolutionary Computing: EvoWorkshops 2002*. Berlin: Springer, pp. 70–79.

Doerner, K., M. Gronalt, R.F. Hartl, M. Reimann, C. Strauss, and M. Stummer. (2002a). "SavingsAnts for the Vehicle Routing Problem." In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G.R. Raidl (eds.), *Applications of Evolutionary Computing: EvoWorkshops 2002*. Berlin: Springer, pp. 11–20.

Doerner, K., W.J. Gutjahr, R.F. Hartl, C. Strauss, and C. Stummer. (2002b). "Investitionsentscheidungen bei mehrfachen Zielsetzungen und künstliche Ameisen." In P. Chamoni, R. Leisten, A. Martin, J. Minnemann, and H. Stadtler (eds.), *Operations Research Proceedings 2001: Selected Papers of the International Conference on Operations Research (OR 2001)*. Berlin: Springer, pp. 355–362.

Dorigo, M. (1992). "Optimisation, Learning, and Natural Algorithms." Ph.D. Thesis, Politecnico di Milano.

Dorigo, M. and G. Di Caro. (1999). "The Ant Colony Optimization Meta-Heuristic." In D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization*. London: McGraw-Hill, pp. 11–32.

Dorigo, M. and L. Gambardella. (1997). "Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem." *IEEE Transactions on Evolutionary Computation* 1, 53–66.

Dorigo, M., V. Maniezzo, and A. Colorni. (1996). "The Ant System: Optimization by a Colony of Cooperating Agents." *IEEE Transactions on Systems, Man and Cybernetics* 26, 29–41.

Ehrgott, M. and X. Gandibleux. (2000). "A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization." *OR Spektrum* 22, 425–460.

Finkel, R. and J. Bentley. (1974). "Quad-Trees: A Data Structure for Retrieval on Composite Keys." *Acta Informatica* 4, 1–9.

Fonseca, C. and P. Fleming. (1993). "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization." In S. Forrest (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Francisco: Morgan Kauffman, pp. 416–423.

Gambardella, L., E. Taillard, and G. Agazzi. (1999). "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows." In D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization*. London: McGraw-Hill, pp. 64–76.

Gandibleux, X., N. Mezdaoui, and A. Freville. (1997). "A Tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problems." In: R. Caballero, F. Ruiz, and R. Steuer (eds.), *Advances in Multiple Objective and Goal Programming*. Berlin: Springer, pp. 291–300.

Gutjahr, W.J. (2002). "ACO Algorithms with Guaranteed Convergence to the Optimal Solution." *Information Processing Letters* 82, 145–153.

Habenicht, W. (1983). "Quad Trees: A Datastructure for Discrete Vector Optimization Problems." In P. Hansen (ed.), *Essays and Surveys on Multiple Criteria Decision Making*. Berlin: Springer, pp. 136–145.

Hanne, T. (2000). "Global Multiobjective Optimization Using Evolutionary Algorithms." *Journal of Heuristics* 6, 347–360.

Hansen, M. (2000). "Tabu Search for Multiobjective Combinatorial Optimization: TAMOCO." *Control and Cybernetics* 29, 799–818.

Hapke, M., A. Jaszkiewicz, and R. Slowinski. (2000). "Pareto Simulated Annealing for Fuzzy Multi-Objective Combinatorial Optimization." *Journal of Heuristics* 6, 329–345.

Horn, J., N. Nafpliotis, and D. Goldberg. (1994). "A Niched Pareto Genetic Algorithm for Multiobjective Optimization." In *Proceedings of the First IEEE Conference on Evolutionary Computing*. Piscataway, pp. 82–87.

Iredi, S., D. Merkle, and M. Middendorf. (2001). "Bi-Criterion Optimization with Multi Colony Ant Algorithms." In: E. Zitzler et al. (eds.), *Evolutionary Multi-Criterion Optimization*. Berlin: Springer, pp. 359–372.

McMullen, P. (2001). "An Ant Colony Optimization Approach to Addressing a JIT Sequencing Problem with Multiple Objectives." *Artificial Intelligence in Engineering* 15, 309–317.

Murata, T. and H. Ishibuchi. (1995). "MOGA: Multi-Objective Genetic Algorithms." In *Proceedings of the Second IEEE International Conference on Evolutionary Computing*, Perth, pp. 289–294.

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

Ringuest, J. and S. Graves. (1990). "The Linear R&D Project Selection Problem: An Alternative to Net Present Value." *IEEE Transactions on Engineering Management* 37, 143–146.

Roli, A., C. Blum, and M. Dorigo. (2001). "ACO for Maximal Constraint Satisfaction Problems." In *Proceedings of the Fourth Metaheuristics International Conference*, Porto, pp. 187–191.

Schaffer, J. (1985). "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms." In J. Grefenstette (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. Hillsdale: Lawrence Erlbaum, pp. 93–100.

Serafini, P. (1994). "Simulated Annealing for Multi Objective Optimization Problems." In G. Tzeng, H. Wang, V. Wen, and P. Yu (eds.), *Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*. New York: Springer, pp. 283–292.

Srinivas, N. and K. Deb. (1994). "Multiobjective Optimization Using Non-Dominated Sorting in Genetic Algorithms." *Evolutionary Computation* 2, 221–248.

Steuer, R., L. Gardiner, and J. Gray. (1996). "A Bibliographic Survey of the Activities and International Nature of Multiple Criteria Decision Making." *Journal of Multi-Criteria Decision Analysis* 5, 195–217.

Stuetzle, T. and M. Dorigo. (1999). "ACO Algorithms for the Quadratic Assignment Problem." In D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization*. London: McGraw-Hill, pp. 33–50.

Stummer, C. (1998). *Projektauswahl im betrieblichen F&E-Management*. Wiesbaden: Gabler.

Stummer, C. and K. Heidenberger. (2001). "Interactive R&D Portfolio Selection Considering Multiple Objectives, Project Interdependencies, and Time: A Three-Phase Approach." In D. Kocaoglu and T. Anderson (eds.), *Technology Management in the Knowledge Era*. Portland: Picmet, pp. 423–428.

Sun, M. and R. Steuer. (2000). "Quad Tree Data Structures for Use in Large-Scale Discrete Multiple Criteria Problems." In Y. Shi and M. Zeleny (eds.), *New Frontiers of Decision Making for the Information Technology Era*. Singapore: World Scientific, pp. 48–71.

Ulungu, E.L. and J. Teghem. (1994). "Multi-Objective Combinatorial Optimization Problems: A Survey." *Journal of Multi-Criteria Decision Analysis* 3, 83–101.

Ulungu, E.L., J. Teghem, and P. Fortemps. (1995). "Heuristics for Multiobjective Combinatorial Optimization by Simulated Annealing." In J. Gu, C. Chen, Q. Wei, and S. Wang (eds.), *Proceedings of the Sixth National Conference on Multiple Criteria Decision Making*, Windsor, pp. 228–238.

Ulungu, E.L., J. Teghem, and C. Ost. (1998). "Efficiency of Interactive Multi-Objective Simulated Annealing through a Case Study." *Journal of the Operational Research Society* 49, 1044–1050.

White, D. (1990). "A Bibliography on the Applications of Mathematical Programming Multiple-Objective Methods." *Journal of the Operational Research Society* 41, 669–691.

Zitzler, E. and L. Thiele. (1999). "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach." *IEEE Transactions on Evolutionary Computation* 3, 257–271.