# Nature-inspired metaheuristics for multiobjective activity crashing

K.F. Doerner [a], W.J. Gutjahr [b], R.F. Hartl [a], C. Strauss [a], C. Stummer [a]

[a] *Department of Management Science, University of Vienna, Bruenner Str. 72, A-1210 Vienna, Austria*

[b] *Department of Statistics and Decision Support Systems, University of Vienna, Universitaetsstr. 5/3, A-1010 Vienna, Austria*

**Abstract**

Many project tasks and manufacturing processes consist of interdependent time-related activities that can be represented as networks. Deciding which of these subprocesses should receive extra resources to speed up the whole network (i. e., where activity crashing should be applied) usually involves the pursuit of multiple objectives amid a lack of *a priori* preference information. A common decision support approach lies in first determining efficient combinations of activity crashing measures and then pursuing an interactive exploration of this space. As it is impossible to exactly solve the underlying multiobjective combinatorial optimization problem within a reasonable computation time for real world problems, we have developed proper solution procedures based on three major (nature-inspired) metaheuristics. This paper describes these implementations, discusses their strengths, and provides results from computational experiments.

*Key words:* Heuristics, Multicriteria, Decision making, Project management

## 1 Introduction

Meeting customers' demand for short production, lead and service times while simultaneously using available resources as economically as possible is a critical challenge in competitive markets. A project manager thus regularly faces a decision problem that follows a general pattern: carrying out a project on

* Corresponding author. Tel.: +431-4277-38113; fax: -38094.
*Email address:* `karl.doerner@univie.ac.at` (K.F. Doerner).

time necessitates crashing certain activities by exploiting extra resources (e. g., using additional manpower, assigning highly-skilled personnel to specific jobs, improving machines or equipment, subcontracting certain tasks, etc. [31,36]), which, on the other hand, increases the costs of the project.

Traditionally, this time-cost problem is addressed by CPM-based approaches [25] that assume both unlimited resources and the existence of a continuous duration-cost function. However, given the discrete nature of most resources, activities can often be crashed only stepwise and/or to a limited extent or must even be left un-crashed. Such discrete versions of the activity crashing problem have been introduced by several authors in a deterministic [22,26], as well as in a stochastic context [11,15,20,21].

In practice, the problem is even more complicated because several conflicting objectives (e. g., overall project duration versus additional costs for applying activity crashing measures) usually drive the decision-making process. While the popular goal programming approaches require extensive *a priori* preference information that decision-makers often are not able or willing to provide [27], the alternative interactive decision support procedures allow the decision-makers to specify their preferences gradually and, thus, to participate in and to control the decision process [35]. Typically, such an interactive approach consists of a solution generation phase and a solution evaluation phase. While there are several appropriate approaches for the latter [34], identifying the set of the efficient alternatives for the first phase is a problem that becomes very demanding for larger problem instances. Here, metaheuristic approaches come into play because they regularly provide a favorable compromise between the quality of the (approximation of the) solution space and the required computing time; for a state-of-the-art survey cf. Ehrgott and Gandibleux [17].

We have developed proper implementations of three nature-inspired metaheuristics, namely Nondominated Sorting Genetic Algorithm II (NSGA-II), Pareto Simulated Annealing (PSA) and Pareto Ant Colony Optimization (PACO), for the activity crashing problem at hand. This paper will discuss their individual strengths and compare their performance in computational experiments based on problem instances whose network structures are taken from an on-line library and randomly generated crashing measures.

The remainder of the paper is organized as follows: Section 2 formalizes the problem of activity crashing under multiple objectives. Section 3 describes in detail the implementation of PACO, PSA and NSGA-II. The experimental design is outlined in Section 4, while results of the numerical experiments are given in Section 5. And finally, Section 6 summarizes our findings and provides an outlook to further research.

## 2 Problem description

### *2.1 Multicriteria analysis*

As outlined in the introduction, we intend to provide the decision maker with a set of feasible solutions to a multiobjective optimization problem by eliminating those solutions that are not Pareto-efficient. A solution is called *Pareto-efficient* if there is no other solution that dominates it, where the dominance relation for $K$ objective functions $f_k(x)$ $(k = 1, \ldots, K)$ is defined as follows (without loss of generality, we refer to a formulation of the problem as a minimization problem): solution $x^{(1)}$ *dominates* solution $x^{(2)}$, if

$$f_k(x^{(1)}) \leq f_k(x^{(2)}) \quad \text{for all } k,$$

and there is at least one $k$ such that

$$f_k(x^{(1)}) < f_k(x^{(2)}).$$

For abbreviation, we will sometimes use the shorter term "efficient" instead of "Pareto-efficient". We define the *Pareto frontier* as the set of objective function value vectors $(f_1(x), \ldots, f_K(x))$ corresponding to Pareto-efficient solutions $x$.

In many cases, one cannot expect to be able to determine the set of all Pareto-efficient solutions *exactly*, at least not for problem instances of a realistic magnitude. Accordingly, one needs to establish approximations of a sufficiently high quality for this set.

### *2.2 Mathematical model*

In this subsection, we provide a formal definition for the problem investigated in this paper. Let us represent the given activity structure as an activity-on-nodes network, whose nodes are numbered by $1, \ldots, n$. (For activity-on-nodes and activity-on-arcs representations of project networks, cf. Tavares [32].) Arcs are used to indicate precedence relationships between nodes (activities). The time required for activity $i$ is denoted by $d(i)$. For given values $d(i)$, the *shortest project time* $\delta$ is defined as the length of a critical path.

We assume that a finite set $M(i)$ of *measures* is assigned to each node $i$. A measure $x_i \in M(i)$ is any means that influences the duration of the activity connected with node $i$. Each measure $x_i$ is characterized by three values:

- the (modified) duration $d(i, x_i)$ of activity $i$ resulting as a consequence of the application of measure $x_i$ (since we are interested in speed-ups, we assume $d(i, x_i) \leq d(i)$), and

- two costs $c_1(i, x_i)$ and $c_2(i, x_i)$ incurred by the application of $x_i$. There are several situations where distinguishing between two different cost types is convenient. An example: Cost $c_1(i, x_i)$ is a financial cost term, e. g., money spent for paying overtime work in order to speed up activity $i$. Cost $c_2(i, x_i)$ comprises expenses that possibly cannot be immediately converted into monetary units, e. g., they can represent deteriorations in the product quality or the level of dissatisfaction among employees caused by overtime or stress.

It is always assumed that $M(i)$ contains at least the *null measure* $x_i^{(0)}$ (meaning "do nothing"), characterized by

$$d(i, x_i^{(0)}) = d(i) \quad \text{and} \quad c_1(i, x_i^{(0)}) = c_2(i, x_i^{(0)}) = 0,$$

Note that for some activities the null measure might be the only possible measure assigned, with the consequence that the corresponding activity cannot be crashed. Further, we assume that the measures $x_i^{(s)} \in M(i)$ are sorted in descending order with respect to the durations $d(i, x_i)$, i. e., if $s < s'$, then $d(i, x_i^{(s)}) \geq d(i, x_i^{(s')})$. Thus, the decision variables are the chosen measures $x_i \in M(i)$ assigned to each single node. In other words, a feasible solution is a combination of measures – or, formally speaking, an element $x$ of $M(1) \times \ldots \times M(n)$.

Based on the assumptions above, we derive three (non-decreasing) *objective functions*, assigning duration and costs to each feasible solution $x = (x_i)$:

(i) Objective 1:
$$f_1(x) = \varphi_1(\delta(x)), \tag{1}$$
where $\delta(x)$ is the shortest project time after reduction of the durations $d(i)$ to the values $d(i, x_i)$.

(ii) Objective 2:
$$f_2(x) = \varphi_2\left(\sum_{i=1}^{n} c_1(i, x_i)\right) \tag{2}$$

(iii) Objective 3:
$$f_3(x) = \varphi_3\left(\sum_{i=1}^{n} c_2(i, x_i)\right) \tag{3}$$

With $\varphi_1$, $\varphi_2$ and $\varphi_3$ chosen as identity functions, $f_1$ expresses the project duration, whereas $f_2$ and $f_3$ represent overall invested costs of the two types. The functions $\varphi_1$, $\varphi_2$ and $\varphi_3$ allow for monotonous transformations of these performance parameters, if necessary. For example, in some cases it is not the project duration itself that matters, but rather the *tardiness*, i. e., the difference between project termination time and a pre-defined due date. Similarly, for one of the cost factors, whether or by which amount a certain budget limit is exceeded may be more relevant than the value of the cost factor itself.

For convenience, we shall use a slightly simplified notation in the description of the algorithms in the following Section 3: If, for some $i$, the measure set $M(i)$ consists of the $\nu_i + 1$ elements $x^{(0)}, x^{(1)}, \ldots, x^{(\nu_i)}$, a choice $x_i = x_i^{(s)}$ from $M(i)$ can be identified by indicating the index $s$ $(0 \leq s \leq \nu_i)$ of the measure within the measure set. As we have noted, the measures $x_i^{(s)} \in M(i)$ are always kept sorted in descending order with respect to the activity durations. Therefore, larger values of $s$ correspond to smaller durations $d(i, x_i^{(s)})$, so that we can interpret $s$ as the *crashing level* of activity $i$. Choosing $s = 0$ means that the null measure is applied. Let $s_i$ be the index $s$ chosen for activity $i$. The overall solution $x$ can then be represented by the vector $x = (s_1, \ldots, s_n)$ of indices, where $0 \leq s_i \leq \nu_i$ for each $i = 1, \ldots, n$. Thus, the solution space becomes

$$\{0, \ldots, \nu_1\} \times \{0, \ldots, \nu_2\} \times \ldots \times \{0, \ldots, \nu_n\}.$$

In accordance with this simplified notation, we eventually write $c_1(i, s)$ instead of $c_1(i, x_i^{(s)})$ and $c_2(i, s)$ instead of $c_2(i, x_i^{(s)})$.

In contrast to classical activity crashing problems on continuous decision spaces that are easily solved by applying LP approaches or methods of nonlinear optimization, the problem defined above is a *hard* combinatorial problem due to its discrete and nonconvex nature. Note that even its single-objective counterpart problem (e. g., when combining the three objective functions to a weighted average) is NP-complete, since it can be reduced to a *knapsack* problem. (The proof is similar to that given by Gutjahr [20] for a related problem.)

Finally, let us note that our problem formulation is closely related to the so-called *discrete time-cost tradeoff problem* as defined by De et al. [9]. The differences are the following: (i) Instead of only a single cost criterion, we consider two independent cost criteria. (ii) We allow that for the definition of the three objective functions, both the time criterion and the two cost criteria are subjected to monotonous transformations. (iii) In [9], the solution paradigm consists in optimizing time on budget constraints or minimizing costs on time constraints, respectively; if this is done in a parametrical way, the time-cost curve is obtained. This approach is not applicable anymore for our case of three criteria, because when transiting from two to more than two criteria the "efficient frontier" is no longer a frontier but becomes a surface and it is not possible to parameterize a surface. Therefore, we provide a "true" multicriteria problem formulation (i. e., one with more than two objective functions in contrast to a bicriteria formulation) and aim to determine the set of Pareto-efficient solutions.

## 3 Solution procedures

### *3.1 Pareto Ant Colony Optimization*

PACO is an extension of the Ant Colony Optimization (ACO) metaheuristic that has evolved from the Ant System paradigm [6]. The ACO approach imitates the real-world behavior shown by ants when they search for food. Ants exchange information about food sources via pheromone, a chemical substance that they deposit on the ground as they move along. Short paths from the nest to a food source obtain a larger amount of pheromone per time unit than longer paths. As a result, an ever increasing number of ants is attracted to such routes, which in turn reinforces these pheromone trails. Artificial ants not only imitate the learning behavior described above, they also apply additional, problem-specific heuristic information. ACO has been successfully applied to various hard combinatorial optimization problems [1,3,12,16,18]; moreover, the convergence of specific ACO algorithms to optimal solutions has been shown [19]. By applying the ideas of ACO to the multiobjective case, PACO has proven to be a highly efficient heuristic technique in the field of multiobjective portfolio selection [13,14,30].

We developed two variants of PACO for the activity crashing problem at hand. PACO-LSpan, a variant that works in a manner analogous to the procedure described in Doerner et al. [13], uses a “lifespan” concept: the artificial unit called *ant* iteratively selects activities to increase crashing levels until a randomly determined lifespan has expired. The alternative variant PACO-Assign uses a different solution construction mechanism: Each ant visits each activity $i$ exactly once and assigns a crashing level $s_i$ to it; this level is not changed afterwards. We provide a description of the PACO-Assign algorithm in the following section, while the alternative PACO-LSpan is discussed in Subsection 3.1.5. In our experiments, PACO-Assign turned out to be distinctly more efficient than PACO-LSpan with respect to the chosen evaluation criteria. The only exception existed for one specific criterion (metric $Q_R$; see Section 5.1 for a discussion on the evaluation metrics) where PACO-Assign already produced superior results compared to the other tested heuristics. Therefore, we will omit presenting the numerical results for PACO-LSpan for the sake of brevity.

| | |
|---|---|
| $\Gamma$ | number of ants in each generation, |
| $\tau_{is}^k$ | pheromone information for activity $i$, crashing level $s$ and objective $k$, |
| $\eta_{is}^k$ | attractiveness information for activity $i$, crashing level $s$ and objective $k$, |
| $p_k$ | randomly generated, ant-specific weight/preference for objective $k$, |
| $\alpha, \beta$ | parameters for the random-proportional decision rule, |

| | |
|---|---|
| $R$ | number of ants that are allowed to increment pheromone values, |
| $\rho$ | persistence factor for pheromone update. |

```
procedure PACO-Assign () {
    initialize solution set $M$ by the empty set;
    create $\Gamma$ ants;
    initialize pheromone matrices $\tau^k$ $(k = 1, 2, 3)$ with constant entries;
    repeat until termination criterion is satisfied {
        for $ant = 1$ to $\Gamma$ {
            determine the weight $p_k$ for each objective $k$ at random;
            for $i = 1$ to $n$
                select a crashing level $s_i$ using formula (5);
            check efficiency of solution $x = (s_1, \ldots, s_n)$ w. r. t. $M$;
            if solution $x$ is efficient {
                store solution $x$ in $M$;
                remove solutions dominated by $x$ from $M$;
        } }
        for each objective $k$
            from the solutions just found by the $\Gamma$ ants,
            determine the $R - 1$ best solutions $x^{rk}$ for objective $k$
            $(r = 1, \ldots, R - 1)$;
        do pheromone update using formula (6);
} }
```

In an initialization phase, a number of $\Gamma$ ants are generated. In the so-called construction phase of the algorithm, each ant constructs a solution $x$ by applying a random-proportional decision rule (see Subsection 3.1.3) while using randomly generated, individual objective weights $p_k$ as well as "attractiveness information" and "pheromone information" (see Subsections 3.1.1 and 3.1.2). Both types of information are stored in matrices $\tau^k = (\tau^k_{is})$ and $\eta^k = (\eta^k_{is})$, respectively. If the identified solution $x$ is efficient with respect to the current elements of the set $M$, it is added to $M$, and elements of $M$ dominated by $x$ are removed. Finally, a pheromone update takes place; the specifics of the update are described in Subsection 3.1.4.

### *3.1.1 Attractiveness information*

In Ant Colony Optimization, "attractiveness" (also called "visibility") denotes the result of a heuristic pre-evaluation of how good a certain construction step will presumably be.

For objective $k$, the attractiveness information is stored in a matrix $\eta^k$. The rows of this matrix correspond to the activities $i$ that can be crashed, and the columns correspond to the different crashing levels $s = 0, \ldots, \nu_i$ for each activity $i$. (As a consequence, different rows may be filled up to different lengths.) The value $\eta^k_{is}$ represents the attractiveness information for crashing activity $i$ to the level $s$ with respect to objective $k$.

For the cost-related objectives 2 and 3, the attractiveness values are represented by the reciprocals of the costs of type 1 or 2, respectively, incurred by crashing activity $i$ to the level $s$:

$$\eta^k_{is} = [c_{k-1}(i,s) + 1]^{-1} \quad (k = 2, 3). \tag{4}$$

We do not use relevant attractiveness information for objective 1, i. e., we set

$$\eta^1_{is} = 1.$$

### 3.1.2 Pheromone information

"Pheromone" is the term used in Ant Colony Optimization for information that has been obtained from experience with solutions that have already been evaluated: The pheromone values assigned to construction steps that have turned out as favorable are increased, while those assigned to unsuccessful construction steps are decreased. In PACO, specific pheromone values are defined for each of the objectives.

For objective $k$, the pheromone information is stored in a matrix $\tau^k$. The rows of this matrix correspond to the activities $i$ that may be crashed, and the columns correspond to the different crashing levels $s = 0, \ldots, \nu_i$ of each activity $i$. (As in the case of the attractiveness matrix, different rows may be filled up to different lengths.) The value $\tau^k_{is}$ represents the current pheromone information for crashing activity $i$ to the level $s$ with respect to objective $k$.

### 3.1.3 Decision rule

Given the attractiveness information and the pheromone information, PACO-Assign chooses a crashing level $s$ for the current activity $i$ according to the following random-proportional rule: Value $s$ is selected with probability

$$\mathcal{P}_s(i) = \frac{\sum_{k=1}^{3} \left[p_k \cdot \tau^k_{is}\right]^\alpha \cdot \left[\eta^k_{is}\right]^\beta}{\sum_{h=0}^{\nu_i} \sum_{k=1}^{3} \left[p_k \cdot \tau^k_{ih}\right]^\alpha \cdot \left[\eta^k_{ih}\right]^\beta} \quad (s = 0, \ldots, \nu_i). \tag{5}$$

This probability distribution is parameterized by the exponents $\alpha$ and $\beta$, which determine the relative influence of pheromone and attractiveness, respectively.

### *3.1.4 Pheromone update*

The $\Gamma$ solutions produced by the ants are evaluated according to each criterion $k$ to obtain the $R-1$ best solutions for each criterion. After that, pheromone values are updated using the following rule: First, each pheromone value is reduced through multiplication with a factor $\rho$ $(0 < \rho < 1)$, the so-called *persistence factor*; then, for each $k$, pheromone entries in matrix $\tau^k$ that belong to one of the best $R-1$ solutions according to criterion $k$ are reinforced by increments depending on the rank of the solution (highest increment for the best solution, etc.). Technically, the update rule is as follows: Consider a fixed criterion $k$ and the corresponding pheromone matrix $\tau^k$; let $x^{rk}$ denote the solution ranked as the $r$-th best with respect to objective $k$; then, for each $i$ and each $s$,

$$\tau_{is}^k = \rho\tau_{is}^k + \sum_{r=1}^{R-1} \Delta\tau_{is}^{rk} \tag{6}$$

with

$$\Delta\tau_{is}^{rk} = \begin{cases} R-r, & \text{if } x_i^{rk} = s, \\ 0, & \text{otherwise.} \end{cases}$$

This update is performed for each $k$.

Tests have shown that setting $R = 3$ leads to good results. However, the performance is not very sensitive with respect to this parameter.

### *3.1.5 PACO-LSpan*

In this variant, for each ant, a lifespan $L$ is drawn from a uniform distribution with a minimum value of 0 and a maximum value of $L_{max}$, where $L_{max} = \sum_{i=1}^{n} \nu_i$. Each ant starts with initial solution $(0, \ldots, 0)$, i. e., a solution containing only null measures. Instead of going through the activities in the pre-defined order (as in PACO-Assign) and deciding for each activity at which level it should be crashed (including level 0 as an option), the variant PACO-LSpan allows each ant to decide at each construction step to which activity $i$ it will move for the next crashing action. The crashing action consists in an increment of the current value for variable $s_i$ by one unit. The process is repeated until the number of construction steps has reached the predefined lifespan $L$ of the ant.

### *3.2 Pareto Simulated Annealing*

In order to compare the implemented PACO results with those of a multiobjective simulated annealing variant, we apply a technique introduced by Czyzak and Jaszkiewicz [8], called "Pareto Simulated Annealing" (PSA). An extension of the multiobjective simulated annealing algorithms proposed by Serafini [29], and by Ulungu et al. [33], PSA uses a population-based simulated annealing approach that keeps a sample $S$ of solutions, the so-called *generating sample*, in each iteration. The aim is (a) to move $S$ towards the set of Pareto-efficient solutions, while (b) favoring dispersion among the elements of $S$. The last goal is achieved as follows: For a given $x \in S$, the weights of the objectives are changed in such a way that $x$ is given a tendency of moving away from its closest neighbor $x'$ in $S$. This is done by increasing the weights of objectives on which $x$ is better than $x'$ and decreasing the weights of the other objectives.

| | |
|---|---|
| $S$ | generating sample set of current feasible solutions, |
| $\sigma$ | number of elements in $S$, |
| $M$ | solution set |
| $w_{jk}$ | weight of criterion $k$ for the $j$-th element of sample set $S$, |
| $a$ | weight modification factor, |
| $T$ | temperature parameter for simulated annealing, |
| $L$ | number of iterations on each temperature level, |
| $b$ | temperature reduction factor (annealing factor), |
| $N_m$ | maximum number of level modifications. |

Part of each simulated annealing algorithm involves a procedure for the transition to a random *neighbor solution*. We compute a neighbor as follows: For a given solution $x = (s_1, \ldots, s_n)$, we draw a number $\kappa$ of crashing level modifications uniformly from $\{1, \ldots, N_m\}$. Now, in a loop with $\kappa$ iterations, a random position $i$ is selected, and the crashing level $s_i$ is either increased by one (if possible), or decreased by one (if possible). The decision on "increase" or "decrease" is made with probability 0.5 for each alternative. Note that by pursuing this approach we do not restrict ourselves to single local changes (case $\kappa = 1$), but allow changes in several components.

A pseudo-code formulation of our implementation of PSA is given below. As in the previous subsections, $K$ and $n$ are the number of objective functions and the number of activities, respectively, $f_k$ is the $k$-th objective function, and $x = (s_1, \ldots, s_n)$ is a feasible solution.

```
procedure PSA () {
  initialize $S$ with the empty set;
  repeat until $S$ contains $\sigma$ solutions {
    generate a random vector $x = (s_1, \ldots, s_n)$ of length $n$ by
      selecting $s_i$ randomly from $\{0, \ldots, \nu_i\}$ $(i = 1, \ldots, n)$;
    add $x$ to sample set $S$;
  }
  initialize $M$ with the empty set;
  for $j = 1$ to $\sigma$
    if ($j$-th solution $x$ in $S$ is efficient w. r. t. $M$)
      add $x$ to solution set $M$ and remove dominated solutions;
  initialize temperature parameter $T$;
  repeat until termination criterion is satisfied {
    for $l = 1$ to $L$
      for $j = 1$ to $\sigma$ {
        $x$ = $j$-th solution in $S$;
        choose a random neighbor solution $y$ to $x$;
        if ($y$ is efficient w. r. t. $M$), add $y$ to solution set $M$
          and remove dominated solutions from $M$;
        $x'$ = element in $S$ non-dominated by $x$ that has minimum
          number of different components to $x$;
        if (first run or no $x'$ found) {
          for $k = 1$ to 3
            draw random weight $w_{jk}$;
          normalize weights $w_{jk}$ to $\sum_k w_{jk} = 1$;
        }
        else {
          for $k = 1$ to 3 {
            if ($x$ better than $x'$ according to criterion $k$)
              $w_{jk} = a w_{jk}$;
            else
              $w_{jk} = w_{jk}/a$;
          }
          normalize weights $w_{jk}$ to $\sum_k w_{jk} = 1$;
        }
        accept $y$ (i. e., replace $x$ as the $j$-th solution in $S$ by $y$) with
            probability $\min\left(1, \exp\left(\sum_k w_{jk}\left(f_k(x) - f_k(y)\right)/T\right)\right)$;
      }
      set $T = bT$;
} }
```

## *3.3 Nondominated Sorting Genetic Algorithm II*

There are several variants of multiobjective genetic algorithm techniques; for a comprehensive survey, we refer the reader to Coello [4]. For our problem context, we implemented the *Nondominated Sorting Genetic Algorithm II* (NSGA-II) by Deb [10].

The basic idea of NSGA-II is to create a sequence of generations of solutions, to sort the current generation into nondomination levels, to perform a finer evaluation of solutions from the same nondomination level by using a crowding distance measure, and to combine subsequent parent-offspring sets to candidate sets for the selection of the next generation. We use the following notation to describe the algorithm:

| | |
|---|---|
| $S_P$ | sample set of solutions – parent population, |
| $S_Q$ | sample set of solutions – offspring population, |
| $\sigma$ | number of elements in $S_P$ and in $S_Q$ (fixed integer), |
| $M$ | solution set (i. e., set of all proposed efficient solutions), |
| $\mu$ | mutation probability (fixed parameter between 0 and 1). |

NSGA-II uses an auxiliary procedure fast-nondominated-sort($S_P$) that partitions a given set $S_P$ of solutions into sets $\mathcal{F}_1$, $\mathcal{F}_2$ etc., where $\mathcal{F}_l$ denotes the $l$-th nondomination level of $S_P$. (The first nondomination level consists of the nondominated elements in $S_P$, the second nondomination level consists of the nondominated elements after removing the elements of $\mathcal{F}_1$ from $S_P$, etc.). For the details of fast-nondominated-sort(), we refer the reader to Deb [10].

Furthermore, we apply an auxiliary procedure called crowding-distance-assignment($S_P$), which assigns to each element $x \in S_P$ a measure indicating how close other solutions of $S_P$ lie to $x$ in objective space. In our implementation, we used the Euclidean distance to the nearest neighbor point in objective space to quantify this measure.

An order relation $\prec$ is defined based on the nondomination level numbers rank($x$) and the crowding distance measures distance($x$): For two solutions, $x \prec y$ if rank($x$) $<$ rank($y$), or if rank($x$) $=$ rank($y$) and distance($x$) $>$ distance($y$).

As in ordinary genetic algorithms, for each parent population $S_P$, an offspring population $S_Q$ is computed by the operators selection, mutation and crossover. The crowded-comparison operator $\prec$ is used for fitness evaluation in the selection step. For the mutation step, we use the same procedure as applied in

PSA for the transition to a neighbor: If a solution $x$ is to be mutated, it is replaced by a random neighbor solution as described in Section 3.2. Crossover is done in a straightforward manner, namely as two-point crossover of strings $x$ and $y$.

```
procedure NSGA-II () {
  initialize both $S_P$ and $S_Q$ with the empty set;
  repeat until $S_P$ contains $\sigma$ solutions {
    generate a random vector $x = (s_1, \ldots, s_n)$ of length $n$ by
      selecting $s_i$ randomly from $\{0, \ldots, \nu_i\}$ $(i = 1, \ldots, n)$;
    add $x$ to $S_P$;
  }
  initialize $M$ with the empty set;
  for $j = 1$ to $\sigma$
    if ($j$-th solution $x$ in $S_P$ is efficient w. r. t. $M$)
      add $x$ to solution set $M$ and remove dominated solutions;
  make a new population $S_Q$ from $S_P$;
  repeat until termination criterion is satisfied {
    combine parent and offspring generation: $S_R = S_P \cup S_Q$;
    apply fast-nondominated-sort to $S_R$ to obtain nondominated
      levels $\mathcal{F}_1, \mathcal{F}_2, \ldots$;
    initialize $S'_P$ with the empty set;
    $l := 1$;
    while $(|S'_P| + |\mathcal{F}_l| \leq \sigma)$ {
      do crowding-distance-assignment($\mathcal{F}_l$);
      $S'_P := S'_P \cup \mathcal{F}_l$;
      $l := l + 1$; }
    sort $\mathcal{F}_i$ according to order $\prec$;
    add the first $\sigma - |S'_P|$ elements of $\mathcal{F}_l$ to $S'_P$;
    make a new population $S'_Q$ from $S'_P$;
    set $S_P := S'_P$ and $S_Q := S'_Q$;
    for $j = 1$ to $\sigma$
      if ($j$-th solution $x$ in $S_Q$ is efficient w. r. t. $M$)
        add $x$ to solution set $M$ and remove dominated solutions;
} }
```

## 4 Experimental design

### *4.1 Network structure of problem instances*

As a means of ensuring a fair comparison in terms of solution quality and performance, we generated random problem instances based on various complex network-structures taken from literature. Figure 1 illustrates such a complex network with 43 activities and 60 arcs taken from [25]. We refer to this network as to "Moder1" in the result description. Figure 2 depicts a network-structure with 73 nodes and 113 arcs taken from [25] as well. This network will be referred to as "Moder2" in the result description. Both networks have been enriched by crashing measures that may be characterized by values of (monetary) cost and time.

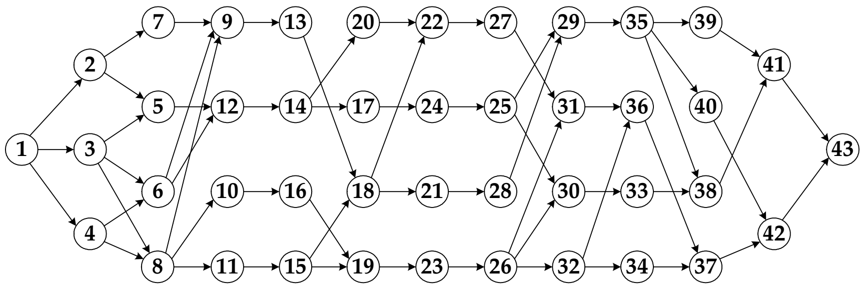

Fig. 1. Structure of a network with 43 activities.

In addition, computational experiments have been run for two groups of even larger instances taken from the on-line library provided by Kolisch and Sprecher [24]. Each of these two groups comprise five networks containing 90 activities ("Kolisch1" to "Kolisch5" in the result description) and 120 activities ("Kolisch11" to "Kolisch15" in the result description), respectively.

### *4.2 Cost and time structure of problem instances*

We randomly generated cost and time values for the crashing activities for the above-mentioned network structures:

A. *Time structure.* The duration $d(i)$ of an activity $i$ follows a uniform distribution on the interval $[1, 10]$. We define that crashing reduces the activity duration $d(i)$ by one time unit in each single crashing step, but always keep the remaining duration positive: $d(i, x^{(s)}) = \max(d(i) - s, 1)$. The functions $\varphi_1, \ldots, \varphi_3$ have been chosen as identity functions.

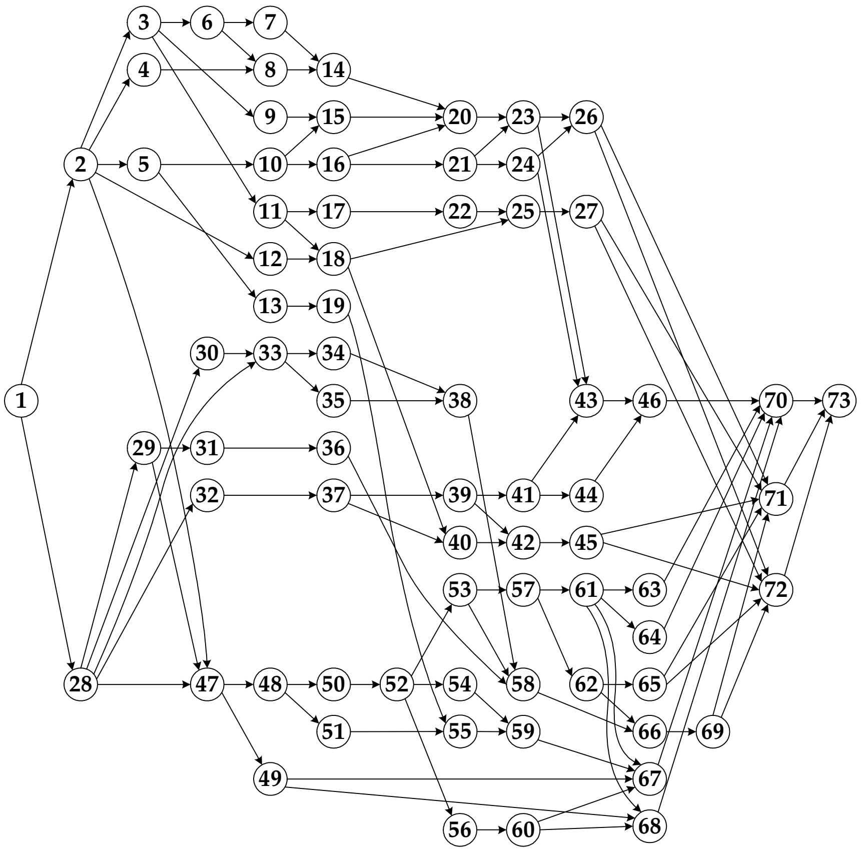

Fig. 2. Structure of a network with 73 activities.

B. *Number of crashing alternatives for each activity.* Applying a truncated geometric distribution with parameter $p = 0.8$ determines the number $\nu_i$ of measures (different from the null measure) for each activity $i$. The distribution is truncated in such a way that the maximum value of $\nu_i$ is equal to 10.

C. *Cost structure.* Each crashing measure $x_i$ triggers two cost values $c_1(i, x_i)$ and $c_2(i, x_i)$ representing the two different cost types. In our problem instance generation procedure, both cost terms depend on the achieved reduction $s$ of the activity duration and are computed as

$$c_k(i, x_i^{(s)}) = \sum_{j=1}^{s} \xi_{ijk} \quad (k = 1, 2), \tag{7}$$

for $1 \leq s \leq \nu_i$ and $c_k(i, x_i^{(0)}) = 0$, where each random variable $\xi_{ijk}$ is drawn independently from an uniform distribution on the interval $[0.5, 1.5]$. Thus, $\xi_{ijk}$ represents marginal costs of type $k$ required for further reducing the duration of activity $i$ $(i = 1, \ldots, n)$ given that it has already been crashed by $j - 1$ measures $(j = 1, \ldots, \nu_i)$.

### *4.3 Parameters for the three approaches*

In this subsection we describe the parameter values chosen for the test runs of Pareto Ant Colony Optimization, Pareto Simulated Annealing and the Non-dominated Sorting Genetic Algorithm II at our test instances (for an overview see Table 2).

The parameter settings of the PACO chosen for the computational experiments ($\alpha = 1$, $\beta = 1$, $\rho = 0.9$, $\Gamma = 20$) were taken from other applications in which they have proven to be advantageous [3], and were pre-tested for the problem under consideration. The entries in the pheromone matrix were initialized with the value 1.

For PSA, some parameter settings described by Czyzak and Jaszkiewicz [8] were directly applied to our problem instances (e. g., weight modification and annealing factor), whereas others had to be adapted to the size of our problem instances: We chose a size $\sigma$ of 200 for the sample set. The weight modification factor $a$ was set to 1.05, and $L$ was set to 2, such that 400 trials on neighbor solutions were made on each temperature level. Finally, the temperature is reduced by a factor $b = 0.95$. The maximum number of level modifications $N_m$ was set to the value 16.

Also the standard parameter values for NSGA-II, as described in the literature, had to be adjusted to the size of our problem instances. A population size $\sigma$ of 200 feasible portfolios was used for the entire set of instances. Furthermore, we applied a mutation rate $\mu$ of 0.1, as tests have shown that this rate is adequate.

Table 2
Parameter settings for numerical experiments.

| PACO | PSA | NSGA-II |
|---|---|---|
| $\Gamma = 20$ | $\sigma = 200$ | $\sigma = 200$ |
| $\alpha = \beta = 1$ | $a = 1.05$ | $\mu = 0.1$ |
| $R = 3$ | $L = 2$ | |
| $\rho = 0.9$ | $b = 0.95$ | |
| | $N_m = 16$ | |

## 5 Computational results

We performed all runs on a personal computer with a Pentium IV-2.4 GHz microprocessor, 256 MB RAM, and the operating system Windows 2000; all procedures were implemented in C++ and compiled using the Borland compiler v. 6. All of the algorithms were given the same runtime of 60 CPU seconds. Five runs with different random numbers were performed for each algorithm and each test instance.

### *5.1 Evaluation metrics*

Numerous metrics have been proposed for evaluating multiobjective metaheuristics [5,23,38]. While many pairs of such metrics produce correlated results, there are also metrics that are capable of measuring the achievement of conflicting goals. Therefore, the evaluation of multiobjective metaheuristics can itself be regarded as a multiobjective decision analysis problem.

Evaluation metrics of multiobjective metaheuristics that follow the Pareto approach can be categorized according to some of the principal aims they pursue. There seem to be at least three such aims that are complementary to each other: (1) a good approximation of the set of Pareto-efficient solutions (i. e., of the Pareto frontier), (2) a fairly equal distribution of the proposed solutions (either in solution space or in objective space), and (3) coverage of a broad range of the region of the Pareto frontier. For the results of our experimental tests, we have chosen one specific metric from each of these three groups, and added a fourth metric that allows a pairwise comparison of metaheuristics also in cases where the Pareto frontier is unknown.

All applied metrics have been computed based on *normalized* instead of absolute objective function values. Normalization is performed by multiplying each objective function value $f_k(x)$ by a *range equalization factor*

$$\pi_k = 1 \,/\, \mathrm{range}\,(k),$$

where range($k$) measures the range of the objective function values for $f_k(x)$ as the difference of an upper and a lower bound of these values [23]. In our case, we have set

$$\mathrm{range}\,(1) = \sum_i d(i),$$

where the sum is over the activities $i$ of a critical path for the considered test instance. This is the difference of $f_1(x)$ between the worst case and an "utopia point", where $\delta$ has been crashed to zero. Similarly, we have set

$$\text{range}\,(k) = \sum_{i=1}^{n} c_{k-1}(i, x_i^{(\nu_i)}) \quad (k = 2, 3),$$

which represents the difference between the maximum possible cost and cost zero for each of the two cost types.

We denote the normalized objective function value by using $\bar{f}_k(x)$. The set of normalized solution points, $\{(\bar{f}_1(x), \ldots, \bar{f}_K(x)) \,|\, x \in M\} \subseteq \mathbb{R}^K$, where $M$ is the set of solutions proposed by a given metaheuristic, is denoted by symbols $A$ or $B$.

*Metric 1:* For measuring the degree by which the set $A$ of proposed points (more precisely: normalized solution value vectors) approximates the Pareto frontier, we have chosen the metric $Q_4(A)$ in the notation of Jaszkiewicz [23]; this metric was originally introduced by Czyzak and Jaszkiewicz [7]. For determining $Q_4(A)$, one must rely on a reference set $RS$ that is identical to or at least a good approximation of the Pareto frontier $PF$. We computed $RS$ for a given test instance as the set obtained from the union of points proposed by any of the metaheuristics in any of the random runs by removing dominated points. $Q_4(A)$ is given by

$$Q_4(A) = \frac{1}{|RS|} \sum_{r \in RS} \min_{z \in A} D(z, r),$$

where $D(.,.)$ is a distance measure in objective space, for which we took the Euclidean distance.

*Metric 2:* For measuring how uniformly the points in $A$ are distributed in objective space, we have chosen the *Spacing metric* $Q_5(A)$ by Schott [28]:

$$Q_5(A) = \sqrt{\frac{1}{|A| - 1} \sum_{z \in A} (\bar{D} - D(z))^2},$$

where

$$D(z) = \min_{z' \in A} \sum_{k=1}^{K} |z_k - z'_k|,$$

and $\bar{D}$ is the mean of all values $D(z)$.

*Metric 3:* For measuring how well the whole possible range of the Pareto frontier is covered by the points in $A$, we have applied the following simple measure:

$$Q_R(A) = \frac{1}{K} \sum_{k=1}^{K} (R_k^{max}(A) - R_k^{min}(A)),$$

where

$$R_k^{max}(A) = \max\{\bar{f}_k(x) \,|\, x \in A\}, \quad R_k^{min}(A) = \min\{\bar{f}_k(x) \,|\, x \in A\}.$$

*Metric 4:* For a mutual comparison of the sets $A$ and $B$ produced by two metaheuristics, we applied a metric introduced by Zitzler and Thiele [37]; following Jaszkiewicz [23], we abbreviate it by $Q_6(A, B)$.

$$Q_6(A, B) = \frac{|\{z^2 \in B \,|\, \exists z^1 \in A : z^1 \succeq z^2\}|}{|B|},$$

where $z^1 \succeq z^2$ if $z^1$ dominates $z^2$ or $z^1 = z^2$. $Q_6(A, B) = 1$ indicates that every point in $B$ is dominated by or equal to some point in $A$, whereas $Q_6(A, B) = 0$ indicates that no point in $B$ is dominated or equal to some point in $A$.

From a project manager's point of view the decision on which of the above metrics to focus on primarily depends on his/her current requirements. If the manager needs to ensure that proposed efficient solutions in general are identical or close to one of the solutions out of a reliable reference set, metric $Q_4$ has to be favored. If the uniform distribution in objective space is of importance (e. g., for ensuring availability of proper, but not necessarily true efficient, decision alternatives for each potential combination of (implicit) objective weights), a project manager may opt to primarily rely on metric $Q_5$. Otherwise, if all what matters is the determination of the correct broad ranges of the given objective functions, metric $Q_R$ may be the best choice. Metric $Q_6$, finally, is probably more of interest for academics evaluating the performances of different metaheuristic procedures. However, note that prioritizing these metrics itself may constitute a multiobjective decision problem as stated above.

## *5.2 Analysis*

Table 3 shows the results obtained for the metric $Q_4$, which measures how well the Pareto frontier is approximated. Each entry is an average of $Q_4(A)$ over the five test runs for each heuristic and each test instance; smaller values of $Q_4$ correspond to better results. One can observe that PSA produced the best results and NSGA-II produced the poorest ones for the smaller "Moder" test instances. This effect changed for the larger "Kolisch" test instances:

Table 3
Evaluation on Metric $Q_4$.

| $Q4$ | Kolisch1 | Kolisch2 | Kolisch3 | Kolisch4 | Kolisch5 |
|---|---|---|---|---|---|
| PACO | 0.039 | 0.040 | 0.043 | 0.040 | 0.047 |
| PSA | 0.174 | 0.202 | 0.139 | 0.181 | 0.163 |
| NSGA | 0.051 | 0.056 | 0.051 | 0.048 | 0.069 |
| $Q4$ | Kolisch11 | Kolisch12 | Kolisch13 | Kolisch14 | Kolisch15 |
| PACO | 0.040 | 0.039 | 0.022 | 0.049 | 0.035 |
| PSA | 0.211 | 0.190 | 0.229 | 0.178 | 0.170 |
| NSGA | 0.127 | 0.097 | 0.101 | 0.087 | 0.050 |
| $Q4$ | Moder1 | Moder2 | | | |
| PACO | 0.699 | 0.175 | | | |
| PSA | 0.467 | 0.112 | | | |
| NSGA | 0.737 | 0.192 | | | |

Here PACO usually produced the best results, the results of NSGA-II were only marginally worse (and, in three cases, even slightly better), and PSA performed the worst. The fact that the PSA outcomes drop from first to last place when passing from the smaller to the larger test instances indicates that PSA would possibly have required a larger runtime on the larger instances.

Table 4 shows the results obtained for the Spacing-metric $Q_5$. Again, the entries represent the averages for the five test runs; smaller values of $Q_5$ correspond to better results. NSGA-II produces the best results in all cases except one (in which PSA performs best), while PACO turns out to be distinctly inferior to the two other approaches. This can easily be explained by the fact that both PSA and NSGA-II use specific mechanisms to distribute the solutions equally over the Pareto frontier approximation, whereas PACO does not apply such a mechanism.

Table 5 shows the results obtained for the metric $Q_R$ measuring the width of the range covered by the proposed solutions; once again, the entries represent averages for the five test runs. In this case, larger values of $Q_R$ correspond to better results. For the smaller "Moder" instances, PSA produced the best results, followed by PACO. However, for the larger "Kolisch" instances, PACO proved to be best, followed by PSA. This ranking is consistent over the 10 test

Table 4
Evaluation on Metric $Q_5$.

| $Q5$ | Kolisch1 | Kolisch2 | Kolisch3 | Kolisch4 | Kolisch5 |
|---|---|---|---|---|---|
| PACO | 0.0033 | 0.0024 | 0.0022 | 0.0016 | 0.0019 |
| PSA | 0.0011 | 0.0027 | 0.0017 | 0.0009 | 0.0017 |
| NSGA | 0.0008 | 0.0008 | 0.0015 | 0.0004 | 0.0003 |
| $Q5$ | Kolisch11 | Kolisch12 | Kolisch13 | Kolisch14 | Kolisch15 |
| PACO | 0.0016 | 0.0012 | 0.0010 | 0.0024 | 0.0015 |
| PSA | 0.0037 | 0.0007 | 0.0048 | 0.0017 | 0.0015 |
| NSGA | 0.0024 | 0.0006 | 0.0010 | 0.0013 | 0.0003 |
| $Q5$ | Moder1 | Moder2 | | | |
| PACO | 0.0033 | 0.0023 | | | |
| PSA | 0.0027 | 0.0010 | | | |
| NSGA | 0.0016 | 0.0016 | | | |

Table 5
Evaluation on Metric $Q_R$.

| $Q_R$ | Kolisch1 | Kolisch2 | Kolisch3 | Kolisch4 | Kolisch5 |
|---|---|---|---|---|---|
| PACO | 0.431 | 0.455 | 0.455 | 0.429 | 0.405 |
| PSA | 0.293 | 0.345 | 0.349 | 0.313 | 0.349 |
| NSGA | 0.288 | 0.265 | 0.272 | 0.259 | 0.248 |
| $Q_R$ | Kolisch11 | Kolisch12 | Kolisch13 | Kolisch14 | Kolisch15 |
| PACO | 0.417 | 0.567 | 0.412 | 0.593 | 0.432 |
| PSA | 0.312 | 0.412 | 0.345 | 0.445 | 0.332 |
| NSGA | 0.182 | 0.290 | 0.118 | 0.301 | 0.231 |
| $Q_R$ | Moder1 | Moder2 | | | |
| PACO | 0.209 | 0.254 | | | |
| PSA | 0.414 | 0.365 | | | |
| NSGA | 0.131 | 0.242 | | | |

instances.

Table 6
Number of proposed efficient solutions.

| # PE | Kolisch1 | Kolisch2 | Kolisch3 | Kolisch4 | Kolisch5 |
|---|---|---|---|---|---|
| PACO | 19.0 | 25.0 | 20.4 | 25.8 | 17.6 |
| PSA | 16.8 | 15.6 | 17.4 | 16.2 | 17.2 |
| NSGA | 13.4 | 17.4 | 12.8 | 17.6 | 11.6 |
| # PE | Kolisch11 | Kolisch12 | Kolisch13 | Kolisch14 | Kolisch15 |
| PACO | 22.0 | 28.0 | 22.0 | 24.2 | 25.8 |
| PSA | 13.4 | 18.2 | 10.8 | 16.8 | 15.0 |
| NSGA | 9.8 | 18.8 | 6.6 | 15.0 | 15.4 |
| # PE | Moder1 | Moder2 | | | |
| PACO | 11.4 | 11.2 | | | |
| PSA | 25.0 | 14.8 | | | |
| NSGA | 8.4 | 13.4 | | | |

For an explanation of why PSA outperforms PACO in the smaller instances, whereas this effect is reversed for the larger instances, we might consider Table 6 presenting the average number of solutions proposed as efficient by each heuristic. It makes clear that PSA proposes a larger number of solutions than PACO in the smaller instances, but fewer solutions than PACO in the larger instances.

Table 7 shows pairwise comparisons of the three heuristics measured by $Q_6$. In this evaluation, the union of the five solution sets produced by the five runs, minus dominated elements, has been "sent into the competition" for each test instance and each heuristic. We can see that in these comparisons, NSGA-II performs best (outperforming PACO in all but one case), followed by PACO. Although PACO usually achieves a better approximation to the Pareto-frontier than NSGA-II, the solutions delivered by NSGA-II are able to cover (i. e., to dominate or to equalize) a larger percentage of those delivered by PACO than *vice versa*. This seeming contradiction is also explained by the observation that PACO usually produces the largest number of solutions, which facilitates an approximation of the points on the Pareto frontier, but only gives a comparably small advantage in dominating solutions produced by other approaches.

Table 7
Evaluation on Metric $Q_6$.

| Kolisch1 | PACO | PSA | NSGA | Kolisch2 | PACO | PSA | NSGA |
|---|---|---|---|---|---|---|---|
| PACO | . | 0.845 | 0.436 | PACO | . | 0.675 | 0.261 |
| PSA | 0.107 | . | 0.001 | PSA | 0.407 | . | 0.083 |
| NSGA | 0.903 | 0.893 | . | NSGA | 0.958 | 0.870 | . |
| **Kolisch3** | **PACO** | **PSA** | **NSGA** | **Kolisch4** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.988 | 0.266 | PACO | . | 0.926 | 0.161 |
| PSA | 0.208 | . | 0.000 | PSA | 0.289 | . | 0.069 |
| NSGA | 0.980 | 1.000 | . | NSGA | 0.965 | 1.000 | . |
| **Kolisch5** | **PACO** | **PSA** | **NSGA** | **Kolisch11** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.988 | 0.175 | PACO | . | 1.000 | 0.196 |
| PSA | 0.024 | . | 0.000 | PSA | 0.392 | . | 0.000 |
| NSGA | 1.000 | 1.000 | . | NSGA | 0.980 | 1.000 | . |
| **Kolisch12** | **PACO** | **PSA** | **NSGA** | **Kolisch13** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.976 | 0.345 | PACO | . | 0.961 | 0.677 |
| PSA | 0.346 | . | 0.023 | PSA | 0.441 | . | 0.000 |
| NSGA | 0.954 | 1.000 | . | NSGA | 0.270 | 0.828 | . |
| **Kolisch14** | **PACO** | **PSA** | **NSGA** | **Kolisch15** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.923 | 0.014 | PACO | . | 0.574 | 0.074 |
| PSA | 0.00 | . | 0.000 | PSA | 0.508 | . | 0.103 |
| NSGA | 0.99 | 1.000 | . | NSGA | 1.000 | 0.869 | . |
| **Moder1** | **PACO** | **PSA** | **NSGA** | **Moder2** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.623 | 0.000 | PACO | . | 0.595 | 0.194 |
| PSA | 0.906 | . | 0.013 | PSA | 0.574 | . | 0.000 |
| NSGA | 1.000 | 0.951 | . | NSGA | 0.852 | 1.000 | . |

To investigate the influence of computation time on the relative performance of the three approaches in relation to each other, we repeated the $Q_6$ evaluations at the instance “Kolisch1” for larger runtimes of 2, 5 and 10 minutes per run. The results are shown in Table 8. Note that the following parameter values have been adapted to the runtime: If the runtime was multiplied by a given factor, in the case of PACO, the value $1 - \rho$ was divided by this factor, and in the case of PSA, the number $L$ of iterations at a specific temperature level

Table 8
Development of $Q_6$ with increasing computation time.

| Kolisch1 / 2 min | PACO | PSA | NSGA |
|---|---|---|---|
| PACO | . | 0.82 | 0.39 |
| PSA | 0.01 | . | 0.00 |
| NSGA | 0.52 | 1.00 | . |
| **Kolisch1 / 5 min** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.62 | 0.32 |
| PSA | 0.15 | . | 0.08 |
| NSGA | 0.59 | 0.82 | . |
| **Kolisch1 / 10 min** | **PACO** | **PSA** | **NSGA** |
| PACO | . | 0.60 | 0.02 |
| PSA | 0.47 | . | 0.00 |
| NSGA | 0.98 | 1.00 | . |

was multiplied by this factor. NSGA-II uses no parameter that require a direct adaptation to the runtime; increasing the number of generations is sufficient.

One can see that the advantage of PACO over PSA decreases with increasing computation time, confirming the conjecture above that PSA requires more runtime to unfold its capability. On the other hand, the advantage of NSGA-II over PACO *and* PSA becomes overwhelming for longer computation times, at least in the measure of metric $Q_6$. It should be emphasized that also computation time must be considered as a quality criterion for a heuristic, especially if the heuristic is intended for incorporation into an *interactive* decision support system in which repeated optimization runs with changed, refined or extended parameter assignments should be made feasible.

## 6 Conclusions

Project managers in real-world applications are regularly confronted with conflicting requirements: Early due dates necessitate speeding up certain activities by crashing measures (using additional manpower, assigning highly-skilled personnel to specific jobs, improving machines or equipment, subcontracting of certain tasks, etc.), which, on the other hand, increase the costs of the project. Selecting the “right” processes and applying the “right” crashing measures under time and cost considerations is a key issue. One alternative for solving problems with multiple objectives involves an interactive approach that consists of an initial solution generation phase and a succeeding solution evaluation phase. In this article, three nature-inspired metaheuristics (namely, Nondominated Sorting Genetic Algorithm II, Pareto Simulated Annealing, and Pareto Ant Colony Optimization) were investigated as potential tools for identifying the set of efficient alternatives in the first phase.

We designed and implemented variants of the three approaches for the solution of the considered activity crashing problem. To compare the solution quality, we applied the three metaheuristics to 12 heterogeneous random problem instances; their network structure was taken from literature, while crashing alternatives and associated costs were generated randomly. For the evaluation, four different metrics were used, three of which are commonly applied evaluation metrics taken from the multiobjective optimization literature.

In short, we establish the following: Under the condition of a given (rather scarce) computation time, Pareto Ant Colony Optimization proved to be slightly superior to the other approaches in a metric measuring approximation of the Pareto frontier and in a metric measuring width of the covered range in objective space. Conversely, the Nondominated Sorting Genetic Algorithm II and Pareto Simulated Annealing were better in a metric measuring equal distribution of the proposed solutions. Moreover, the Nondominated Sorting Genetic Algorithm II outperformed the two other approaches in a "pairwise comparison" metric measuring mutual dominance of proposed solutions; this superiority became even more distinct with increasing runtimes.

Future research will focus on real world requirements by, for instance, considering stochastic influences on activity durations, applying the approaches to large networks and accounting for constrained resources. A further enhancement would provide an additional option for speeding up a project by making it possible to change the activity sequence through a change in a network's topology. On the level of multiobjective heuristic optimization, the experimental comparison should be extended to other metaheuristic techniques (e. g., memetic algorithms or particle swarm optimization) and/or a module for automatic parameter tuning (e. g., based on F-Race [2]) may be added.

## Acknowledgements

The authors would like to express their appreciation to Monika Treipl for her valuable technical support and to Michael Halling who was involved in an early stage of the project. Furthermore, the authors thank three reviewers for valuable comments on a previous version of this paper.

## Appendix: Parameter Tests

Our results are based on specific parameter settings for the three tested algorithms. It cannot be expected that these settings are optimal for each test instance: Even if the parameter combinations would be "optimized" in some sense for each of the tested algorithms on a specific instance (note that due to the existence of more than one evaluation metric, parameter tuning is not a single-objective optimization task, but a multicriteria decision problem!), it would still remain questionable whether the chosen parameter combination is still suitable for the other test instances or for new instances as they will occur in practical applications. Nevertheless, for our results to be conclusive, it is important to verify at least two properties: First, we have to convince ourselves that the performance of the three heuristic algorithms is at least *relatively robust* with respect to parameter changes. Secondly, we must have evidence that in our tests, the chosen parameter values did not *systematically* favor one of the algorithms over the other two.

To check this, we tested different alternative parameter sets for the three algorithmic approaches, and we compared the results with the standard parameter sets reported in Section 5.2. To each algorithm, we also tested three modified variants of it: We modified the evaporation rate for the PACO algorithm (variant (1), (2), (3)), we modified the cooling process in the PSA algorithms (variant (4), (5), (6)), and we modified the mutation rate in the NSGA-II (variant (7), (8), (9)). Table 9 contains the modified settings. Each modified algorithm was tested against the standard variants of the two other algorithms. Then, we compared the variants on the basis of our used metrics.

The results are shown in Tables 10–13. The numbers in the tables indicate for how many test instances (from our set of 12 instances) the algorithm named in the corresponding line, eventually modified as indicated in the corresponding column, is the winner of the three algorithms in solution quality; the numbers in dots report for how many test instances the algorithm is the second best. It can be observed that the results obtained by our standard parameter settings remain rather robust after performing the changes. Sometimes the solution quality is improved by a parameter modification, sometimes it gets worse, but it never happens that the overall ranking of the algorithms is changed. To some extent, the observed (small) changes seem to be due to random effects rather than due to significant differences in performance: e.g., the solution quality became slightly better when we decreased the mutation rate in the NSGA-II as well as when we increased the mutation rate.

Moreover, Tables 10–13 show that all the three algorithms can profit in a comparable order of magnitude from a fine-tuning of the parameters, such that our tests with the standard parameter settings did not systematically prejudice one of the algorithms. This indicates that the main experimental results of this paper are stable with respect to parameter modifications.

Table 9
Parameter settings for numerical experiments.

| PACO | PSA | NSGA-II |
|---|---|---|
| (0) see Table 2 | (0) see Table 2 | (0) see Table 2 |
| (1) $\rho = 0.8$ | (4) $L = 4, b = 0.9$ | (7) $\mu = 0.05$ |
| (2) $\rho = 0.95$ | (5) $L = 6, b = 0.85$ | (8) $\mu = 0.2$ |
| (3) $\rho = 0.99$ | (6) $a = 1.01$ | (9) $\mu = 0.3$ |

Table 10
Evaluation on Metric $Q_4$

| $Q_4$ | (0) | (1) | (2) | (3) |
|---|---|---|---|---|
| PACO | 10 (2) | 12 (0) | 10 (2) | 5 (7) |
| PSA | 2 (0) | 0 (1) | 0 (2) | 0 (1) |
| NSGA-II | 0 (10) | 0 (11) | 3 (8) | 7 (4) |

| $Q_4$ | (0) | (4) | (5) | (6) |
|---|---|---|---|---|
| PACO | 10 (2) | 10 (2) | 10 (2) | 10 (2) |
| PSA | 2 (0) | 1 (0) | 1 (0) | 1 (0) |
| NSGA-II | 0 (10) | 1 (10) | 1 (10) | 1 (10) |

| $Q_4$ | (0) | (7) | (8) | (9) |
|---|---|---|---|---|
| PACO | 10 (2) | 10 (0) | 9 (1) | 10 (0) |
| PSA | 2 (0) | 1 (1) | 1 (1) | 0 (2) |
| NSGA-II | 0 (10) | 1 (11) | 2 (10) | 2 (10) |

Table 11
Evaluation on Metric $Q_5$

| $Q_5$ | (0) | (1) | (2) | (3) |
|---|---|---|---|---|
| PACO | 1 (3) | 1 (4) | 2 (2) | 3 (3) |
| PSA | 1 (9) | 2 (8) | 2 (6) | 2 (6) |
| NSGA-II | 11 (1) | 10 (2) | 8 (4) | 9 (3) |
| $Q_5$ | (0) | (4) | (5) | (6) |
| PACO | 1 (3) | 0 (5) | 1 (6) | 1 (4) |
| PSA | 1 (9) | 3 (5) | 3 (5) | 2 (5) |
| NSGA-II | 11 (1) | 9 (2) | 9 (2) | 9 (3) |
| $Q_5$ | (0) | (7) | (8) | (9) |
| PACO | 1 (3) | 1 (3) | 0 (5) | 1 (3) |
| PSA | 1(9) | 0 (10) | 1 (8) | 3 (6) |
| NSGA-II | 11 (1) | 12 (0) | 12 (0) | 9 (2) |

Table 12
Evaluation on Metric $Q_R$

| $Q_R$ | (0) | (1) | (2) | (3) |
|---|---|---|---|---|
| PACO | 10 (2) | 11 (1) | 11 (1) | 12 (0) |
| PSA | 2 (10) | 1 (11) | 1 (11) | 0 (12) |
| NSGA-II | 0 (0) | 0 (0) | 0(0) | 0 (0) |
| $Q_R$ | (0) | (4) | (5) | (6) |
| PACO | 10 (2) | 10 (2) | 10 (2) | 10 (2) |
| PSA | 2 (10) | 2 (10) | 2 (10) | 2 (10) |
| NSGA-II | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| $Q_R$ | (0) | (7) | (8) | (9) |
| PACO | 10 (2) | 10 (0) | 10 (0) | 10 (0) |
| PSA | 2 (10) | 1 (11) | 1 (11) | 2 (10) |
| NSGA-II | 0 (0) | 1 (1) | 1 (1) | 0 (2) |

## References

[1] Bauer A, Bullnheimer B, Hartl R, Strauss C. Minimizing total tardiness on a single machine using Ant Colony Optimization. Central European Journal of

Table 13
Proposed efficient solutions

| $PE$ | (0) | (1) | (2) | (3) |
|---|---|---|---|---|
| PACO | 10 (1) | 12 (0) | 11 (1) | 11 (1) |
| PSA | 2 (6) | 0 (7) | 1 (7) | 1 (7) |
| NSGA-II | 0 (5) | 0 (5) | 0 (4) | 0 (4) |
| $PE$ | (0) | (4) | (5) | (6) |
| PACO | 10 (1) | 9 (2) | 9 (2) | 9 (2) |
| PSA | 2 (6) | 3 (8) | 3 (8) | 3 (8) |
| NSGA-II | 0 (5) | 0 (0) | 0 (2) | 0 (2) |
| $PE$ | (0) | (7) | (8) | (9) |
| PACO | 10 (1) | 10 (1) | 10 (0) | 10 (0) |
| PSA | 2 (6) | 0 (7) | 0 (9) | 1 (10) |
| NSGA-II | 0 (5) | 2 (5) | 2 (2) | 1 (2) |

Table 14
Evaluation on Metric $Q_6$

| $Q_6$ | (0) | (1) | (2) | (3) |
|---|---|---|---|---|
| PACO | 0 (11) | 0 (12) | 0 (12) | 0 (12) |
| PSA | 0 (1) | 0 (0) | 0 (0) | 0 (0) |
| NSGA-II | 12 (0) | 12 (0) | 12 (0) | 12 (0) |
| $Q_6$ | (0) | (4) | (5) | (6) |
| PACO | 0 (11) | 0 (11) | 0 (11) | 0 (11) |
| PSA | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| NSGA-II | 12 (0) | 12 (0) | 12 (0) | 12 (0) |
| $Q_6$ | (0) | (7) | (8) | (9) |
| PACO | 0 (11) | 0 (11) | 0 (11) | 0 (11) |
| PSA | 0 (1) | 0 (1) | 0 (1) | 0 (1) |
| NSGA-II | 12 (0) | 12 (0) | 12 (0) | 12 (0) |

Operations Research 2000;8(2):125-141.

[2] Birrattari M, Stützle T, Paquete L, Varrentrapp K. A racing algorithm for configuring metaheuristics. In: Langdon WB et al., editors. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002). San Francisco: Morgan Kaufmann, 2002. p. 11-18.

[3] Bullnheimer B, Hartl RF, Strauss C. An improved Ant System algorithm for the vehicle routing problem. Annals of Operations Research 1999;89(1-4):319-328.

[4] Coello Coello CA, Van Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems. New York: Kluwer, 2002.

[5] Collette Y, Siarry P. Three new metrics to measure the convergence of metaheuristics towards the Pareto frontier and the aesthetic of a set solutions in biobjective optimization. Computers and Operations Research 2005;32(4):773-792.

[6] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Varela FJ, Bourgine P, editors. Toward a practice of autonomous systems: Proceedings of The First European Conference on Artificial Life. Elsevier, 1992. p. 134-142.

[7] Czyzak P, Jaszkiewicz A. A multiobjective metaheuristic approach to the localization of a chain of petrol stations by the capital budgeting model. Control and Cybernetics 1996;25(1):177-187.

[8] Czyzak P, Jaszkiewicz A. Pareto Simulated Annealing: a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 1998;7(1):34-47.

[9] De P, Dunne EJ, Gosh JB, Wells CE. Complexity of the discrete time-cost tradeoff problem for project networks. Operations Research 1997;45(2):302-306.

[10] Deb K. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6(2):182-197.

[11] Demeulemeester EL, Herroelen WS. Project scheduling: a research handbook. Boston, Kluwer, 2002.

[12] Doerner KF, Gronalt M, Hartl RF, Reimann M, Strauss C, Stummer M. SavingsAnts for the Vehicle Routing Problem. In: Cagnoni S et al., editors. Applications of evolutionary computing: EvoWorkshops 2002. Berlin: Springer, 2002. p. 11-20.

[13] Doerner KF, Gutjahr WJ, Hartl RF, Strauss C, Stummer C. Pareto Ant Colony Optimization: a metaheuristic approach to multiobjective portfolio selection. Annals of Operations Research 2004;131(1-4):79-99.

[14] Doerner KF, Gutjahr WJ, Hartl RF, Strauss C, Stummer C. Pareto Ant Colony Optimization with ILP preprocessing in multiobjective project portfolio selection. European Journal of Operational Research (to appear).

[15] Doerner KF, Gutjahr WJ, Kotsis G, Polaschek M, Strauss C. Enriched workflow modelling and stochastic Branch-and-Bound. European Journal of Operational Research 2006 (to appear).

[16] Doerner KF, Hartl RF, Reimann M. Cooperative Ant Colonies for optimizing resource allocation in transportation. In: Boers EJW et al., editors. Applications of evolutionary computing: EvoWorkshops 2001. Berlin: Springer, 2001. p. 70-79.

[17] Ehrgott M, Gandibleux X. Approximative solution methods for multiobjective combinatorial optimization. Top 2004;12(1):1-63.

[18] Gambardella L, Taillard E, Agazzi G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F, editors. New ideas in optimization. London: McGraw-Hill, 1999. p. 64-76.

[19] Gutjahr WJ. ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters 2002;82(3):145-153.

[20] Gutjahr WJ, Strauss C, Toth M. Crashing of stochastic processes by sampling and optimisation. Business Process Management Journal 2000;6(2):65-83.

[21] Gutjahr WJ, Strauss C, Wagner E. A stochastic Branch-and-Bound approach to activity crashing in project management. INFORMS Journal on Computing 2000;12(2):125-135.

[22] Hindelang TJ, Muth JF. A dynamic programming algorithm for decision CPM networks. Operations Research 1979;27(2):225-241.

[23] Jaszkiewicz A. Evaluation of multiple objective metaheuristics. In: Gandibleux X et al., editors. Metaheuristics for multiobjective optimization. Springer: Berlin, 2004. p. 66-89.

[24] Kolisch R, Sprecher A. PSPLIB: a project scheduling problem library. European Journal of Operational Research 1997;96(1):205-216.

[25] Moder JJ, Phillips CR, Davis EW. Project management with CPM, PERT and Precedence Diagramming. New York: Van Nostrand, 1983.

[26] Panagiotakopoulos D. A CPM time-cost computational algorithm for arbitrary activity cost functions. INFOR 1977;15(2):183-195.

[27] Sarker R, Newton C. Solving a multiple objective linear program using simulated annealing. Asia-Pacific Journal of Operational Research 2001;18(1):109-120.

[28] Schott JR. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Dept. of Aeronautics and Astronautics, MIT. Cambridge, 1995.

[29] Serafini P. Simulated annealing for multi objective optimization problems. In: Tzeng G et al., editors. Multiple criteria decision making: expand and enrich the domains of thinking and application. New York: Springer, 1994. p. 283-292.

[30] Stummer C, Sun M. New multiobjective metaheuristic solution procedures for capital investment planning. Journal of Heuristics 2005;11(3):183-199.

[31] Sunde L, Lichtenberg S. Net-present-value cost/time tradeoff. International Journal of Project Management 1995;13(1):45-49.

[32] Tavares LV. A review of the contribution of operational research to project management. European Journal of Operational Research 2002;136(1):1-18.

[33] Ulungu EL, Teghem J, Fortemps P. Heuristics for multiobjective combinatorial optimization by simulated annealing. In: Gu J et al., editors. Proceedings of the Sixth National Conference on Multiple Criteria Decision Making. Windsor, 1995. p. 228-238.

[34] Vincke P. Multicriteria decision-aid. Chichester: Wiley, 1992.

[35] Witus G. Decision support for planning and resource allocation in hierarchical organizations. IEEE Transactions on Systems, Man, and Cybernetics 1986;16(6):927-942.

[36] Yau C, Ritchie E. Project compression: a method for speeding up resource constrained projects which preserve the activity schedule. European Journal of Operational Research 1990;49(1):140-152.

[37] Zitzler E, Thiele L. Multiple objective evolutionary algorithms: a comparative case study and the strength of the Pareto approach. IEEE Transactions on Evolutionary Computation 1999;3(4):257-271.

[38] Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 2003;7(2):117-132.