Pareto Ant Colony Optimization with ILP preprocessing in multiobjective project portfolio selection

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

[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*

[c] *Department of Management Science and Statistics, University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio, Texas 78249-0631,USA*

Abstract

One of the most important, common and critical management issues lies in determining the "best" project portfolio out of a given set of investment proposals. As this decision process usually involves the pursuit of multiple objectives amid a lack of a priori preference information, its quality can be improved by implementing a two-phase procedure that first identifies the solution space of all efficient (i. e., Pareto-optimal) portfolios and then allows an interactive exploration of that space. However, determining the solution space is not trivial because brute-force complete enumeration only solves small instances and the underlying NP-hard problem becomes increasingly demanding as the number of projects grows. While metaheuristics in general provide an attractive compromise between the computational effort necessary and the quality of an approximated solution space, Pareto Ant Colony Optimization (P-ACO) has been shown to perform particularly well for this class of problems. In this paper, the beneficial effect of P-ACO's core function (i. e., the learning feature) is substantiated by means of a numerical example based on real world data. Furthermore, the original P-ACO approach is supplemented by an integer linear programming (ILP) preprocessing procedure that identifies several efficient portfolio solutions within a few seconds and correspondingly initializes the pheromone trails before running P-ACO. This extension favors a larger exploration of the search space at the beginning of the search and does so at a low cost.

Key words: Ant Colony Optimization, Project Portfolio Selection, Multiobjective Combinatorial Optimization, Integer Linear Programming, Preprocessing, Hybrid Optimization Method

1 Introduction

Multiobjective combinatorial optimization (MOCO) has become a very active area of research [6,7] with selecting a portfolio of projects out of dozens of competing (capital) investment proposals being an application of particular high practical relevance (e. g., in research and development investment planning [10]). The recent interest in MOCO last but not least is due to the fact that many managers no longer are willing to provide extensive a priori preference information such as complex utility functions, but instead demand for decision support that allows them to only gradually specify their preferences and, thus, to participate and to control the decision process. Typically, a corresponding system starts off with (partially) identifying the efficient (i. e., non-dominated or Pareto-optimal) portfolio candidates and then supports the decision-maker to interactively explore these solutions. However, the first-mentioned task is an NP hard problem and, thus, (meta-)heuristic approaches come into play as they provide an attractive tradeoff between solution quality and the computational effort required for determining a sufficient approximation of the solution space.

Accordingly, several multiobjective versions of Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) have been developed so far (for a survey cf. [7]), but it has not been before recently that a proper Ant Colony System has been proposed as well. Although this so-called Pareto Ant Colony Optimization (P-ACO; cf. [3,4]) approach already outperforms SA and GA approaches for the investigated project portfolio selection problem, its performance can be further enhanced by integrating an initial preprocessing step. To that end, this paper will describe an integer programming (IP) procedure that

identifies several (supported) efficient solutions within a few seconds and utilizes them to appropriately initialize the pheromone trails (that represent the learning component of the algorithm). The effect of achieving higher diversification at comparatively low cost will be demonstrated by means of numerical experiments.

The remainder of the paper is organized as follows: Section 2 describes the project portfolio selection problem at hand. Section 3 provides an overview of the P-ACO approach, while Section 4 introduces the supplemental IP preprocessing procedure and presents numerical results. Finally, Section 5 offers conclusions and suggests directions for further research.

2 Problem Description

The problem formulation of determining the "best" project portfolio (i. e., subset) out of a given set of N research and development project proposals originated from a cooperation with the ebm-papst GmbH & Co. KG, a German enterprise that grew to the world market leader with innovative external rotor motors and fans; a detailed description of the underlying integer linear mathematical programming model is provided by Stummer and Heidenberger [14]. In short, portfolios are modeled as vectors $x = (x_1, \ldots, x_N)$, where the binary variables x_i indicate whether a project i is part of the portfolio ($x_i = 1$) or not ($x_i = 0$). Projects are characterized by (i) the benefits $b_{i,l,t}$ they provide in the B benefit categories l (e. g., cash flow and/or sales) and the T planning periods t (e. g., financial years) and by (ii) their resource consumptions $r_{i,q,t}$ in the R resource categories q (e. g., funds, manpower, and production capacity). The benefit values of a project portfolio x thus is computed as

$$b_{l,t}(x) = \sum_i b_{i,l,t} \cdot x_i + \sum_j d_{j,l,t}(x) \qquad (1)$$
$$\text{for } l = 1, \ldots, B \text{ and } t = 1, \ldots, T,$$

where the sum of the project benefits is adjusted by potential synergy or cannibalism effects $d_{j,l,t}(x)$. Note that functions $d_{j,l,t}(x)$ remain linear though the J project interactions j usually refer to more than just two investment proposals (for details cf. [14]). Required resources $r_{q,t}(x)$ of type q (with $q = 1, \ldots, R$) and for planning period t are determined in a similar way.

Furthermore, the model traces both benefit and resource values separately for each planning period instead of aggregating them to a (discounted) total value and, thus, provides valuable time profiles of expected benefits and costs (for a discussion cf. [11]). The resulting $K = B \cdot T$ objectives may be formulated as

$$u_k(x) = b_{l,t}(x) \quad \text{for } k = l + (t-1) \cdot B\,. \qquad (2)$$

These objectives are subject to two types of restrictions. The first group of restrictions ensures that each feasible portfolio contains no more than a given maximum (not less than a given minimum) number of projects out of a certain subset of projects. Thus, a minimum number of projects that deal with emerging technologies can be guaranteed, the number of projects based on conventional concepts (even if they seem attractive in a short-time perspective) may be restricted, or balancing policies (e. g., with respect to novel and ongoing projects) may be implemented. A representative example for these restrictions is

$$\sum_i v_{i,j} \cdot x_i \geq m_j, \qquad (3)$$

where m_j stands for a required minimum number of predefined projects that have to be included in a portfolio and $v_{i,j}$ indicates whether or not project i

is in the corresponding subset j of effected projects.

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

$$r_{q,t}(x) \leq R_{q,t} \quad \text{for } q = 1, \ldots, R \text{ and } t = 1, \ldots, T, \text{ and} \tag{4}$$

$$b_{l,t}(x) \geq B_{l,t} \quad \text{for } l = 1, \ldots, B \text{ and } t = 1, \ldots, T. \tag{5}$$

Apparently, this problem is a generalization of the classical bin packing problem which is known to be NP-hard.

3 Pareto Ant Colony Optimization

3.1 Solution Procedure

The Ant Colony approach imitates the behavior shown by real ants when searching for food. They communicate information about food sources via pheromone, which they secrete as they move along. When an ant finds a food source it returns to the nest. As ants on short (i. e., attractive) paths will return to the nest faster, more pheromone will be deposited on the shorter paths. Moving ants accordingly choose their path with a probability that depends on the amount of pheromone detected and, consequently, paths that are more frequently travelled become more attractive and, by means of that self-strengthening behavior, will be used more often. Further, the pheromone "evaporates" over time, so that pheromone trails of infrequently travelled paths become weaker while attractive paths are reinforced. And finally, artificial ants not only imitate the learning behavior described above, but often

apply additional, problem-specific heuristic information. While such artificial ant colony systems have been successfully applied to various single-objective problems, several extensions have been necessary in order to be able to tackle the multiobjective project portfolio selection problem at hand (for a detailed discussion of these modifications cf. [4]).

Basically, each iteration of the P-ACO algorithm starts off with generating Γ ants each with an empty portfolio $x = (0, \ldots, 0)$ and randomly generated lifespan Ξ and objective weights (i. e., individual preferences) $p = (p_1, \ldots, p_K)$. In the succeeding construction step, each ant adds projects to its portfolio x and to that end applies a pseudo-random-proportional project selection rule that is influenced by the heuristic information η_i and the pheromone information τ_i (cf. Section 3.2 below). If both the feasibility and efficiency of the constructed portfolio are confirmed, the solution candidate is stored in the set of the proposed efficient project portfolios and candidates that are dominated by the new entry are removed from that set. After all ants of a population have delivered their portfolios, the best as well as the second-best portfolio x for each objective k are used to properly update the pheromone values (cf. Section 3.3 below). In pseudo-code, the P-ACO algorithm can be outlined as follows:

procedure `P-ACO` () {
  Initialization of `P-ACO`; /* create Γ ants, set pheromone vectors to 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..N];
      set $x = (0, ..., 0)$; /* 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 (6) 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 (9)
} } }

3.2 Decision Rule

For each objective k the pheromone information is stored in a vector $(\tau_1^k, \ldots, \tau_N^k)$ where τ_i^k indicates whether or not adding a project i promises favorable effects on a portfolio's outcome. The heuristic information (often also called "visibility"), on the other hand, refers to the fitting of a particular project candidate with respect to a partially constructed portfolio and, accordingly, an aggregated value of visibility $0 \leq \eta_i(x) \leq 1$ is calculated for each available project candidate i (in a matched manner for our problem at hand; for the sake of brevity we omit the details and refer the reader to [4]). Based on the above information the following pseudo-random-proportional decision rule is

used to add another project to the current portfolio:

$$i = \begin{cases} \arg\max_{i \in \Omega(x)} \left\{ \left[\sum_{k=1}^{K} \left(p_k \cdot \tau_i^k \right) \right]^{\alpha} \cdot [\eta_i(x)]^{\beta} \right\} & \text{if} \quad q \leq q_0 \\ \\ \hat{i} & \text{otherwise,} \end{cases} \tag{6}$$

where q is a random number uniformly distributed in $[0, 1)$ and q_0 is a parameter ($0 \leq q_0 < 1$) representing the probability that the project with the highest aggregate value for pheromone and visibility is selected. Given that the drawing of q results in a value such that $q > q_0$, the random variable $\hat{i}$ is selected according to the following probability distribution:

$$\mathcal{P}_i(x) = \begin{cases} \frac{\left[\sum_{k=1}^{K} \left(p_k \cdot \tau_i^k \right) \right]^{\alpha} \cdot [\eta_i(x)]^{\beta}}{\sum_{h \in \Omega(x)} \left(\left[\sum_{k=1}^{K} \left(p_k \cdot \tau_h^k \right) \right]^{\alpha} \cdot [\eta_h(x)]^{\beta} \right)} & \text{if} \quad i \in \Omega \\ \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

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

3.3 Pheromone Update

A local pheromone update is performed once an artificial ant has added a project i to a portfolio. Then, pheromone values τ_i^k are decreased for all K pheromone vectors by applying the local pheromone update rule

$$\tau_i^k = (1 - \rho) \cdot \tau_i^k + \rho \cdot \tau_0 \,, \tag{8}$$

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 projects that have not yet been chosen. As a result, the diversity of the solutions is enhanced.

The global pheromone takes place right after all ants of a population have

proposed portfolio solutions and their feasibility and efficiency have been determined. The update rule for each objective k is

$$\tau_i^k = (1 - \rho) \cdot \tau_i^k + \rho \cdot \Delta\tau_i^k, \tag{9}$$

where ρ stands for the evaporation rate (with $0 \leq \rho \leq 1$). Pheromone information is increased by a quantity $\Delta\tau_i^k = 2\delta$ if a project i is included in a population's best portfolio with respect to objective k (and $\Delta\tau_i^k = 0$ otherwise). Subsequently, another (analogous) update is made for the second-best portfolio using a pheromon quantity of $\Delta\tau_i^k = \delta$. Tests with various pheromone update strategies have shown that working with just the two best ants and assigning pheromone quantity $\delta = 5$ leads to attractive results. However, it is noteworthy, that parameter modifications have only small influence on P-ACO's overall performance [4].

4 The ILP Preprocessing Procedure

Several dozen iterations are necessary in order to approach the efficient frontier when starting from scratch (i. e., with all pheromone values τ_i^k initialized by a constant value of $\tau_0 = 1$). Since the management of pheromone is quite expensive, the basic idea behind introducing an additional preprocessing step lies in shortening that "start-up-phase" [12,13]. To achieve this aim, an integer programming model is used to identify several efficient project portfolios without accessing pheromone information. These solutions are then utilized to appropriately initialize the pheromone vectors (i. e., to set initial values for all the τ_i^k) before the P-ACO algorithm is actually started.

Procedurally, the preprocessing step consists of two phases: In phase 1, a

single-objective problem is derived from the original multiobjective model, thereby making it possible to solve several (single-objective) linear problem instances of the integer programming model. The model generation is based on repeatedly drawing random objective weights w_k from a uniform distribution within the domain $[0, 1)$. Alternatively, these weights could be determined systematically; whether this approach is advantageous will be subject to further research. Regardless of which approach is used, each set of weights makes it possible to formulate a separate (binary) linear program that seeks to maximize

$$u(x) = \sum_k w_k \cdot u_k(x), \tag{10}$$

while still taking into consideration the three types of constraints (3)-(5) described above. However, one should note that different weights may nevertheless result in identical solutions. Accordingly, the number of efficient portfolios identified in the preprocessing step depends primarily on both the chosen number of iterations and the number of supported efficient solutions that are generally available (a figure which varies considerably among individual problems). Furthermore, the preprocessing procedure will only identify so-called supported efficient portfolios, which regularly comprise only a small subset of the set of all efficient solutions (for a discussion cf. [6]).

The aim of phase 2 in the preprocessing step is to make use of the identified (supported) efficient portfolios and to appropriately initialize the pheromone vectors τ^k. To that end, we investigated two procedures. The first simply involves searching for the best and the second-best portfolio with respect to each objective k. Beginning from a default setting of $\tau_i^k = 1$ the initial pheromone value is increased by $\delta \cdot \rho = 0.5$ (note that ρ was set to $\rho = 0.1$ in our experiments) for projects i that are included in the second-best portfolio $x^k_{\text{second}-\text{best}}$

and/or by another $2\delta \cdot \rho = 1$ for projects being included in the best portfolio x^k_{best}:

$$\tau^k_i = 1 + \delta \cdot \rho \cdot x^k_{\text{second-best},i} + 2\delta \cdot \rho \cdot x^k_{\text{best},i} \qquad \forall i, k. \tag{11}$$

The second procedure lets all the portfolios found in the preprocessing step influence the setting of the initial pheromone values. For that purpose one may normalize the objective values $u_k(x)$ to a range of $[0, 1]$:

$$\hat{u}_k(x) = \frac{u_k(x) - u^{\min}_k}{u^{\max}_k - u^{\min}_k} \qquad \forall k. \tag{12}$$

Then the projects i contained in a portfolio x add a value of $10 \cdot \rho = 1$ to the pheromone values τ^k_i of objective k where the project portfolio provides the highest normalized objective value $\hat{u}_k(x)$ (compared with the values of portfolio x in the other objectives):

$$\tau^k_i = 1 + \sum_x 10 \cdot \rho \cdot b_k(x) \cdot x_i \qquad \forall i, k. \tag{13}$$

Here, function $b_k(x)$ indicates whether portfolio x has its highest (normalized) objective value in objective k ($b_k(x) = 1$) or not ($b_k(x) = 0$). Numerical experiments showed that the latter procedure performs best, and therefore was applied to the numerical example presented in Section 5.3.

5 Numerical Analysis

5.1 Comparison of Solution Quality

The following section describes the computational tests that were performed in order to substantiate two issues: first, the contribution provided by P-ACO's learning component, which is a conceptual characteristic; and second,

P-ACO's improvement by means of an ILP preprocessing step.

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

The parameter settings of P-ACO chosen for the computational experiments ($\alpha = 1$, $\beta = 1$, $\rho = 0.1$, $\Gamma = 10$) have proven to be advantageous in other applications and were pre-tested for the problem under consideration. These pre-tests led to the parameter setting $q_0 = 0.4$ and $\tau_0 = 1$ being chosen, as a higher level of diversification compared to Dorigo and Gambardella [5], where $q_0 = 0.9$ and $\tau < 1$, is desirable for the application to MOCO problems. In Doerner et al. (2004) [4] a numerical study showed the superiority of the development of the solution quality of P-ACO over that of a simulated annealing re-implementation, PSA by Czyzak and Jaszkiewicz [1], and a genetic algorithm, NSGA by Deb [2]. For the sake of completeness we furthermore implemented a tabu search approach, MOTS by Hansen [9]. The test instance is based on real world data provided by the R&D department of ebm-papst GmbH & Co. KG (cf. Section 2). The results in section 5.1 and 5.2 are based on this real-world example. The characteristics of the problem are explained in detail in section 5.2. For the results in section 5.3 we generated eight additional random problem instances with the same number of projects as in the real-world example, with five and ten objectives and with few and many restrictions.

Figure 1 shows that P-ACO suggests faster a greater number of actually efficient solutions than any of the other three alternative approaches for the problem at hand. The upper left graph in figure 1 shows the results in terms

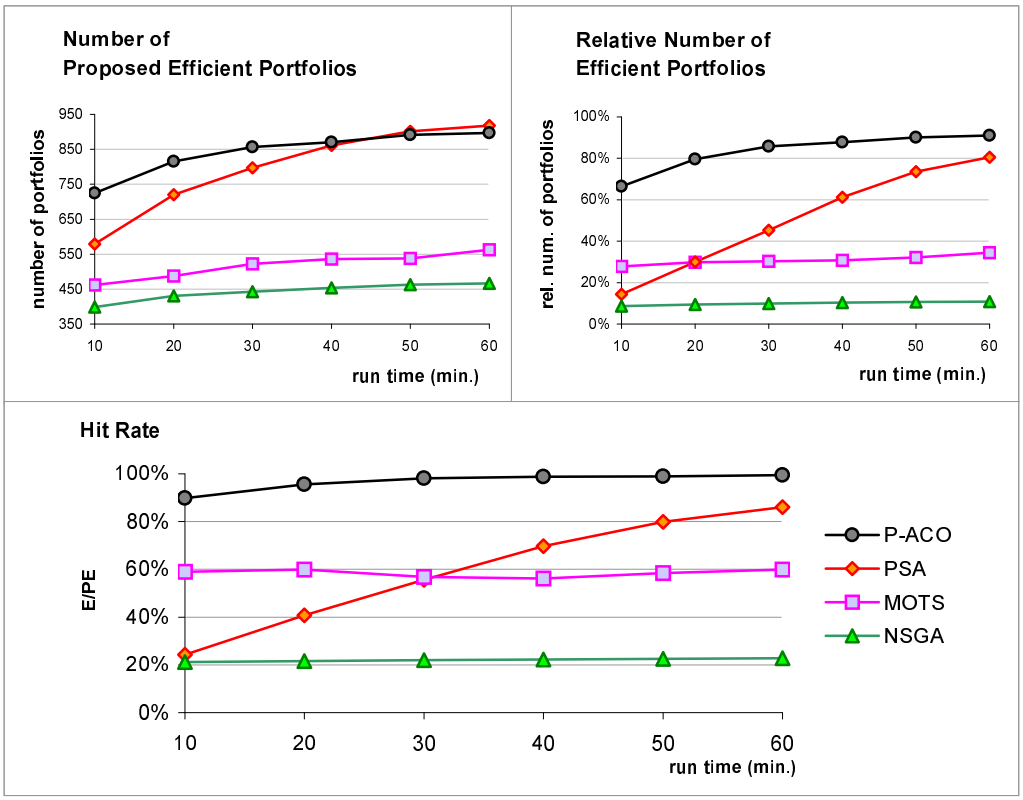

Fig. 1. Comparison of P-ACO's solution quality with three alternative approaches

of absolut values, whereas the upper right graph gives the relative number of actually efficient portfolios; the bottom graph visualizes the "hit rate" which is the ratio of the actually efficient portfolios compared to the proposed ones. The hit rate of MOTS even declines after 30 and 40 minutes of run time as it proposes more portfolios as efficient ones which turn out to be dominated ones. Further tests have shown, that PSA does not generate better results than P-ACO even when run times are doubled for both approaches.

While all other competing approaches have a complexity of $O(N)$, the complexity of the introduced P-ACO approach using preprocessing is $O(N^2)$, where N denotes the number of projects. It can be reduced by other pos-

sible problem encodings, which was not investigated in this paper. The results of P-ACO with preprocessing are superior although it's theoretical complexity is higher than the other approaches' complexity.

5.2 Value of Learning

We present a numerical study that quantifies the value of learning by comparing the results of an approach based solely on heuristic information (i. e., P-ACO with a disabled learning function by setting $\alpha = 0$ in Equation (7)) with those achieved by applying P-ACO, which benefits from its learning component. In the context of P-ACO "learning" denotes the management of information (collecting, storing, and discarding) about previous solutions in the pheromone trails that aims to improve future solutions. Making use of the above mentioned real world data, the numerical study outlines a rather complex decision-making situation in which any "intuitive" favoring of certain project combinations in advance is not permitted. 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 project portfolios.

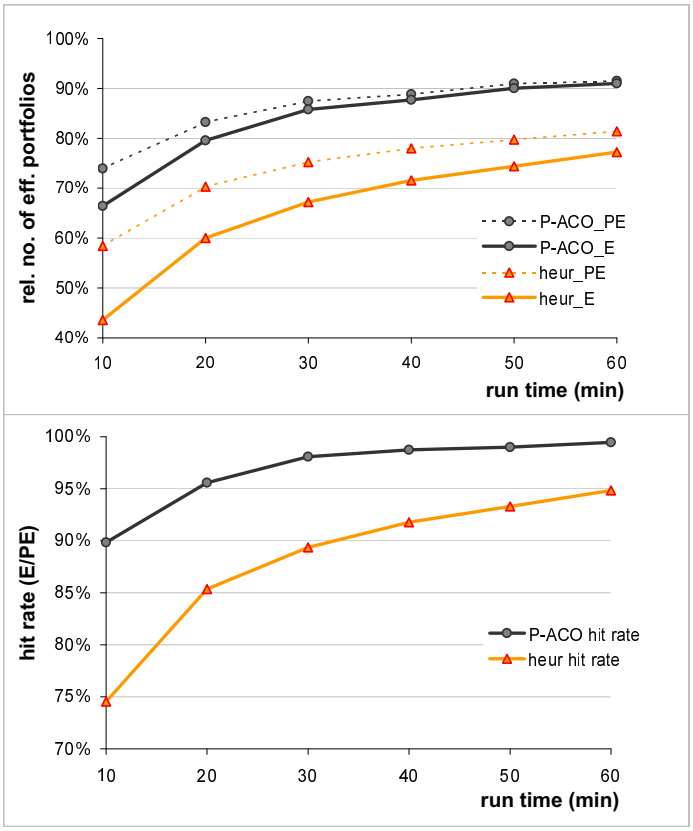

Fig. 2. The Value of Learning

To provide a yardstick for comparing the results, we have chosen the relative number PE of proposed efficient portfolios and the relative number E of proposed portfolios appearing in the efficient set (i. e., those proven actually efficient through complete enumeration). We observe the values found by each approach (i. e., the pure heuristic and P-ACO) after 10, 20, 30, 40, 50 and 60 minutes of run time. The upper graph of figure 2 shows the relative num-

ber of proposed efficient (PE) portfolios as dashed lines, whereas the relative number of actually efficient (E) portfolios is represented by a bold line. After 10 minutes of run time, P-ACO provides 23% more efficient portfolios than the heuristic approach; in other words, the learning component improves the heuristic by more than 50%. After 60 minutes of run time, the gap between the two approaches is still 14%, which is an improvement of the heuristic by a third. Furthermore, the numerical study shows that P-ACO includes relatively few dominated portfolios among the potentially efficient portfolios that it identifies; in contrast, the percentage of portfolios that the pure heuristic approach proposes as efficient ones that are in fact dominated ones is comparatively high.

This distinction is apparent in the upper graph in figure 2 as the distance between the dashed and the solid line, which decreases for both approaches over time. The lower part of figure 2 focuses on that ratio and explicitly visualizes the development of the ratio of efficient to proposed efficient portfolios over time for both approaches. The learning component of P-ACO provides a hit rate of 90% after 10 minutes of run time, whereas the heuristic only manages to attain 75% per cent. Even after one hour of run time, there is a 5% difference. The fact that less than 0.1% of the total search space had to be analyzed (i. e., on average 0.85 million portfolios) to establish 92% of the efficient project portfolios (after 60 minutes) can be interpreted as a promising indicator for P-ACO's ability to generate satisfying solutions within a reasonable computation timeframe; this would seem to hold true even for problems that are too large to be enumerated completely. Our numerical tests establish that P-ACO is unquestionably superior to the pure heuristic and provide evidence that ACO's learning feature makes an essential contribution to the

solution quality.

5.3 Improvements by Preprocessing

The following section provides results for the computational tests, which were performed in order to provide an insight into how the solution quality of P-ACO can be enhanced by applying the described ILP preprocessing procedure. Figure 3 shows in the upper graph the results computed for the real world problem and several comparable random problem instances with thirty portfolios, five to ten objectives and a small number of restrictions generated with the problem generator described in Doerner et al. [4]. We present values averaged over five computational runs per problem instance. The run time alternatives were set to 2, 4, 6, 8, and 10 minutes, thus providing an insight into the development of the solution quality of each of the three alternative preprocessing variations. To provide a fair comparison between all approaches, the time required to perform the preprocessing procedure reduces the run time available to perform the subsequent P-ACO procedure itself. The bottom graph in figure 3 visualizes the results of one selected problem instance for which the complete enumeration revealed that this instance has 621 efficient project portfolios.

The first alternative A aims at an improvement of the adaptive memory used by P-ACO. It adjusts the initial pheromone vectors by taking those supported efficient portfolios into account, which were determined in the ILP preprocessing step. The second alternative B focuses on the data structure supporting identification, storage and retrieval of non-dominated portfolios proposed, which is a so-called quadtree. Quad trees generalize classic binary trees to

K-dimensions [8]. The nodes of the tree store the project portfolios. 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). This hierarchical structure implies that only a small percentage of all possible pairwise comparisons needs to be performed to verify efficiency [15]. Inserting the supported efficient solutions, which are the result of the a priori ILP optimization, will save some computational expensive insert, delete and reorganization procedures of finally dominated portfolios in the quadtree. The third alternative C combines both variants (A and B) to exploit the outcome of preprocessing.

Alternative C shows better results than the other two (A and B) in terms of number of efficient portfolios found. After 2 minutes of run time the advanced P-ACO version with both variants provides on the average 8% more efficient portfolios than P-ACO without any preprocessing and for the selected case 17% more efficient portfolios. After 4 minutes of run time the difference is still 6% on the average and 5% in the selected instance which corresponds to 32 portfolios out of the total 621 efficient ones. Note that in the selected case less than 20 supported efficient solutions identified through the preprocessing optimization, have lead to an immediate increase of Pareto-optimal solutions by the amount of 100 portfolios after 2 minutes of run time. Exploiting the supported efficient solutions found in the preprocessing step thus lead to an over-proportional pay-back effect: in the case of the two simple approaches A and B the initial 20 supported efficient solutions are augmented by a factor of 4 after 2 minutes of run time, whereas the combined approach C even reaches a factor of 5. Note that exploiting one unique information (i. e., supported efficient portfolios) at two different points in the procedure (i. e., initialization

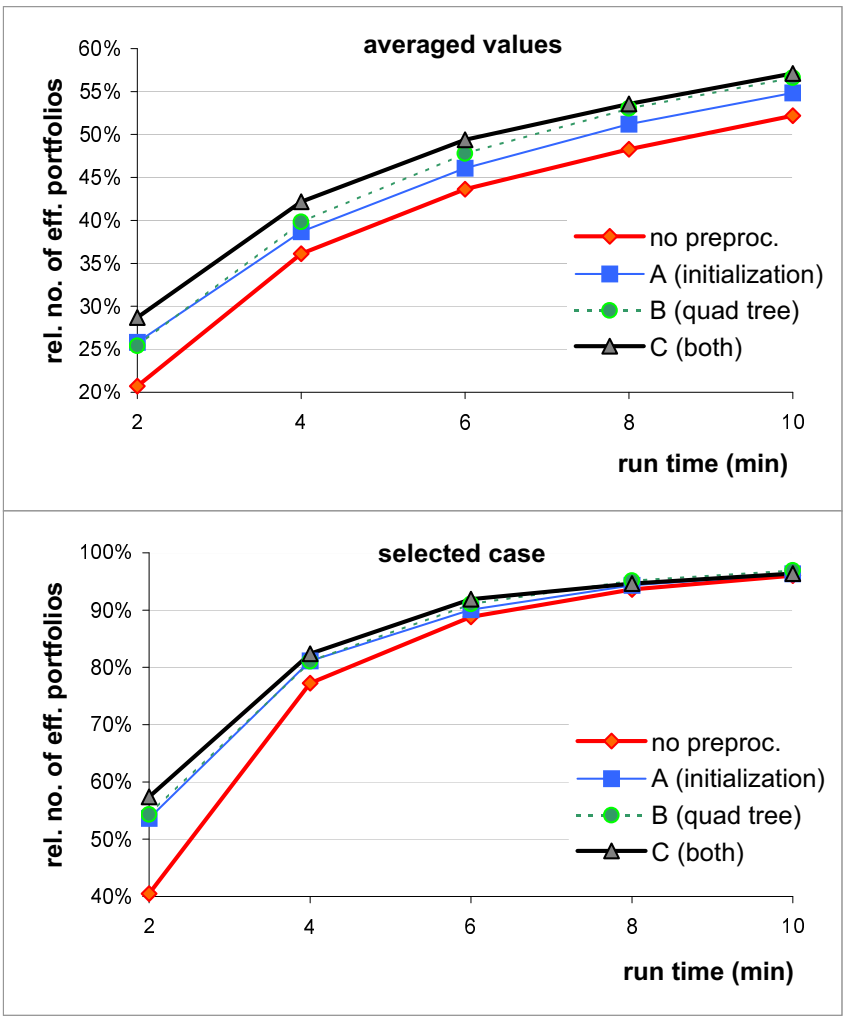

Fig. 3. Improvements by Preprocessing

of the pheromone vectors and nodes in the quad tree, respectively) has an augmenting effect.

In our selected numerical example the benefits of the preprocessing step are levelled out after 10 minutes of run time because almost the entire number of efficient portfolios has been identified. It is worth noting that we were compelled to apply the procedures to a problem size that could still be enumerated

completely in order to provide a quantitative measure of the solutions' quality. However, given that real world problems might contain a far larger number of project proposals, the initial subperiod of two minutes could correspond to far longer computing times when applied to such large problems. Whereas in the selected case the advantage of preprocessing is levelled out quite soon, in the averaged results that trend is clearly slowed down (i. e., the initial 8% advantage declines down to only 6% after 10 minutes of run time).

Other pre-tests have shown that P-ACO with an ILP preprocessing step performs particularly promising on problems where few restrictions are imposed and a large number of efficient project portfolios exists. There are two explanations for this finding: Firstly, it seems that in the mentioned case the ACO-inherent visibility can take considerable advantage of the efficient solutions identified during the preceding preprocessing phase while the basic visibility feature regularly suffices for problems with many restrictions because these restrictions anyway reduce the number of efficient portfolios. The second reason lies in the ILP nature of the pre-process: as the preprocessing procedure focuses on supported efficient portfolios and only few of them exist in problems with numerous maximum and minimum constraints the effect of the pre-process may diminish or even disappear. As the analysis of preprocessing for other meta-heuristics (e. g., Simulated Annealing, Genetic Algorithms, and Tabu Search) goes beyond the scope of this paper, future research will focus on determining the effects of similar preprocessing procedures for meta-heuristics.

6 Conclusions

Influenced by the critical role that multiobjective combinatorial optimization plays in the decision-making process at the strategic management level, recent research activities have focused on heuristic approaches for such NP-hard problems. Our paper substantiates the positive benefits of the learning feature found in Pareto Ant Colony Optimization, which has proven to be an efficient method in this challenging problem class. The influence of the learning feature on the solution quality of P-ACO is shown by providing a numerical study based on real world data. It is demonstrated that the developed P-ACO algorithm yields better results than simply applying the underlying heuristic without learning. Furthermore, it is shown that ILP preprocessing improves performance. The results of numerical experiments involving a two step ILP preprocessing procedure that supports P-ACO underscore the positive effect that the preprocessing procedure has on P-ACO's solution quality. Further research will focus on testing alternative preprocessing functions, investigating alternative, less complex problem encodings and explore what, if any, effects comparable preprocessing procedures have on other meta-heuristics (e. g., Simulated Annealing, Genetic Algorithms, and Tabu Search).

Acknowledgements

The authors would like to thank Leonidas Sakalauskas and three anonymous referees for valuable comments and suggestions which strengthened this paper, Erich Obwexer for the implementation of the ILP model, and Thomas Bertolini from the ebm-papst GmbH & Co. KG, for providing real world data and for

his support of this work. Further, Christian Stummer acknowledges financial support from the Austrian Science Fund (FWF) under Grant No. J2351-N04.

References

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

[2] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, Chichester, 2001.

[3] K. Doerner, W.J. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, Ant Colony Optimization in multiobjective portfolio selection, in: Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, pp. 243-248.

[4] K. Doerner, W.J. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, Pareto Ant Colony Optimization: A metaheuristic approach to multiobjective portfolios selection, Annals of Operations Research, forthcoming.

[5] M. Dorigo, L. Gambardella, Ant Colony System: A cooperative learning approach to the travelling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1997) 53-66.

[6] M. Ehrgott, X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spektrum 22 (2000) 425-460.

[7] M. Ehrgott, X. Gandibleux, Approximative solution methods for multiobjective combinatorial optimization, Top – The Journal of the Spanish Statistical and Operations Research Society, forthcoming.

[8] W. Habenicht, Quad trees: A datastructure for discrete vector optimization problems, in: P. Hansen (Ed.), Essays and Surveys on Multiple Criteria Decision Making, Springer, Berlin, 1983, pp. 136-145.

[9] M. Hansen, Tabu search for multiobjective combinatorial optimization: TAMOCO, Control and Cybernetics 29 (2000) 799-818.

[10] K. Heidenberger, C. Stummer, Research and development project selection and resource allocation: A review of quantitative modelling approaches, International Journal of Management Reviews 1 (1999) 197-224.

[11] J. Ringuest, S. Graves, The linear R&D project selection problem: An alternative to net present value, IEEE Transactions on Engineering Management 37 (1990) 143-146.

[12] C. Solnon, Ants can solve constraint satisfaction problems, IEEE Transactions on Systems, Man and Cybernetics 6 (2002) 347-357.

[13] C. Solnon, Boosting ACO with a preprocessing step, in: E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, H. Tijink (Eds.), Applications of Evolutionary Computing: EvoWorkshops 2002, Springer, Berlin, pp. 163-172.

[14] C. Stummer, K. Heidenberger, Interactive R&D portfolio analysis with project interdependencies and time profiles of multiple objectives, IEEE Transactions on Engineering Management 50 (2003) 175-183.

[15] M. Sun, R.E. Steuer, Quad tree data structures for use in large-scale discrete multiple criteria problems, in: Y. Shi, M. Zeleny (Eds.), New Frontiers of Decision Making for the Information Technology Era, World Scientific, Singapore, 2000, pp. 48-71.