Ant Colony Optimization for Resource-Constrained Project Scheduling

Daniel Merkle[1], Martin Middendorf[2], Hartmut Schmeck[3]
Institute for Applied Computer Science and Formal Description Methods
University of Karlsruhe
D-76128 Karlsruhe, Germany
{[1]merkle,[2]middendorf,[3]schmeck}@aifb.uni-karlsruhe.de
++49 721 608-{[1]6591,[2]3705,[3]4242}

Abstract

An ant colony optimization approach (ACO) for the resource-constrained project scheduling problem (RCPSP) is presented. Combinations of two pheromone evaluation methods are used by the ants to find new solutions. We tested our ACO algorithm on a set of large benchmark problems from the PSPLIB. Compared to several other heuristics for the RCPSP including genetic algorithms, simulated annealing, tabu search, and different sampling methods our algorithm performed best on the average. For some test instances the algorithm was able to find new best solutions.

1 INTRODUCTION

The resource-constrained project scheduling problem (RCPSP) is a general scheduling problem that contains the job-shop, flow-shop, and open-shop problem as special cases. The RCPSP has attracted many researchers during the last years (see (Brucker et al., 1999) for an overview). Since it is an NP-hard problem, different kinds of heuristics for it have been proposed. A comparison of various heuristics on a set of benchmark problems is given in (Hartmann and Kolisch, 2000). The compared heuristics include priority based methods (e.g. multi priority rules, forward-backward scheduling, sampling methods) that use different serial or parallel schedule generation schemes and where the selection of the priority rule is biased by the influence of a random device. Moreover, meta-heuristics as genetic algorithms, a simulated annealing approach, and tabu search have been tested.

In this paper we propose an Ant Colony Optimization (ACO) approach for the RCPSP (see (Dorigo and Di Caro, 1999) for an introduction to ACO). The ACO approach has recently been applied to scheduling problems, as Job-Shop, Flow-Shop, and Single Machine Tardiness problems (see (Bauer et al., 1999; den Besten et al., 1999; Colorni et al., 1994; Merkle and Middendorf, 2000; Stützle, 1998; van der Zwaan and Marques, 1999)). In ACO several generations of artificial ants search for good solutions. Every ant of a generation builds up a solution step by step going through several probabilistic decisions. In general, ants that found a good solution mark their paths through the decision space by putting some amount of pheromone on the edges of the path. The following ants of the next generation are attracted by the pheromone so that they will search in the solution space near good solutions. In addition to the pheromone values the ants will usually be guided by some problem specific heuristic for evaluating the possible decisions.

The algorithms proposed in (Bauer et al., 1999) and (Stützle, 1998) for the single machine total tardiness problem and the flow-shop problem respectively use a pheromone matrix $T = \{T_{ij}\}$ where pheromone is added to an element T_{ij} of the pheromone matrix when a good solution was found where job j is the ith job on the machine. The following ants of the next generation then directly use the value of T_{ij} to estimate the desirability of placing job j as the ith job on the machine when computing a new solution. We call this the *direct evaluation* (of the values in the pheromone matrix).

A different way to evaluate the pheromone matrix was proposed in (Merkle and Middendorf, 2000). Instead of using only the value of T_{ij} the ants use $\sum_{k=1}^{i} T_{kj}$ to compute the probability of placing job j as the ith on the machine. We call this the *summation evaluation* (of the values in the pheromone matrix). A difficulty using direct evaluation occurs if the ant does not choose job j as the ith job in the schedule even if τ_{ij} has

a high value. If in this case the values $T_{i+1,j}, T_{i+2,j}, \ldots$ are small then job j might be scheduled much later than at the ith place. This is in general bad for many scheduling problems and in particular when the tardiness of jobs is to be minimized or when precedence constraints might hinder other jobs to be scheduled. Using summation evaluation is likely that this will not happen.

In this paper we propose a combination of direct and summation evaluation for solving the RCPSP. We use T_{ij} respectively $\sum_{k=1}^{i} T_{kj}$ to compute the probability that activity j is the ith activity used by a serial scheduling generation scheme.

The paper is organized as follows. The resource-constrained scheduling problem is defined in Section 2. In Section 3 we describe the serial generation scheme. Our basic ant algorithm is described in Section 4. Variants of the ant algorithm are described in Section 5. The benchmark problems that were used for our tests and the parameter settings of the algorithms are described in Section 6. Experimental results are reported in Section 7. A conclusion is given in Section 8.

2 RESOURCE-CONSTRAINED SCHEDULING PROBLEM

The RCPSP is the optimization problem to schedule the activities of a project such that the makespan of the schedule is minimized while given precedence constraints between the activities are satisfied and resource requirements of the scheduled activities per time unit do not exceed given capacity constraints for the different types of resources.

Formally, $\mathcal{J} = \{0, \ldots, n+1\}$ denotes the set of activities of a project. We assume that a precedence relation is given between the activities. $\mathcal{K}$ is a set of k resource types. $\mathcal{R} = \{R_1, \ldots, R_k\}$ is the set of resource capacities where $R_i > 0$ is the constraint for resources of type $i \in [1, k]$. Every activity $j \in \mathcal{J}$ has a completion time p_j and resource requirements $r_{j,1}, \ldots, r_{j,k}$ where $r_{j,i}$ is the requirement for a resource of type i per time unit when activity j is scheduled.

Let $\mathcal{P}_j$ be the set of immediate predecessors of activity j. 0 is the only start activity, that is it has no predecessor, and $n+1$ is the only end activity, that is it has no successor. We assume that the start activity and the end activity have no resource requirements and have processing time zero. A schedule for the project is represented by the vector $(s_0, s_1, \ldots, s_{n+1})$ where s_j is the start time of activity $j \in \mathcal{J}$. If s_i is the start time of activity i then $f_i = s_i + p_i$ is its finishing time. For

a schedule the *start time* is the minimum start time $\min\{s_j \mid j \in \mathcal{J}\}$ of the activities, the *finish time* is the maximum finish time $\max\{s_j + p_j \mid j \in \mathcal{J}\}$ of the activities and the *makespan* is the difference between finish time and start time. Observe that the start time of a schedule equals s_0 and the finish time equals f_{n+1}.

A schedule is *feasible* if it satisfies the following constraints: i) activity $j \in \mathcal{J}$ must not be started before all its predecessors are finished, that is $s_j \geq s_i + p_i$ for every $s_i \in \mathcal{P}_j$, and ii) the resource constraints have to be satisfied, that is at every time unit t the sum of the resource requirements of all scheduled activities does not exceed the resource capacities, that is for every resource of type i it holds that $\sum_{s_j \in \mathcal{J}, s_j \leq t < s_j + p_j} r_{j,i} \leq R_i$.

The RCPSP problem is to find a feasible schedule with minimal makespan for a given project with resource constraints.

3 SCHEDULE GENERATION SCHEME

We used a serial schedule generation scheme (SGS) that is a standard heuristic method for RCPSP (cf. (Kolisch and Hartmann, 1999). SGS starts with a partial schedule that contains only the start activity 0 at time 0. Then SGS constructs the complete schedule in n steps where at each step one activity is added to the partial schedule constructed so far. In every step one activity j is selected from the set of available activities (that is, activities that have not been scheduled so far and where each predecessor has been scheduled). It is known that for every RCPSP instance it is possible to obtain an optimal solution by SGS (e.g. (Kolisch and Hartmann, 1999)) .

For every eligible activity j let EF_j be the maximum finishing time of all its immediate predecessors plus p_j. Let LF_j denote the latest finishing time of activity j that is calculated by backward recursion from an upper bound of the finishing time of the project (cf. (Elmaghraby, 1977)). Then the start time of activity j is the earliest time in $[EF_j - p_j, LF_j - p_j]$ such that all resource constraints are satisfied.

4 THE ANT ALGORITHM

The general idea of our ACO approach is to use an ant algorithm for deciding which activity from the set of eligible activities should be scheduled next by the SGS. The general principle of our ant algorithm is similar to an ant algorithm called AS-TSP for the traveling salesperson problem of (Dorigo, 1992; Dorigo et al.,

1996). In every generation each of m ants constructs one solution. An ant selects the activities in the order in which they will be used by the serial schedule generation scheme. For the selection of an activity the ant uses heuristic information as well as pheromone information. The heuristic information, denoted by η_{ij}, and the pheromone information, denoted by τ_{ij}, are indicators of how good it seems to schedule activity j as the ith using the SGS. The heuristic value is generated by some problem dependent heuristic whereas the pheromone information stems from former ants that have found good solutions.

The next activity is chosen according to the probability distribution over the set of eligible activities $\mathcal{D}$ determined either by direct evaluation according to

$$p_{ij} = \frac{[\tau_{ij}]^\alpha\, [\eta_{ij}]^\beta}{\sum_{h\in\mathcal{D}} [\tau_{ih}]^\alpha\, [\eta_{ih}]^\beta} \qquad (1)$$

or by summation evaluation according to

$$p_{ij} = \frac{(\sum_{k=1}^{i} [\tau_{kj}])^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{h\in\mathcal{D}}(\sum_{k=1}^{i} [\tau_{kh}])^\alpha \cdot [\eta_{ih}]^\beta} \qquad (2)$$

where α and β are constants that determine the relative influence of the pheromone values and the heuristic values on the decision of the ant. As a heuristic we use an adaption of the LFT heuristic (Davis and Patterson, 1975) that schedules activities according to growing values of LF. The relative differences between the latest finishing times are usually small for activities that become eligible late. Therefore we use the absolute differences to the maximum latest finishing time of an eligible activity as a heuristic value. In particular, the heuristic values η_{ij} are computed for the eligible activities according to

$$\eta_{ij} = \max_{k\in\mathcal{D}} LF_k - LF_j + 1$$

The best solution found so far and the best solution found in the current generation are then used to update the pheromone matrix. But before that some of the old pheromone is evaporated on all the edges according to
$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij}$$

where parameter ρ determines the evaporation rate. The reason for this is that old pheromone should not have a too strong influence on the future. Then, for every activity $j \in \mathcal{J}$ some amount of pheromone is added to element (ij) of the pheromone matrix where i is the place of activity j in the best solution found

so far. This is an elitist strategy that leads ants to search near the best found solution. The amount of pheromone added is $\frac{\rho}{2T^*}$ where T^* is the makespan of the best found schedule, that is,

$$\tau_{ij} = \tau_{ij} + \rho \cdot \frac{1}{2T^*}$$

The same is done also for the best solution found in the current generation, that is for every activity $j \in \mathcal{J}$ pheromone is added to (ij) when i is the place of activity j in the best solution found in the current generation.

The algorithm runs until some stopping criterion is met, e.g. a certain number of generations has been done or the the average quality of the solutions found by the ants of one generation has not changed for several generations.

5 ADDITIONAL FEATURES

Some additional features of our algorithm are described in this section.

5.1 COMBINATION OF DIRECT AND SUMMATION EVALUATION

Summation evaluation was introduced in (Merkle and Middendorf, 2000) and applied to the total tardiness problem. For this problem it is important not to schedule some jobs too late which is exactly what the summation evaluation enforces. For the RCPSP the situation is somewhat different. The precedence relation requires that some activities should be scheduled not too late but it is also important to schedule groups of activities at the same time that have resource requirements fitting to the constraints of the problem. Therefore, for some activity there might be several places in the sequence used by SGS which are good while other places in between are not that good. Such a behaviour can be better modelled using direct evaluation rather than summation evaluation. Therefore, instead of using only either direct evaluation or the summation evaluation we propose to use combinations of both evaluation strategies for the RCPSP. The combinations were obtained as follows. We use a parameter c that determines the relative influence of direct evaluation and summation evaluation. The probability distribution used by an ant for choosing the next activity is computed as in formula (1) but with the following "new" values τ'_{ij} that replace the values τ_{ij}

$$\tau'_{ij} := c \cdot x_i \cdot \tau_{ij} + (1 - c) \cdot y_i \cdot \sum_{k=1}^{i} \tau_{kj}$$

where $x_i := \sum_{h \in \mathcal{D}} \sum_{k=1}^{i} \tau_{kh}$ and $y_i := \sum_{h \in \mathcal{D}} \tau_{ih}$ are factors to adjust the relative influence of direct and summation evaluation. Observe, that for $c = 0$ we obtain the direct evaluation and for $c = 1$ the summation evaluation.

5.2 LOCAL OPTIMIZATION STRATEGY

The influence of a local optimization strategy that is applied to the solutions found by the ants is also studied in this paper. A 2-opt strategy was used to improve the solutions found by the ants. 2-opt considers swaps between pairs of activities in a solution. For every pair (i, j), $i < j$ of activities it is checked exactly once whether the schedule derived by SGS using the sequence where i and j are exchanged is feasible and better than the schedule derived from the old sequence. If this is the case, the new sequence is fixed for testing the remaining swaps between pairs of activities. Since the 2-opt strategy takes much computation time, we applied it only to the best solution that was found by the m ants in a generation (this approach was also used in Bauer et al. (1999)).

5.3 FORGETTING THE BEST FOUND SOLUTION

A best found solution that was stable for many generations has a great influence on the pheromone values since we apply an elitist strategy when doing the pheromone update, that is pheromone is added after every generation along the best found solution. Thus, during long runs it can happen that the algorithm converges too early to the best found solution. To come up with this problem we allowed that in every generation with some small probability $p_w > 0$ the best found solution is exchanged with the best solution in that generation, even if this solution is worse than the (old) best found solution.

6 BENCHMARK PROBLEMS AND PARAMETERS

As benchmark problems we used a set of test instances which is available in the Project Scheduling Library (PSPLIB) (Kolisch et al., 1999; Kolisch and Sprecher, 1996). From this library we used the test set j120.sm where each project consists of 120 activities. These are the largest problem instances contained in the PSPLIB.

The benchmark problems in the set j120.sm were generated by varying the following three problem parameters: network complexity (NC), resource factor (RF), and resource strength (RS). NC defines the average number of precedences per activity. RF determines the average percent of different resource types for which each activity (besides the start and the end activity) has a nonzero-demand. RS defines how scarce the resources are. A value of 0 defines the capacity of each resource to be no more than the maximum demand of all activities while a value of 1 defines the capacity of each resource to be equal to the demand imposed by the earliest start time schedule.

The set j120.sm contains 600 problem instances, each having 120 activities and 4 resource types. The set contains 10 instances for each combination of the following parameter values: $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$, and $RS \in \{0.1, 0.2, 0.4, 0.5\}$.

In the following AS-RCPSP denotes the algorithm as described in Section 4 with a value $c = 0.5$, that is direct evaluation and summation evaluation have an equal influence. AS-RCPSP with the additional 2-opt strategy (cf. Subsection 5.2) for local optimization is denoted by AS-RCPSP-LO. Notation AS-RCPSP-c, respectively AS-RCPSP-LO-c is used to explicitly denote the value of c that was used.

The parameters for the test runs are: $\alpha = \beta = 1$, $p_w = 0.01$. In every generation we used $m = 5$ ants. The combination between direct and summation evaluation were tested with values for $c \in \{0, 0.25, 0.5, 0.75, 1\}$. Since the 2-opt strategy is time consuming we applied it only at every 20th generation in algorithm AS-RCPSP-LO-c.

To be able to compare our ACO algorithms with the heuristics that have been compared in the study of (Hartmann and Kolisch, 2000) we performed tests with AS-RCPSP where the maximum number of generations was set to 1000. Since we use 5 ants per generation in every test run exactly 5000 schedules were computed - which is the same as used in the tests of (Hartmann and Kolisch, 2000). For our test runs we set $\rho = 0.02$.

We also evaluated the potential of the ACO algorithm if more evaluations of schedules are allowed. For this we tested AS-RCPSP-LO with a maximum number of 10000 generations in each run. For this longer test runs we used a ρ value of 0.01, which is slightly smaller than the value of ρ used in the short runs. A smaller ρ value often means that the algorithm tends to converge later

which is desirable for runs over more generations.

All tests have been performed on a Pentium III 500 MHz processor. One run of AS-RCPSP over 1000 generations takes about 25 sec. One run AS-RCPSP-LO over 10000 generations takes about 25 minutes.

7   EXPERIMENTAL RESULTS

The influence of the c value that determines the relative influence of direct evaluation and summation evaluation on the solution quality was tested for AS-RCPSP-c and AS-RCPSP-LO-c. Table 1 shows the average deviation of the quality of the obtained solutions from lower bounds that can be obtained from a critical path heuristic (Stinson et al., 1978) (the lower bounds were provided by Hartmann and Kolisch). For different values of c the best performance was obtained for $c = 0.5$ for both algorithms — AS-RCPSP-c and AS-RCPSP-LO-c. The average deviation was 36.65% for AS-RCPSP-0.5 after 1000 generations and 33.68% for AS-RCPSP-LO-0.5 after 10000 generations. The worst performance was obtained for $c = 1$ which corresponds to pure direct evaluation with an average deviation of 40.59% for AS-RCPSP-1 and of 36.07% for AS-RCPSP-LO-1.

We also compared the results of AS-RCPSP-c and AS-RCPSP-LO-c to the best solutions which are currently (Jan. 2000) known for the benchmark problems in j120.sm. Both algorithms found the largest number of solutions that are at least as good as the known best solutions when the parameter c was set to 0.5. AS-RCPSP and AS-RCPSP-LO found for 174, respectively 205, of the 600 test problems solutions that are at least as good as the best solutions that were known before. While AS-RCPSP could not improve any best known solutions AS-RCPSP-LO found new best solutions for 2 test problems.

The results for AS-RCPSP-LO-c after 100, 1000, and 10000 generations are compared in Figure 1. The figure also shows the influence of the local optimization strategy. AS-RCPSP-LO-c is better than AS-RCPSP-c after 1000 generations for every value of c that was tested. But it has to be noted that the computation time of AS-RCPSP-LO-c is about 6 times the computation time of AS-RCPSP for the same number of generations.

To further investigate the potential of AS-RCPSP-LO-0.5 we also tested it on the benchmark set j120.sm with the parameters $m = 20$ ants, $\rho = 0.005$ and a maximum of 20000 iterations. For this parameter values we found an average deviation from the critical path lower bounds of 32.97% (compared to 33.68% with the parameter values as described above). The average deviation from the known best solutions is only 0.02%. For 278 of the 600 test instances a solution was found that is at least as good as the known best solution and for 15 test instances improved best solution have been found.

We compared AS-RCPSP to various other heuristics for the RCPSP that are included in an extensive experimental study by (Hartmann and Kolisch, 2000). This study considers the following heuristics: i. three deterministic single/pass heuristics with regret based random sampling from (Kolisch, 1996a,b), ii. two single/pass heuristics with adaptive regret based random sampling (Kolisch and Drexl, 1996; Schirmer, 1998), iii. four genetic algorithms of (Hartmann, 1998) and (Leon and Ramamoorthy, 1995), iv. a simulated annealing algorithm of (Bouleimen and Lecocq, 1998). These heuristics were also compared with two pure random sampling methods using two different heuristics for building up a schedule. The random sampling 2 heuristic in Table 2 is a pure random sampling using SGS for schedule generation. In this study all heuristics were allowed to generate and evaluate at most 5000 schedules for each problem instance.

Table 2 compares the results from (Hartmann and Kolisch, 2000) for the 600 problem instances in j120.sm with the result of AS-RCPSP. The table shows that AS-RCPSP performed better than all the other heuristics. The second best heuristic is a genetic algorithm of (Hartmann, 1998) that performed only slightly worse than AS-RCPSP.

A recent tabu search algorithm for RCPSP not included in the study of (Hartmann and Kolisch, 2000) was proposed by (Nonobe and Ibaraki, 1999). This algorithm found in 30000 iteration steps for 219 problems in j120.sm a solution that was at least as good as the best known solution at that time (June 1999) and improved 50 solutions. One run of the algorithm took about 10 minutes on a Sun Ultra 2 (300 MHz, 1GB memory). Compared to this AS-RCPSP behaves very good. It found 174 best solutions (but compared to the improved bounds from January 2000) in about 25 seconds for one run on a Pentium III 500 Mhz processor.

To study the influence of the c-value on AS-RCPSP-c in more detail we computed the entropy of the probability distributions over all eligible activities that are considered by the ants for choosing the next activity. That is for every ith decision, $i \in [1, 120]$ of an ant during the process of constructing a solution we computed the entropy

Table 1: Comparison between AS-RCPSP-c and AS-RCPSP-LO-c for different values of c: results are averaged over all 600 problem instances from benchmark set j120.sm, comparison is done with respect to deviation from critical path lower bounds, the number of solutions found that are at least as good as the best known solutions, and number of improved best solutions (in parentheses).

c	AS-RCPSP-c 1000 generations ($\rho = 0.02$)		AS-RCPSP-LO-c 10000 generations ($\rho = 0.01$)	
	Deviation from LB in %	number of best solutions	Deviation from LB in %	number of best solutions
0	37.77	158(0)	34.06	194(2)
0.25	37.16	169(0)	33.78	193(1)
0.50	36.65	174(0)	33.68	205(2)
0.75	37.70	171(0)	34.55	199(0)
1	40.59	136(0)	36.07	160(0)

Table 2: Comparison of AS-RCPSP with different randomized heuristics (cf. (Hartmann, 1998)): results are averaged over all 600 problem instances from benchmark set j120.sm, every heuristic was allowed to construct and evaluate 5000 solutions

Algorithm	Reference	deviation from LB in %
AS-RCPSP		36.65
GA 1	(Hartmann, 1998)	36.74
SA	(Bouleimen and Lecocq, 1998)	37.68
GA 2	(Hartmann, 1998)	38.49
adaptive sampling 1	(Schirmer, 1998)	38.70
single pass/sampling 1	(Kolisch, 1996b)	38.75
single pass/sampling 2	(Kolisch, 1996a,b)	38.77
adaptive sampling 2	(Kolisch and Drexl, 1996)	40.45
GA 3	(Leon and Ramamoorthy, 1995)	40.69
single pass/sampling 3	(Kolisch, 1996b)	41.84
GA 4	(Hartmann, 1998)	42.25
random sampling 1	(Kolisch, 1995)	43.05
random sampling 2	(Kolisch, 1995)	47.61

$$e_i = -\sum_{j \in \mathcal{D}} p_{ij} \log p_{ij}$$

Figures 2 to 4 show entropy curves for different generations of ants. Each point of the curve is averaged over 300 values obtained from the 5 ants in the generation for 60 problem instances (we took the first instance of every problem type contained in j120.sm). One observation is that the entropy values corresponding to decisions in the middle of the construction process of a solution are larger than at the end or at the beginning of the process. A reason for this is that decisions in the middle have a larger set of eligible activities. Another observation is that for direct evaluation the entropy values are much smaller than for summation evaluation or the combination of direct and summation evaluation.

It is interesting that for direct evaluation the entropy for the first decisions of an ant shrinks very fast, e.g. all entropy values are below 0.2 for the first 80 decisions after generation 1500. In contrast to this, for summation evaluation the entropy values for the later decisions shrink faster than for decisions at the beginning. Only the combination of direct and summation evaluation shows for all generations nearly symmetrical curves, that is decisions at the beginning and at the end have similar entropy values. Also in this case the entropy values are higher in later generations than for pure direct or pure summation evaluation. This indicates an advantage of the combined evaluation method, it prevents the algorithm to converge too early.
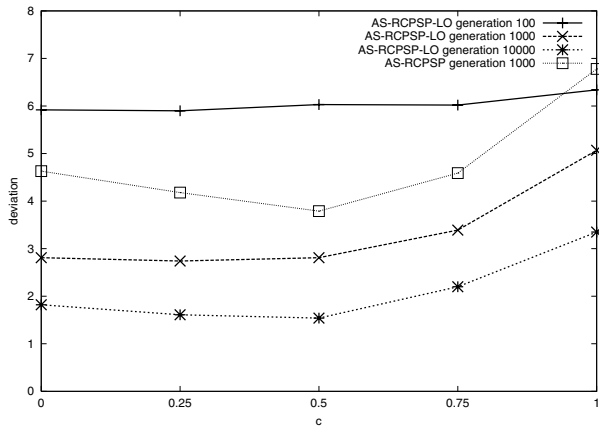
Figure 1: Percentage of deviation from the best known solutions for AS-RCPSP-LO-c and AS-RCPSP-c with different values of c: results are averaged over all 600 problem instances from benchmark set j120.sm.
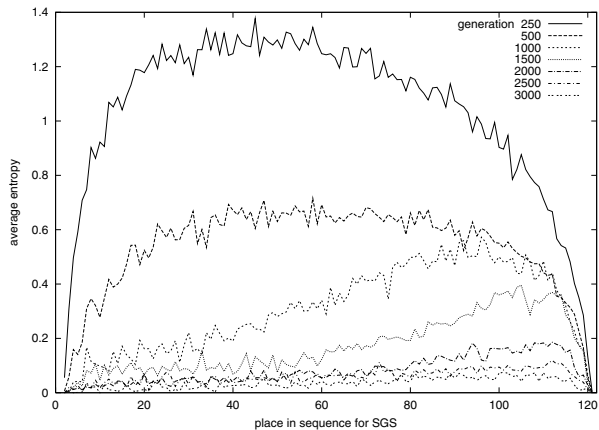

Figure 3: Entropy of the probability distribution used by the ants for AS-RCPSP-0.5: results are averaged over 5 ants per generation for 60 problem instances.

Figure 2: Entropy of the probability distribution used by the ants for AS-RCPSP-1: results are averaged over 5 ants per generation for 60 problem instances.

Figure 4: Entropy of the probability distribution used by the ants for AS-RCPSP-0: results are averaged over 5 ants per generation for 60 problem instances.

8 CONCLUSION

In this paper we have introduced an ACO approach for RCPSP. A combination of direct and summation evaluation methods is used by the ants for the construction of a new solution. We compared our approach with the results of various other randomized heuristics for the RCPSP including genetic algorithms and simulated annealing on a large set of benchmark problems. Under the constraint that every algorithm is allowed to compute and evaluate the same restricted number of solutions our algorithm performed best. Moreover, we showed that an additional 2-opt strategy leads to improved results while using more computation time. With the additional 2-opt strategy our algorithm was
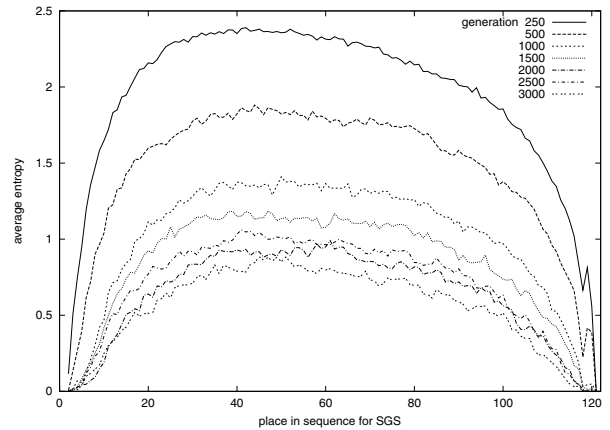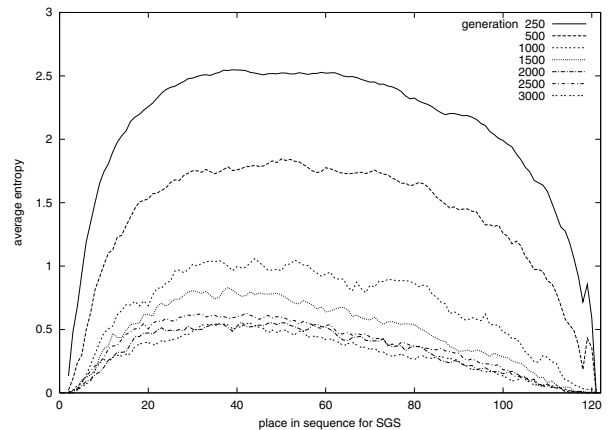
able to find new best solutions for several problems in the benchmark set.

References

Bauer, A., Bullnheimer, B., Hartl, R., and Strauss, C. (1999). An ant colony optimization approach for the single machine total tardiness problem. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), 6-9 July Washington D.C., USA*, pages 1445–1450.

Bouleimen, K. and Lecocq, H. (1998). A new efficient simulated annealing algorithm for the resource constrained project scheduling problem. *European Journal of Operational Research*. to appear.

Brucker, P., Drexel, A., Möhring, R., Neumann, K.,

and Pesch, E. (1999). Resource-constraint project scheduling: Notation, classification, models, and methods. *European Journal of Operations Research*, 112:3–41.

Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M. (1994). Ant system for job-shop scheduling. *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, 34:39–53.

Davis, E. W. and Patterson, J. H. (1975). A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, 21(8):944–955.

den Besten, M., Stützle, T., and Dorigo, M. (1999). Scheduling single machines by ants. Technical Report IRIDIA/99-16, IRIDIA, Université Libre de Bruxelles, Belgium.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica , Politecnico di Milano, Italy. pp. 140.

Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Systems, Man, and Cybernetics – Part B*, 26:29–41.

Elmaghraby, S. E. (1977). *Activity networks*. Wiley Interscience.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7):733–750.

Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*. to appear.

Kolisch, R. (1995). *Project Scheduling under Resource Constraints*. Production and Logistics. Physica-Verlag.

Kolisch, R. (1996a). Effcient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14(3):179–192.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333.

Kolisch, R. and Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics*, 43(1):23–40.

Kolisch, R. and Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In Weglarz, J., editor, *Handbook on Recent Advances in Project Scheduling*, pages 197–212. Kluwer, Amsterdam.

Kolisch, R., Schwindt, C., and Sprecher, A. (1999). Benchmark instances for project scheduling problems. In Weglarz, J., editor, *Handbook on Recent Advances in Project Scheduling*, pages 147–178. Kluwer, Amsterdam.

Kolisch, R. and Sprecher, A. (1996). PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216. library accessible under `http://www.bwl.uni-kiel.de /Prod/psplib/index.html`.

Leon, V. and Ramamoorthy, B. (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *OR Spektrum*, 17:173–182.

Merkle, D. and Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In *Proceeding of the EvoWorkshops 2000*, number 1803 in Lecture Notes in Computer Science, pages 287–296. Springer Verlag.

Nonobe, K. and Ibaraki, T. (1999). Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). Technical report, Department of Applied Mathematics and Physics, Kyoto University, Japan.

Schirmer, A. (1998). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics*. submitted.

Stinson, J., Davis, E., and Khumawala, B. (1978). Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions*, 10:252–259.

Stützle, T. (1998). An ant approach for the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT '98)*, volume 3, pages 1560–1564. Verlag Mainz, Aachen.

van der Zwaan, S. and Marques, C. (1999). Ant colony optimisation for job shop scheduling. In *Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 99)*.