Minimizing Total Tardiness on a Single Machine Using Ant Colony Optimization

Andreas Bauer
Bernd Bullnheimer
Richard F. Hartl
Christine Strauss

Department of Management Science
University of Vienna
Bruenner Strasse 72
A-1210 Vienna
christine.strauss@univie.ac.at

Abstract

Ant Colony Optimization is a relatively new meta-heuristic that has proven its quality and versatility on various combinatorial optimization problems such as the traveling salesman problem, the vehicle routing problem and the job shop scheduling problem. The paper introduces an Ant Colony Optimization approach to solve the problem of determining a job-sequence that minimizes the overall tardiness for a given set of jobs to be processed on a single, continuously available machine, the Single Machine Total Tardiness Problem. We experiment with various heuristic information as well as with variants for local search. Experiments with 250 benchmark problems with 50 and 100 jobs illustrate that Ant Colony Optimization is an adequate method to tackle the SMTTP.

Keywords: ant colony methods, machine scheduling, meta-heuristic

1 Introduction

Ant Colony Optimization (ACO) is a meta-heuristic that dates back to the early nineties and was first introduced by Dorigo et al. (cf. [8, 9, 14, 18]). Since then ACO has been applied to a variety of classic combinatorial optimization problems and has generated very good results (cf. e.g. [5, 6, 10, 11, 19, 24, 29, 38]). In recent work simplifications and generalizations of the original approach are performed [15, 16].

Ant Colony Optimization models a nature-based, multi-agent process in order to solve hard combinatorial optimization problems. Real ants searching for food are capable to find the shortest path between their nest and a food source without strength of vision but by exchanging information via pheromones. Varying quantities of pheromone which are laid down by each ant on the path taken indicate the distance and quality of the food source. As other ants observe the pheromone trail and are attracted to follow it, the path is marked again, reinforced and will attract even more ants to follow the trail. Roughly speaking, pheromone trails leading to nearby, rich food sources will be more frequented and therefore grow faster than trails leading to remote low-quality food sources.

The above described behavioural pattern of real ant colonies was the inspiration for this solution procedure for combinatorial optimization problems and was the impetus to create artificial ants for simulation purposes. ACO is based on the following analogies: artificial ants scan the solution space while real ants search their natural environment for food; the objective function values represent a mapping of the food source's quality and an adaptive memory is the equivalent of the pheromone trails. Artificial ants are equiped with a heuristic function in order to support their search through the set of feasible solutions. (For a more detailed introduction of the Ant Colony Optimization Meta-Heuristic the reader is referred to [15, 19].)

Gutjahr [25] and more recently Dorigo [15] developed a general and formal representation of the ant algorithm: the generalization of ants are agents who perform *random walks* in a *construction graph*. Each random walk represents a feasible solution of the given combinatorial optimization problem. A walk is a chain of concatenated "node-to-node" moves, and each move is performed on the basis of *transition probabilities*. Gutjahr [25] shows that the solutions of this generalized Ant System converge to the optimal solution of the given problem instance with a probability that can be made arbitrarily close to one.

In this paper we apply ACO to the Single Machine Total Tardiness Problem (SMTTP). This well-known problem is NP-hard [20] and has been tackled by several exact solution processes (cf. [13, 22, 27, 32, 37, 40]) as well as heuristics (cf. [23, 26, 31, 34, 41]).

The study presented shows that ACO can solve SMTTPs satisfactorily and the solution quality can compete with that of other leading heuristics. Testing various heuristic information such as Earliest Due Date (EDD), Modified Due Date (MDD), Look-ahead-MDD (L-MDD) and Shortest Processing Time (SPT), we found MDD to be the most efficient one. The choice of an appropriate local search procedure was based on tests with various interchange variants: pairwise swaps of adjacent jobs and pairwise swaps of any two jobs, so-called transpositions. Further tests focused on the sequence on which the local search should be applied. These mentioned design alternatives were first tested on an example with 10 jobs from literature [26] before the most promising variant was chosen and test runs with 250 problem instances with 50 and 100 jobs were performed and evaluated.

The remainder of this paper is organized as follows: In Section 2 we outline the problem and relevant literature. In Section 3 we present an exact as well as a simplified construction graph for solving the Single Machine Total Tardiness Problem using artificial ants. Section 4 describes in detail the adaption of the ACO algorithm: various alternatives of heuristic information, trail up-date and local search are outlined. Furthermore, computational results are presented and compared with other leading heuristics. Finally we give an outlook on ongoing research. We conclude with a brief summary in Section 5.

2 The Single Machine Total Tardiness Problem

Machine scheduling is a central task in operations and production planning. The main focus is on the efficient allocation of resources to activities over time. In this paper we consider scheduling job operations for sequential processing on a single, continuously available machine such that the total tardiness of all jobs is minimized.

2.1 The Problem

The machine scheduling problem we consider in the following, can be described as follows. There is one single machine, which is used to process n jobs. Scheduling a set of n jobs that are to be processed on a single machine minimizing total tardiness is known as the Single Machine Total Tardiness Problem (SMTTP). A schedule specifies for each job j a time interval throughout which processing is performed on job j. A feasible schedule minimizing the total tardiness for n given jobs is subject to the constraints that no pre-emption of jobs is allowed. The machine can only handle one job at a time, and every job j ($j = 1, \ldots, n$) is available for processing at time zero and requires a positive processing time p_j. Further problem relevant data associated with each job j is the job's due date d_j. The job's completion time C_j and the job's tardiness T_j, where $T_j = max\{0,\ C_j - d_j\}$ are a result of sequencing the jobs. The objective of the SMTTP is to find a feasible processing sequence (out of all possible permutations of n jobs) that includes all jobs exactly once and that minimizes the total tardiness $\sum_{j=1}^{n} T_j$. The SMTTP represents a special case of the generalized total weighted tardiness problem [34] where positive weights are specified for each job. For related problems see [1].

2.2 Solution Methods

The complexity of the Single Machine Total Tardiness Problem remained open for a long time. In 1977 Lawler [27] showed that the SMTTP is pseudopolynomially solvable by an algorithm that is based on decomposition. Only in 1990 Du et al. [20] showed that the problem is NP-hard, which suggests that it is impossible to find an optimal solution without the use of an essentially enumerative algorithm, for which computation times will increase exponentially with problem size.

Optimization techniques, like branch and bound approaches as well as dynamic programming algorithms, were applied to the SMTTP, whereas enumerative algorithms were applied to the general total weighted tardiness problem. Emmons [21] developed several theorems and dominance rules that can be used to restrict the search effort of a branch and bound algorithm. Based on Emmons dominance rules Fisher developed a dual variable Lagrangian problem [22]. Schrage and Baker [37], and Lawler [27] applied a dynamic programming approach to the SMTTP, whereas Potts and Van Wassenhove [32] combined Lawlers decomposition theorem with the approach of Schrage and Baker to implement an efficient algorithm. More recently, Szwarc and Mukhopadhyay [40], and Della Croce et al. [13] presented branch and bound procedures, that are based on findings of Emmons and Lawler.

In practice it may be acceptable - if not necessary - to use a heuristic method incorporating structural problem-specific properties to find an approximate solution for the NP-hard SMTTP. Clearly there is a trade-off between the computational effort in order to obtain a feasible solution and the quality of that solution in terms of objective function value. Wilkerson and Irwin [41] use transposition to improve a feasible basic solution and use the so-called adjacent pair interchange (API) as a core mechanism of their WI heuristic. Fry et al. [23] developed an enhanced WI heuristic by selecting the best out of nine adjacent pair interchange strategies. Applying a neighborhood strategy that is not limited to pairwise swaps Holsenback and Russell [26] developed a heuristic that uses the net benefit of relocation (NBR) as heuristic criterion which is based on Emmons dominance theorems as well. Panwalkar et al. [31] developed the PSK heuristic that seems to be inferior to the NBR heuristic (cf. [36], p. 543).

3 Graphical Representation of the SMTTP

To apply Ant Colony Optimization a graphical representation of the problem, the so-called construction graph mentioned above, is needed. The construction graph for the SMTTP consists of nodes representing different *states*, and arcs representing *transitions*. Ants move from one node (state) to another using the connecting arcs (transitions) and laying down their pheromone trails on these arcs. For some combinatorial optimization problems, like e.g., the traveling salesman problem (TSP), this construction graph is obviously corresponding or even identical to the underlying graph. This is not the case for the SMTTP.

3.1 Exact Construction Graph

The SMTTP may be represented by an acyclic, directed graph with exactly one source node and exactly one sink node. It is sufficient to have only one node representing an unordered subset of jobs already processed because the tardiness of the job scheduled next is independent of their specific sequence. Through each directed arc another job (to be processed next) is added to the list of jobs. In short, nodes are representing a set of jobs, while arcs are the representation of adding a specific job to this set.

The source node is on the first level and represents the beginning of the sequence where no job has been processed yet; the corresponding set is empty. Given the total amount of n jobs the source node has n outgoing arcs, each arc describing the completion of the job it represents.

On the second level each of the n nodes has one incoming and $n-1$ outgoing arcs. On the third level each node has two incoming and $n-2$ outgoing arcs, and so on. The sink node represents the set of all jobs. It has n incoming nodes and no outgoing nodes. So the graph consists of $n+1$ levels with n transitions. In each transition exactly one job is added. Any path from the source node to the sink node represents a feasible solution; the optimal sequence of jobs is the path that minimizes the total tardiness of all jobs. Figure 1 shows examplarily an exact graphical representation of a problem with three jobs to be scheduled.

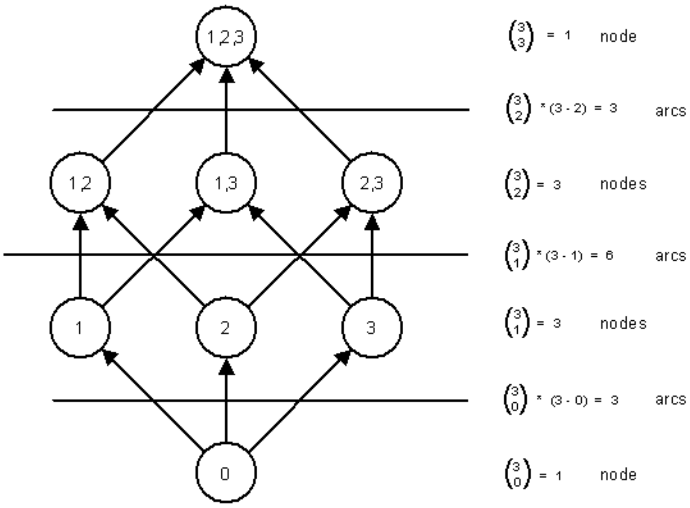

Figure 1: Exact Construction Graph (3 jobs)

In general terms a construction graph with n jobs consists of $n+1$ levels. On level i (where $i = 0, \ldots, n$) there exist $\binom{n}{i}$ nodes. The transition from level i to level $i+1$ consists of $\binom{n}{i} \cdot (n-i)$ arcs. Figure 2 shows the generalized version of an exact graph with n jobs.

For a problem with n jobs the number of arcs is $\sum_{i=0}^{n-1} [\binom{n}{i} \cdot (n-i)] = 2^n \cdot \frac{n}{2}$ and the number of nodes is $\sum_{i=0}^{n} \binom{n}{i} = 2^n$. A simplified representation is used in the following because of the exponential relationship between the number of jobs and the number of arcs.

3.2 Simplified Construction Graph

As the exact representation of the SMTTP implies a construction graph of high complexity (each transition is represented by a unique arc) a simplified construction graph with a reduced complexity would be helpful. To reduce the complexity of the construction graph some (historical) information is neglected, and a move represents the decision to schedule job j as i-th job in the sequence, no matter which jobs have been scheduled previously. Thus, we use only two dimensional matrices to represent the desirability to choose job j to be processed as the i-th job, and to store the pheromone trails.

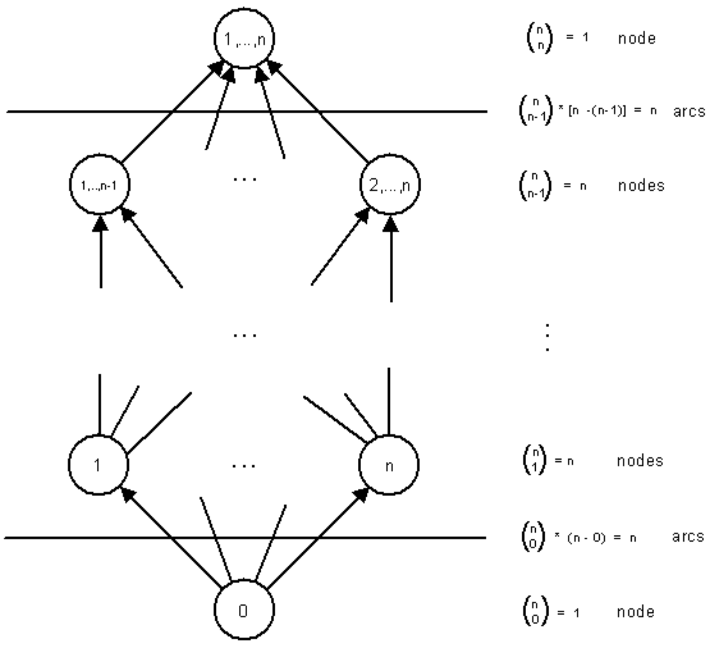

Figure 2: Exact Construction Graph (n jobs)

Thus, in the simplified graph each step looks like the first step in the exact representation (cf. Figure 3).

The simplified representation is of quadratic complexity only. Because of this simplification the position of a job in the sequence is considered to be independent of the sequence of jobs already scheduled. Some information is neglected because the tardiness of a job does not depend on the number of predecessors but on their total processing time. In this simplified version there are arcs which are not permitted: although a job is to be processed exactly once, in each step there is an arc leading to each job (n arcs). That is the reason why ants need to memorize the jobs already scheduled. The exact representation would not require such memory as this information is inherent in the graph itself.

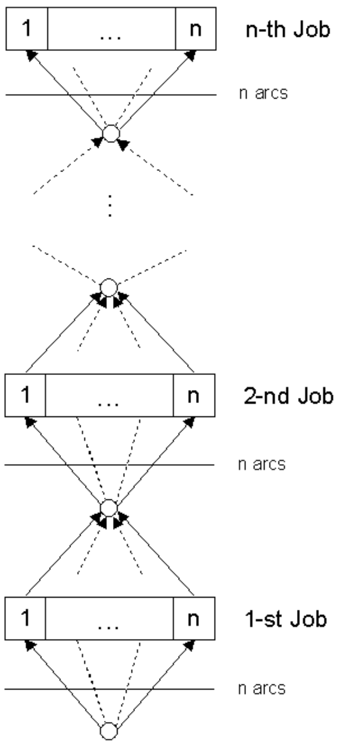

Figure 3: Simplified Representation

4 ACO solving the SMTTP

As the Ant Colony Optimization Meta-Heuristic has proven good solution quality for NP-hard combinatorial optimization problems, the application to the SMTTP seemed reasonable and promising. In a feasible solution each job appears exactly once in the processing sequence, in an optimal solution the total tardiness is minimal. In the following we will first describe the process of solution generation, followed by a description of different strategies to update pheromone trails, and finally adequate local search strategies are discussed.

4.1 Heuristic Information

Each artificial ant iteratively and independently decides which job to append to the sub-sequence generated so far until all jobs are scheduled. Each ant generates a complete solution by selecting a job j to be on the i-th position of the sequence. This selection process is influenced through problem-specific heuristic information, called visibility and denoted by η_{ij}, as well as pheromone trails, denoted by τ_{ij}. The former is an indicator of how good the choice of that job *seems to be*, and the latter indicates how good the choice of that job *was* in former runs. Both matrices are only two dimensional as a consequence of the reduction in complexity described in Section 3.2. The transition probability $\mathcal{P}_{ij}$ that job j be selected to be processed on position i in the sequence is formally given by:

$$\mathcal{P}_{ij} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h\in\Omega} [\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} & \text{if } j \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where Ω is the set of non-scheduled jobs, α determines the relative influence of the pheromone trails and β regulates the relative influence of the visibility.

The most obvious heuristic information to be used is based on the Earliest Due Date (EDD) heuristic, which sorts and schedules jobs according to ascending due dates. If EDD is applied the visibility would read as follows:

$$\text{EDD} \qquad \eta_{ij} = \frac{1}{d_j} \tag{2}$$

The so-called Modified Due Date (MDD) heuristic represents an enhancement of the EDD and is based on Theorem 1 by Emmons [21]. In this case, the jobs are scheduled iteratively. After a job is scheduled, all remaining non-scheduled jobs are again sorted in ascending order, but according to modified due dates. The job yielding the lowest modified due date is appended to the sub-sequence generated so far. Modified due dates are given by $max\{\mathcal{T} + p_j,\ d_j\}$, where $\mathcal{T}$ is the total processing time of all jobs already scheduled. Thus, for an ant colony optimization approach, this MDD heuristic value can formally be written as:

$$\text{MDD} \qquad \eta_{ij} = \frac{1}{max\{\mathcal{T} + p_j,\ d_j\}} \tag{3}$$

Furthermore, MDD may be enhanced to Look-ahead MDD (L-MDD). This heuristic is more complex than EDD and MDD. Again, for each not yet scheduled job j, the total tardines $Tard_j$ is calculated, assuming that all remaining non-scheduled jobs are scheduled according to MDD. Then, the job with the lowest total tardiness is chosen. The corresponding expression for the heuristic information in an ACO algorithm is:

$$\text{L-MDD} \qquad \eta_{ij} = \frac{1}{Tard_j} \tag{4}$$

Obviously, using L-MDD as the heuristic information is computationally more expensive than EDD or MDD.

Another alternative incorporating heuristic information is the Shortest Processing Time (SPT) rule which schedules jobs according to ascending processing times. In formal terms the corresponding expression reads:

$$\text{SPT} \qquad \eta_{ij} = \frac{1}{p_j} \tag{5}$$

SPT generates good solutions in the event that most of the jobs are delayed and even optimal solutions if all jobs are delayed.

4.2 Trail update

After each ant has performed the selection process according to the procedure described above and each ant has generated a feasible sequence, the pheromone trails are globally updated. Good solutions are characterized by relatively low total tardiness and emphasize those sub-sequences which are part of a good solution; whereas those sub-sequences that are part of poor solutions will be only slightly marked. At the same time, evaporation reduces the pheromone trails evenly. The global trail update may formally be written as follows:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t) \tag{6}$$

where the parameter $\rho \in [0,1]$ controls the pheromone decay, and $\Delta\tau_{ij}$ is the amount of pheromone trail added to τ_{ij} by the ants. Analogous to the heuristic information, there are various possibilities of pheromone decay $\Delta\tau_{ij}$ in ACO algorithms. In early versions [19], each ant contributed to the trail update, whereas later versions suggested a ranking of solutions concerning their quality and let only some best ants contribute to the pheromone trails [7]. Other approaches restricted the update to the best ant [17], possibly combined with an upper and lower bound for the trail levels [39]. In this paper we follow the ant colony system (ACS) idea, i.e., only the best solution contributes to the pheromone trail update. Thus, we have $\Delta\tau_{ij}(t) = 1/T^*$

for all edges (i, j) belonging to the best solution found so far, where T^* is the total tardiness of that best solution.

Besides the global update described above, also a local trail update is performed in ACS. After an artificial ant has selected a job to be appended to the existing sub-sequence, the corresponding pheromone trail is updated as follows:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0 \tag{7}$$

with the initial trail intensity

$$\tau_0 = \frac{1}{n \cdot T_{EDD}}$$

where T_{EDD} is the total tardiness for a processing sequence generated according to the EDD rule.

4.3 Local Search

In several other applications ACO algorithms have worked very well if a local search procedure is integrated [5, 6, 17, 39]. Such a local search procedure is applied to the solutions generated by the artificial ants before updating the pheromone trails. We considered pairwise swapping and transposition as two local search alternatives to be part of the ACO-approach for the SMTTP[1]. Both strategies are similar to a 2-opt strategy. Transposition means swapping two adjacent jobs having a predecessor-successor-relation whereas pairwise swapping may be applied to any two jobs in the set of scheduled jobs. Transposition is based on the Adjacent Pairwise Interchange Strategy (API) of Wilkerson and Irwin [41] and is less costly regarding computational effort than swapping.

One of the two strategies described above may be applied either to all sequences generated during an iteration or to the best sequence of the current iteration (iteration-best-method). The former alternative seems to be more promising but represents also the more expensive alternative in terms of computation time. We will address this topic in the next section, when we present our computational results.

[1]Further possible local search procedures to improve a schedule that was constructed by the artificial ants would be NBR and PSK.

4.4 Computational Results

The computational results we present in this section were generated using 250 benchmark problems [34]: 125 instances contain 50 jobs and 125 instances contain 100 jobs[2] from Potts and Van Wassenhove. These problems were generated as follows: for each job j ($j = 1, \ldots, n$) an integer processing time p_j from the uniform distribution [1, 100] and an integer due date d_j from the uniform distribution

$$\left[\sum_{j=1}^{n} p_j \cdot (1 - \mathrm{TF} - \frac{\mathrm{RDD}}{2}), \quad \sum_{j=1}^{n} p_j \cdot (1 - \mathrm{TF} + \frac{\mathrm{RDD}}{2})\right]$$

are generated. The hardness of a problem depends on the tardiness factor, denoted by TF and the relative range of due dates, denoted by RDD. Both, the values for TF and RDD are taken from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. The problem set contains five instances of size $n = 50$ and five instances of size $n = 100$ for each combination of TF and RDD, i.e., 125 instances for 50-jobs-problems and 125 instances for 100-jobs-problems in total.

As mentioned in Section 4.2 we implemented ACS, a more recent ACO algorithm that uses a slightly altered transition rule as opposed to the general one presented in equation (1) in Section 4.1. Here an artificial ant selects job j to be processed on position i as follows:

$$j = \begin{cases} \arg\max_{h \in \Omega} \left\{ [\tau_{ih}]^{\alpha} [\eta_{ih}]^{\beta} \right\} & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{8}$$

where $q_0 \in [0, 1]$ is a tunable parameter, q is a random variable, uniformly distributed over [0,1], and J is a job selected according to (1). In other words, for $q \leq q_0$ the "best" job in terms of trail and visibility is *exploited*, whereas for $q > q_0$, a "good" job is *explored*.

In our tests we used MDD for the heuristic information η_{ij} although tests with problem size $n = 50$ showed that L-MDD yields slightly better. As a very modest improvement of the results caused an increase of the run times by a factor of approximately 30, we refrained from applying L-MDD. Furthermore, we used 20 artificial ants and the following parameter settings: $\alpha = 1$, $\beta = 2$, $q_0 = 0.9$ and $\rho = 0.1$. Preliminary tests showed that applying swaps as local search procedure to each solution generated by an ant yields the best results, but is computationally very expensive. For that reason we used the less costly iteration-best variant, i.e., the local search procedure was

[2] We would like to thank H.A.J. Crauwels for providing the benchmark problems as well as their optimal solutions.

applied only to the best solution of each iteration. In Table 1 we compare the results of the ACO approach with two other heuristic approaches and show in detail results obtained applying NBR and PSK to problems of size $n = 50$ ([36], pp. 541). We report the best solution found as well as the result obtained for five ACO test runs for each instance, leading to 625 runs in total. Our ant algorithm generated optimal solutions for 124 out of 125 problem instances. The 609 optimal solutions (out of 625) and the low average deviation of 0.0060% may be interpreted as an indicator of the robustness of the approach.

		NBR (100)		PSK (100)		ACO (125)		ACO (625)	
TF	RDD	opt.	dev.(%)	opt.	dev.(%)	opt.	dev.(%)	opt.	dev.(%)
0.2	0.2	5	0.00	3	3.48	5	0.00	21	0.0590
0.2	0.4	5	0.00	5	0.00	5	0.00	25	0.0000
0.2	0.6	5	0.00	5	0.00	5	0.00	21	0.0090
0.2	0.8	5	0.00	5	0.00	5	0.00	25	0.0000
0.2	1.0	5	0.00	5	0.00	5	0.00	25	0.0000
0.4	0.2	1	2.10	0	5.11	5	0.00	25	0.0000
0.4	0.4	0	2.12	0	2.12	4	0.01	20	0.0802
0.4	0.6	1	0.72	1	1.36	5	0.00	25	0.0000
0.4	0.8	4	1.58	3	1.58	5	0.00	25	0.0000
0.4	1.0	5	0.00	5	0.00	5	0.00	25	0.0000
0.6	0.2	1	0.69	0	4.26	5	0.00	25	0.0000
0.6	0.4	1	1.75	0	2.84	5	0.00	25	0.0000
0.6	0.6	0	2.00	0	3.62	5	0.00	25	0.0000
0.6	0.8	1	1.20	1	1.64	5	0.00	25	0.0000
0.6	1.0	1	2.83	0	1.88	5	0.00	25	0.0000
0.8	0.2	0	0.26	0	3.83	5	0.00	25	0.0000
0.8	0.4	1	0.12	0	0.46	5	0.00	25	0.0000
0.8	0.6	1	0.30	2	0.19	5	0.00	25	0.0000
0.8	0.8	1	0.91	2	0.08	5	0.00	22	0.0019
0.8	1.0	1	0.70	3	0.01	5	0.00	25	0.0000
1.0	0.2	n/a	n/a	n/a	n/a	5	0.00	25	0.0000
1.0	0.4	n/a	n/a	n/a	n/a	5	0.00	25	0.0000
1.0	0.6	n/a	n/a	n/a	n/a	5	0.00	25	0.0000
1.0	0.8	n/a	n/a	n/a	n/a	5	0.00	25	0.0000
1.0	1.0	n/a	n/a	n/a	n/a	5	0.00	25	0.0000
Ø		44/100	0.86	40/100	1.62	124/125	0.0004	609/625	0.0060

Table 1: Detailed comparison of ACO with NBR and PKS ($n = 50$)

Table 2 shows a comparison on the basis of problem size $n = 100$. Aggregated results of ACO are compared with reported results in literature generated by 13 different leading heuristics. All problem instances were generated using the method by Potts and Van Wassenhove [34] described above but the problem instances were not identical. From [34] we took the results of the following procedures: Wilkerson-Irwin (WI), four decomposition heuristics (DEC/EDD, DEC/WI/S, DEC/WI/D, DEC/RS500), two descent methods (DES and DESO), and two simulated annealing approaches (SALM and SAPW). Furthermore, we took the results for NBR, modified NBR (M-NBR), PSK and composite M-NBR/PSK from [35][3]. We characterize the results for each procedure by the size of the data set, the number of optimal solutions found and the mean relative deviation from the optimum[4]. Russell and Holsenback [35] performed their experiments on a reduced set of 85 instances[5] containing 100 jobs each.

Only our ACO algorithm was able to solve all 125 test instances optimally. The M-NBR/PSK composite heuristic, which applies M-NBR as well as PSK and selects the better of the two solutions obtained, performs well in terms of deviation, but solves only 31 out of 85 instances optimally. Regarding the other criteria, the number of optimal solutions, DEC/WI/D comes closest (98 out of 125), but is still outperformed by our ant algorithm.

A simulated annealing approach, developed by Ben-Daya and Al-Fawzan [4] was applied to selected problem instances (100 jobs) with a tardiness factor TF=0.6 and a range of due dates RDD $\in \{0.4, 0.8\}$. Their approach yielded an average deviation of 0.75% only (cf. [4], p. 65) whereas ACO has an average deviation of 0.00%.

4.5 Possible Extensions

As mentioned in Section 2.2 the generalization of the SMTTP is the Single Machine Weighted Total Tardiness Problem [34]. The goal is to find a processing order of the given jobs that minimizes $\sum_{j=1}^{n} w_j \cdot T_j$, i.e. with minimum total weighted tardiness, where w_j is the weight of job j. Lawler [27] and Lenstra et al. [28] showed that the generalization of the SMTTP, the Single Machine Weighted Total Tardiness Problem, is strongly NP-hard. Potts and Van Wassenhove [33] use a branch and bound algorithm, and Aktuerk and Yildirim [2] present a lower bounding scheme whereas Crauwels et al. [12] propose several local search procedures to solve this problem. Matsuo et

[3] See [34, 35] for details concerning these procedures.

[4] We do not report actual computation times as our implementation was not meant for high-speed computing but for maximum flexibility. Based on our experience we estimate computation times of 10 seconds for an efficient PC-implementation. This means that our ACO algorithm is not as fast as the other heuristics, but this drawback is more than compensated by the better results.

[5] For TF=1.0 no instances at all, and for TF=0.2 only 10 instances with RDD $\in \{0.2, 0.4\}$ were generated.

Procedure	Data set size	Number of optimal solutions	Mean relative deviation
WI[a]	125	54	1.28%
DEC/EDD[a]	125	44	4.03%
DEC/WI/S[a]	125	61	0.59%
DEC/WI/D[a]	125	98	0.16%
DEC/RS500[a]	125	78	0.15%
DES[a]	125	43	0.19%
DESO[a]	125	80	0.20%
SALM[a]	125	23	0.84%
SAPW[a]	125	37	0.73%
NBR[b]	85	16	1.91%
M-NBR[b]	85	24	1.21%
PSK[b]	85	22	2.04%
M-NBR/PSK[b]	85	31	0.0075%
ACO	125	125	0.0000%

[a] Potts and Van Wassenhove (1991)
[b] Russel and Holsenback (1997a)

Table 2: Comparison of ACO with leading heuristics ($n = 100$)

al. [30] use Simulated Annealing to solve the weighted SMTTP and find that a systematic search of the neighborhood outperforms a random search.

In current research we apply ACO to the Single Machine Weighted Total Tardiness Problem. The weights are incorporated into the visibility function, e.g., if the EDD rule is applied the weighted problem would formally read as: $\eta_{ij}^{w} = \frac{w_j}{d_j}$. We expect results of comparable quality like we obtained for the SMTTP.

In a further research focus in the field of machine scheduling we apply ACO to problems where a set of jobs with sequence dependent setup times has to be processed on parallel machines. The aim is to find a schedule with minimal makespan.

5 Conclusion

Ant Colony Optimization (ACO) is a novel meta-heuristic and multi-agent approach that has shown good results in various combinatorial optimization applications, such as traveling salesman, vehicle routing, graph coloring, quadratic assignment, flow and job shop scheduling. In this paper we present a further ACO application and adapt ACO to solve a well-known machine scheduling problem: the Single Machine Total Tardiness Problem.

We integrated Modified Due Date (MDD), a problem-specific procedure, as the heuristic criteria and performed a pairwise swap as a local search procedure on an iteration-best basis. Our adaptation of the ACO algorithm yielded very good results for large problem sizes and outperformed all leading heuristics.

Acknowledgements

We would like to thank Herbert Dawid, Marco Dorigo and Walter Gutjahr for their valuable comments on this research.

References

[1] Anderson, E.J., C.A. Glass and C.N. Potts (1997): Machine scheduling, in: Aarts E. and Lenstra J.K., eds., Local Search in Combinatorial Optimization, Chichester, Wiley, pp. 361-414.

[2] Aktuerk M.S. and M. B. Yildirim(1998): A New Lower Bounding Scheme for the Total Weighted Tardiness Problem, *Computer and Operations Research* **25** (4) pp. 265-278.

[3] Bauer, A. (1998): *Ant Colony Optimization for the Single Machine Total Tardiness Problem*, unpublished Master Thesis (in german), Department of Management Science, University of Vienna, Austria.

[4] Ben-Daya, M. and M. Al-Fawzan (1996): A Simulated Annealing Approach for the One-Machine Mean Tardiness Scheduling Problem, *European Journal of Operational Research* **93** pp. 61-67.

[5] Bullnheimer, B., R.F. Hartl and C. Strauss (1997): *An Improved Ant System Algorithm for the Vehicle Routing Problem*, POM Working Paper No. 10/97, University of Vienna. To appear in *Annals of Operations Research*.

[6] Bullnheimer, B., R.F. Hartl and C. Strauss (1998): *Applying the Ant System to the Vehicle Routing Problem*, in: S. Voss, S. Martello, I. H. Osman and C. Roucairol, eds, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston, pp. 109-120.

[7] Bullnheimer, B., R.F. Hartl and C. Strauss (1999): *A New Rank Based Version of the Ant System - A Computational Study*, *Central European Journal of Operations Research* **7** (1) pp. 25-38.

[8] Colorni, A., M. Dorigo and V. Maniezzo (1991): *Distributed Optimization by Ant Colonies*, Proceedings of European Conference on Artificial Life ECAL'91, Paris, France, Elsevier Publishing, pp. 134-142.

[9] Colorni, A., M. Dorigo and V. Maniezzo (1992): *Investigation of Some Properties of an Ant Algorithm*, Proceedings of Parallel Problem Solving from Nature, Brussels, Elsevier, pp. 509-520.

[10] Colorni, A., M. Dorigo, V. Maniezzo and M. Trubian (1994): Ant System for Job-Shop Scheduling, *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science* **34** (1) pp. 39-53.

[11] Costa, D. and A. Hertz (1997): Ants can Colour Graphs, *Journal of the Operational Research Society* **48** pp. 295-305.

[12] Crauwels, H.A.J., C.N. Potts and L.N. Van Wassenhove (1998): Local search heuristics for the single machine total weighted tardiness scheduling problem, *Informs Journal on Computing* **10** pp. 341-350.

[13] Della Croce, F., R. Tadei, P. Baracco and A. Grosso (1998): A New Decomposition Approach for the Single Machine Total Tardiness Scheduling Problem, *Journal of the Operational Research Society* **49** pp. 1101-1106.

[14] Dorigo, M. (1992): *Optimization, Learning and Natural Algorithms*, Ph.D. Dissertation (in italian), Politecnico di Milano, Milano, Italy.

[15] Dorigo, M. and G. Di Caro (1999): *The Ant Colony Optimization Meta-Heuristic*, in: D. Corne, M. Dorigo and F. Glover, eds., New Ideas in Optimization, McGraw-Hill (to appear).

[16] Dorigo M., G. Di Caro and L.M. Gambardella (1998): *Ant Algorithms for Discrete Optimization*, Technical Report IRIDIA/98-10, Universite Libre de Bruxelles, Belgium. To appear in *Artificial Life*.

[17] Dorigo, M. and L.M. Gambardella (1997): Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation* **1** pp. 53-66.

[18] Dorigo, M., V. Maniezzo and A. Colorni (1991): *Positive Feedback as a Search Strategy*, Technical Report 91-016, Dip. Elettronica, Politecnico di Milano, Italy.

[19] Dorigo, M., V. Maniezzo and A. Colorni (1996): Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics* **26** (1) pp. 29-41.

[20] Du, J. and J.Y.-T. Leung (1990): Minimizing Total Tardiness on One Machine is NP-Hard, *Mathematics of Operations Research* **15** pp. 483-495.

[21] Emmons, H. (1969): One Machine Sequencing to Minimize Certain Functions of Tardiness, *Operations Research* **17** pp. 701-715.

[22] Fisher, M.L. (1976): A Dual Algorithm for the One-Machine Scheduling Problem, *Mathematical Programming* **11** pp. 229-251.

[23] Fry, T.D., L. Vicens, K. Macleod and S. Fernandez (1989): A Heuristic Solution Procedure to Minimize $\bar{T}$ on a Single Machine, *Journal of the Operational Research Society* **40** pp. 293-297.

[24] Gambardella, L.M. and M. Dorigo (1997): *HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem*, Technical Report IDSIA/11-97, IDSIA, Lugano, Switzerland.

[25] Gutjahr, W.J. (1998): *A Generalized Ant System and its Convergence*, Technical Report 98-10, Department of Statistics, Operations Research and Computer Science, University of Vienna, Austria.

[26] Holsenback, J.E. and R.M. Russell (1992): A Heuristic Algorithm for Sequencing on One Machine to Minimize Total Tardiness, *Journal of the Operational Research Society* **43** pp. 53-62.

[27] Lawler, E.L. (1977): A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness, *Annals of Discrete Mathematics* **1** pp. 331-342.

[28] Lenstra, J.K., A.H.G. Rinnooy Kan and P. Bruckner (1977): Complexity of machine scheduling problems, in: P.L. Hammer, E.L. Johnson, B.H. Korte and G.L. Nemhauser, eds., Studies in Integer Programming, Annals of Discrete Mathematics 1, North-Holland, Amsterdam, pp. 343-362.

[29] Maniezzo, V., A. Colorni and M. Dorigo (1994): *The Ant System applied to the Quadratic Assignment Problem*, Technical Report No. 94-28, IRIDIA, Brussels, Belgium.

[30] Matsuo, H., C.J. Suh and R.S. Sullivan (1987): A controlled search simulated annealing method for the single machine weighted tardiness problem, Working paper 87-12-2, Department of Management, University of Texas, Austin.

[31] Panwalkar, S.S., M.L. Smith and C.P. Koulamas (1993): A Heuristic for the Single Machine Tardiness Problem, *European Journal of Operational Research* **70** pp. 304-310.

[32] Potts, C.N. and L.N. Van Wassenhove (1985): A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, *Operations Research* **33** pp. 363-377.

[33] Potts, C.N. and L.N. Van Wassenhove (1982): A Decomposition Algorithm for the Single Machine Tardiness Problem, *Operations Research Letters* **32** pp. 177-181.

[34] Potts, C.N. and L.N. Van Wassenhove (1991): Single Machine Tardiness Sequencing Heuristics, *IIE Transactions* **23** pp. 346-354.

[35] Russell, R.M. and J.E. Holsenback (1997a): Evaluation of Greedy, Myopic and Less-Greedy Heuristics for the Single Machine, Total Tardiness Problem, *Journal of the Operational Research Society* **48** pp. 640-646.

[36] Russell, R.M. and J.E. Holsenback (1997b): Evaluation of leading heuristics for the single machine tardiness problem, *European Journal of Operational Research* **96** pp. 538-545.

[37] Schrage, L.E. and K.R. Baker (1978): Dynamic Programming Solution of Sequencing Problems with Precedence Constraints, *Operations Research* **26** pp. 444-449.

[38] Stützle, T. (1998): *An Ant Approach to the Flow Shop Problem*, Proceedings of EUFIT'98, Aachen, pp. 1560-1564.

[39] Stützle, T. and H. Hoos (1997): *The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem*, Proceedings of IEEE 4th International Conference on Evolutionary Computation ICEC'97, IEEE Press, pp. 308-313.

[40] Szwarc, W. and S. Mukhopadhyay (1996): Decomposition of the Single Machine Total Tardiness Problem, *Operations Research Letters* **19** pp. 243-250.

[41] Wilkerson, L.J, and Irwin, J.D. (1971): An Improved Algorithm for Scheduling Independent Tasks, *AIIE Transactions* **3** pp. 239-245.