An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problems

Daniel Merkle[1] and Martin Middendorf[2]

Institute for Applied Computer Science and Formal Description Methods,
University of Karlsruhe, Germany
{[1]merkle,[2]middendorf}@aifb.uni-karlsruhe.de

Abstract. Ant Colony Optimization is an evolutionary method that has recently been applied to scheduling problems. We propose an ACO algorithm for the Single Machine Total Weighted Tardiness Problem. Compared to an existing ACO algorithm for the unweighted Total Tardiness Problem our algorithm has several improvements. The main novelty is that in our algorithm the ants are guided on their way to good solutions by sums of pheromone values. This allows the ants to take into account pheromone values that have already been used for making earlier decisions.

1 Introduction

Ant Colony Optimization (ACO) is an evolutionary metaheuristic to solve combinatorial optimization problems by using principles of communicative behaviour found in real ant colonies (for an introduction and overview see [5]). Recently the ACO approach has been applied to scheduling problems, like Job-Shop [2, 7], Flow-Shop [13], and the Single Machine Total Tardiness problem [1]. Bullnheimer et al. [1] have compared an ACO algorithm with several other heuristics to solve the Single Machine Total Tardiness problem (e.g. decomposition heuristics, interchange heuristics and simulated annealing). They have shown that the ACO algorithm found the optimal solution of 125 benchmark problems more often than the other heuristics (these benchmark problems where generated with the same method from [12] as the benchmarks problems used in this paper).

In this paper we propose alternative and improved ways to solve the Single Machine Total Tardiness problem by ACO. Moreover, we also study the weighted version of the total tardiness problem.

In ACO algorithms 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 until a solution is found. 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 approach used in [1] and [13] to solve scheduling problems with ACO algorithms is to 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.

Here we propose a different approach. 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. A problem with using only T_{ij} can occur when the ant does not chose job j as the ith job in the schedule. Because, if the $T_{i+1,j}$, $T_{i+2,j}, \ldots$ values are small then job j might be scheduled much later than at the ith place (and possibly long after its due date). It is likely that this will not happen when using $\sum_{k=1}^{i} T_{kj}$. Note, that this approach differs from nearly all other ant algorithms proposed so far, in that we base one possible decision of an ant on several pheromone values. The only other work that uses several pheromone values to estimate the quality of one possible decision is [11]. Moreover, we let the ants make optimal decisions when this is possible and use a heuristic that is a modification of the heuristic used in [1].

This paper is organized as follows. The Single Machine Total Weighted Tardiness Problem is defined in Section 2. In Section 3 we describe an ACO algorithm for the unweighted problem. The pheromone summation rule is introduced in Section 4. Section 5 contains further variants and improvements. The choice of the parameter values of our algorithms used in the test runs and the test instances and are described in Section 6. The results are reported in Section 7. A conclusion is given in Section 8.

2 The Single Machine Total Weighted Tardiness Problem

The Single Machine Total Weighted Tardiness Problem (SMTWTP) is to find for n jobs, where job j, $1 \leq j \leq n$ has a processing time p_j, a due date d_j, and a weight w_j, a non-preemptive one machine schedule that minimizes $T = \sum_{j=1}^{n} w_j \cdot \max\{0, C_j - d_j\}$ where C_j is the completion time of job j. T is called the total weighted tardiness of the schedule. The unweighted case, i.e. $w_j = 1$ for all $j \in \{1, \ldots, n\}$, is the Single Machine Total Tardiness Problem (SMTTP).

It is known that SMTTP is NP-hard in the weak sense [8] and SMTWTP is NP-hard in the strong sense [10]. A pseudopolynomial time algorithm for SMTWTP in case that the weights agree with the processing times (i.e. $p_j < p_h$ implies $w_j \geq w_h$) was given in [10]. Observe, that the last result implies that SMTTP is pseudopolynomial time solvable. For an overview over different heuristics for SMTWTP see [4].

3 ACO Algorithm for SMTTP

The ACO algorithm of Bullnheimer et al. [1] is described in this section. The general idea was to adapt an ACO algorithm called ACS-TSP for the traveling salesperson problem of Dorigo et al. [6] for the SMTTP. In every generation each of m ants constructs one solution. An ant selects the jobs in the order in which they will appear in the schedule. For the selection of a job 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 an indicator of how good it seems to have job j at place i of the schedule. The heuristic value is generated by some problem dependent heuristic whereas the pheromone information stems from former ants that have found good solutions. With probability q_0, where $0 \leq q_0 < 1$ is a parameter of the algorithm, the ant chooses a job j from the set S of jobs that have not been scheduled so far which maximizes

$$[\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}$$

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. With probability $1-q_0$ the next job is chosen according to the probability distribution over S determined by

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{h \in S} [\tau_{ih}]^{\alpha} [\eta_{ih}]^{\beta}}$$

The heuristic values η_{ij} are computed according the Modified Due Date rule (MDD), i.e.,

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

where $\mathcal{T}$ is the total processing time of all jobs already scheduled.

After an ant has selected the next job j, a local pheromone update is performed at element (i, j) of the pheromone matrix according to

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

for some constant ρ, $0 \leq \rho < 1$ and where

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

and T_{EDD} is the total tardiness of the schedule that is obtained when the jobs are ordered according to the Earliest Due Date heuristic (EDD), i.e., with falling values of $1/d_j$. The value τ_0 is also used to initialize the elements of the pheromone matrix.

After all m ants have constructed a solution the best of these solutions is further improved with a 2-opt strategy. The 2-opt strategy considers swaps between all pairs of jobs in the sequence. Then it is checked whether the so derived schedule is the new best solution found so far.

The best solution found so far is then used to update the pheromone matrix. But before that some of the old pheromone is evaporated according to

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij}$$

The reason for this is that old pheromone should not have a too strong influence on the future. Then, for every job j in the schedule of the best solution found so far some amount of pheromone is added to element (ij) of the pheromone matrix where i is the place of job j in the schedule. The amount of pheromone added is ρ/T^* where T^* is the total tardiness of the best found schedule, i.e.,

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

The algorithm stops when some stopping criterion is met, e.g. a certain number of generations has been done or the best found solution has not changed for several generations.

4 The Pheromone Summation Rule

In this section we describe a new approach of using the pheromone values which is used in our ACO algorithm for SMTTP. In general, a high pheromone value τ_{ij} means that it is advantageous to put job j at place i in the schedule. Assume now that by chance an ant chooses to put some job h at place i of the schedule that has a low pheromone value τ_{ih} (instead of a job j that has a high pheromone value τ_{ij}). Then in order to have a high chance to still end up with a good solution it will likely be necessary for the ant to place job j not too late in the schedule when j has a small due date. To some extend the heuristic values η_{lj} for $l > i$ will then force the ant to choose j soon. But a problem occurs when the values τ_{lj} are small (because no good solutions have been found before that have job j at some place $l > i$). Then the product $(\eta_{lj})^\alpha \cdot (\tau_{lj})^\beta$ is small and it is likely that the ant will not choose j soon. In this case the ant will end up with a useless solution having a high total tardiness value.

To handle this problem we propose to let a pheromone value τ_{ij} also influence later decisions when choosing a job for some place $l > i$. A simple way to guaranty this influence is to use the sum of all pheromone values for every job from the first row of the matrix up to row i when deciding about the job for place i. When using this pheromone summation rule we have the following modified decision formulas. An ant chooses as next job for place i in the schedule with probability q_0 the job $j \in S$ that maximizes

$$(\sum_{k=1}^{i} [\tau_{kj}])^\alpha \cdot [\eta_{kj}]^\beta \qquad (2)$$

and with probability $1 - q_0$ job $j \in S$ is chosen according to the probability distribution over S determined by

$$p_{ij} = \frac{(\sum_{k=1}^{i} [\tau_{kj}])^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in S} (\sum_{k=1}^{i} [\tau_{kh}])^{\alpha} \cdot [\eta_{ih}]^{\beta}} \tag{3}$$

5 Further Variations and Improvements

In this section we describe further variations and improvements that we used in our ACO algorithm.

5.1 Modified Heuristic

A problem when using the heuristic values according to formula (1) is that the values of $\max\{\mathcal{T} + p_j, d_j\}$ become much larger — due to $\mathcal{T}$ — when deciding about jobs to place at the end of the schedule than they are when placing jobs at the start of the schedule. As a consequence the heuristic differences between the jobs are, in general, small at the end of the schedule. To avoid this effect we used the following modified η values

$$\eta_{ij} = \frac{1}{\max\{\mathcal{T} + p_j, d_j\} - \mathcal{T}} \tag{4}$$

For the weighted problem SMTWTP we multiplied every value on the right side of equation (4) with the weight w_j of job j. Note that jobs with a small weighted processing time p_j/w_j have a high heuristic value when $\mathcal{T} + p_j \geq d_j$.

5.2 Deterministic Scheduling Between Due Dates

Consider the construction of a schedule for the unweighted problem SMTTP. Assume that some jobs have already been scheduled. Assume further that the sum $\mathcal{T}$ of the processing times of all jobs scheduled so far lies between some due date d_j and a due date $d_h > d_j$ and every other due date is smaller than d_j or larger than d_h. For this case it is easy to show that it is optimal to schedule all jobs with a due date $\leq d_j$ before scheduling a job with a due date $\geq d_h$ as long as the sum of the processing times of the scheduled jobs is at most d_h. Moreover when there are several jobs with due date $\leq d_j$ it is optimal to schedule these jobs ordered by increasing processing times. If the ants apply this deterministic rule whenever possible we say that the ants work locally deterministic. Then the ants will switch between probabilistic and deterministic behaviour.

6 Test Instances and Parameters

We tested the different variants of ACO algorithms on 125 benchmark instances for SMTWTP of size 100 jobs that are included in the OR-Library [14]. These benchmark instances were generated as follows: for each job $j \in [1 : 125]$ an integer processing time p_j is taken randomly from the interval $[1 : 100]$, an integer weight w_j is taken randomly from the interval $[1:10]$ and an integer due date d_j is taken randomly from the interval

$$\left[\sum_{j=1}^{125} p_j \cdot (1 - TF - \frac{RDD}{2}), \sum_{j=1}^{125} p_j \cdot (1 - TF + \frac{RDD}{2})\right]$$

The value RDD (relative range of due dates) determines the length of the interval from which the due dates were taken. TF (tardiness factor) determines the relative position of the centre of this interval between 0 and $\sum_{j=1}^{125} p_j$. The values for TF and RDD are chosen from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. The benchmark set contains five instances for each combination of TF and RDD values. For the unweighted problem SMTTP we used the same benchmark instances but ignored the different weights.

Our results for SMTWTP were compared to the best known results for the benchmark instances that are from [3] and can be found in [14].

The parameters used for the test runs are: $\alpha = 1, \beta = 1, \rho = 0.1, q_0 \in \{0, 0.9\}$. The number of ants in every generation was $m = 20$. Every test was performed with 4 runs on every instance. Every run was stopped after 500 generations.

We used a 2-opt strategy to improve the best solution that was found in every generation which differs slightly from the 2-opt strategy used in [1]. For every pair of jobs it was checked exactly once whether a swap of these jobs improves the schedule. A swap that improves the schedule was fixed immediately. Thus we tried exactly 4950 swaps per generation.

In the following ACS-SMTTP (or short ACS) denotes the algorithm of [1] as described in Section 3 but with the new 2-opt strategy described in the last paragraph. Our algorithm ACS-SMTWTP-Σ is similar to ACS but uses the pheromone summation rule as described in Section 4. Algorithm ACS-SMTWTP-H is similar to ACS but uses the new heuristic from Section 5.1. Algorithm ACS-SMTWTP-D is similar to ACS but additionally uses the deterministic strategy from Section 5.2 for scheduling between due dates. Algorithms that use combinations of new features are denoted by ACS-SMTWTP-XYZ where X,Y,Z$\in \{\Sigma$, H, D$\}$ (e.g. ACS-SMTWTP-HΣ uses the new heuristic and the pheromone summation rule). For shortness we write ACS-XYZ for ACS-SMTWTP-XYZ.

7 Experimental Results

The influence of the pheromone summation rule (called Σ-rule in the following) and the modified heuristic was tested on weighted and unweighted problem instances. Since the parameter q_0 has some influence on the results we performed tests with $q_0 = 0$ and $q_0 = 0.9$.

Table 1 shows the results for SMTWTP. The average total tardiness values found by the ACO algorithms for SMTWTP were compared to the average total tardiness of the best known solutions that are from [3]. The average total tardiness per instance of the best solutions from [3] is 217851.34. Table 1 shows that ACS-ΣH performed better than ACS-H and also that ACS-Σ performed better than ACS (this holds for both cases $q_0 = 0$ and $q_0 = 0.9$). In all cases the difference of the total tardiness values compared to the best known solutions are at least 61.1% lower for the ACO algorithm with Σ-rule (79.5 for ACS-ΣH compared to 204.5 for ACS-H with $q_0 = 0.9$). Moreover, the ACO algorithms with Σ-rule found for more instances a better total tardiness than their counterparts without Σ-rule (at least 5.3 times as often). The differences of the total tardiness values compared to the best known values over the first 200 generations are shown in Figure 1. The best solution of ACS-ΣH was found after an average of 80 generations, which was after less than 3.5 seconds on a 450 MHz Pentium-II processor.

Table 1 also shows that the ACO algorithms with modified heuristic performed in all cases better than their counterparts using the heuristic from [1]. For $q_0 = 0.9$ the advantage of the modified heuristic is smaller than for $q_0 = 0$ (e.g. for $q_0 = 0.9$ ACS-ΣH has a 60.2% smaller difference to optimal total tardiness than ACS-Σ compared to a 92.9% smaller difference for $q_0 = 0$).

Table 1. Influence of pheromone summation rule and new heuristic on solution quality for SMTWTP. Total Tardiness: average difference to total tardiness of best found solutions from [3] (average over 500 test runs, 125 instances and 4 runs for each instance); Better: comparisons between ACS-ΣH and ACS-H (respectively ACS-Σ and ACS), number of instances with smaller average total tardiness (average over 125 instances and 4 runs for each instance).

weighted		ACS-ΣH	ACS-H	ACS-Σ	ACS
Total	$q_0 = 0$	191.8	3024.7	946.1	9914.7
Tardiness	$q_0 = 0.9$	79.5	204.5	200.0	1198.6
Better	$q_0 = 0$	97	2	106	0
	$q_0 = 0.9$	86	16	97	3

Table 2 shows the results for the unweighted problem SMTTP. The results are compared with the average of the best total tardiness values we found for the unweighted instances, i.e. 54309.5. Similarly as for the weighted problem in all cases the ACO algorithms with Σ-rule are better than their counterparts without Σ-rule. Also the modified heuristic performed better in all cases than the heuristic from [1].

Since the 2-opt strategy significantly influences of the quality of the solutions we also compared the ACS-ΣH with ACS-H when using no 2-opt strategy. The results can be found in Table 3 for SMTWTP and in Table 4 for SMTTP. The only case where ACS-ΣH performed not significantly better than ACS-H is the unweighted case with $q_0 = 0.9$. In this case ACS-H found a slightly better average total tardiness ACS-ΣH (difference is 331.5 for ACS-H and 332.3 for ACS-ΣH). On the other hand ACS-ΣH found for more instances better solutions than ACS-H (For 65 instances ACS-ΣH found better solutions than ACS-H whereas ACS-H performed better than ACS-ΣH for 33 instances).

Fig. 1. SMTWTP: Average difference to total tardiness of best found solutions from [3] over the first 200 generations.

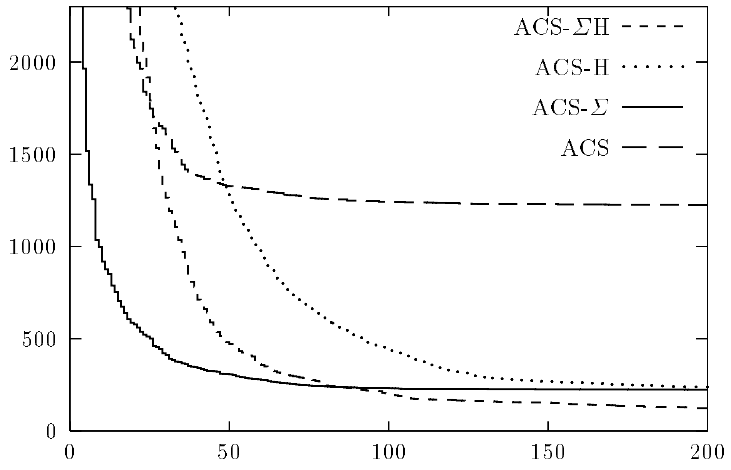

Table 2. Influence of pheromone summation rule and new heuristic on solution quality for SMTTP. Total Tardiness: average difference to total tardiness of best found solutions (average over 500 test runs, 125 instances and 4 runs for each instance); "Better" as in Table 1.

unweighted		ACS-ΣH	ACS-H	ACS-Σ	ACS
Total Tardiness	$q_0 = 0$	47.9	48.5	112.9	256.4
	$q_0 = 0.9$	7.0	19.0	8.7	26.3
Better	$q_0 = 0$	53	32	82	17
	$q_0 = 0.9$	53	22	67	14

Table 3. Influence of pheromone summation rule and new heuristic on solution quality for SMTWTP when using no 2-opt. "Total Tardiness" as in Table 1; "Better" as in Table 1 but comparison between ACS-ΣH and ACS-H.

no 2-0pt, weighted		ACS-ΣH	ACS-H
Total Tardiness	$q_0 = 0$	11894.4	22046.8
	$q_0 = 0.9$	1733.2	1793.5
Better	$q_0 = 0$	76	48
	$q_0 = 0.9$	67	42

Table 4. Influence of pheromone summation rule and new heuristic on solution quality for SMTTP when using no 2-opt. "Total Tardiness" as in Table 2; "Better" as in Table 1 but comparison between ACS-ΣH and ACS-H.

no 2-0pt, unweighted		ACS-ΣH	ACS-H
Total Tardiness	$q_0 = 0$	3943.7	4515.8
	$q_0 = 0.9$	332.3	331.5
Better	$q_0 = 0$	59	50
	$q_0 = 0.9$	65	33

The influence of the deterministic strategy for scheduling between due dates for SMTTP has only a minor influence on the results for the unweighted benchmark instances from the OR-Library. The reason is that these instances have small gaps between the due dates. Thereby, the deterministic strategy does come into play only rarely. Hence, we created new test instances which have two neighboured due dates that have a large gap in between. We changed each of the problem instances from the OR-Library as follows. The jobs were ordered by their due dates and the due dates of jobs 41 to 59 were set to the same due date that job 40 has. The average of the best total tardiness values we found for these modified instances was 56416.3. Table 5 shows for $q_0 = 0$, that ACS-ΣHD performed much better than ACS-ΣH and also that ACS-HD performed much better than ACS-H. For $q_0 = 0.9$ the ACS-ΣHD algorithm could not profit from the deterministic scheduling between due dates.

Table 5. Influence of deterministic strategy between due dates on solution quality for SMTTP and problem instances with modified due dates. "Total Tardiness as in Table 2; "Better" as in Table 1 but comparison between ACS-ΣH and ACS-ΣHD.

unweighted		ACS-ΣH	ACS-ΣHD	ACS-H	ACS-HD
Total Tardiness	$q_0 = 0$	101.4	45.7	120.1	3.8
	$q_0 = 0.9$	2.9	8.7	11.1	9.2
Better	$q_0 = 0$	8	78	1	92
	$q_0 = 0.9$	36	14	29	36

8 Conclusion

We have introduced a new method to use the pheromone values in an Ant Colony Optimization (ACO) algorithm for the Single Machine Total Weighted Tardiness problem. An ACO algorithm using this pheromone summation rule gives better solutions for 125 benchmark than its counterpart that does not use the summation rule. This holds also for the unweighted total tardiness problem. Moreover, we proposed a new heuristic that can be used by the ants when searching for a solution. For the unweighted problem we have shown that the ACO algorithm can profit from ants that switch between a deterministic behaviour (in case that optimal decisions can be made) and the "standard" probabilistic behaviour.

References

1. A. Bauer, B. Bullnheimer, R.F. Hartl, C. Strauss: 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, 1445-1450, 1999.
2. A. Colorni, M. Dorigo, V. Maniezzo, M. Trubian: Ant System for Job-Shop Scheduling; *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, 34: 39–53 (1994).
3. R.K. Congram, C.N. Potts, S. L. van de Velde: An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem; submitted to *INFORMS Journal on Computing.*
4. H.A.J. Crauwels, C.N. Potts, L.N. Van Wassenhove: Local Search Heuristics for the Single Machine Total Weighted Tardiness Scheduling Problem; *INFORMS Journal on Computing*, 10: 341–359 (1998).
5. M. Dorigo, G. Di Caro: The ant colony optimization meta-heuristic; in: D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideas in Optimization*, McGraw-Hill, 1999, 11-32.
6. M. Dorigo, L. M. Gambardella: Ant colony system: A cooperative learning approach to the travelling salesman problem; *IEEE Trans. on Evolutionary Comp.*, 1: 53-66 (1997).
7. M. Dorigo, V. Maniezzo, A. Colorni: The Ant System: Optimization by a Colony of Cooperating Agents; *IEEE Trans. Systems, Man, and Cybernetics – Part B*, 26: 29-41 (1996).
8. J. Du, J.Y.-T. Leung: Minimizing the Total Tardiness on One Machine is NP-hard; *Mathematics of Operations Research*, 15: 483–496 (1990).
9. P. Forsyth, A. Wren: An Ant System for Bus Driver Scheduling; Report 97.25, University of Leeds - School of Computer Studies, 1997.
10. E.L. Lawler: A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness; *Annals of Discrete Mathematics*, i: 331–342 (1977).
11. R. Michels, M. Middendorf: An Ant System for the Shortest Common Supersequence Problem; in: D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideas in Optimization*, McGraw-Hill, (1999) 692-701.
12. C.N. Potts, L.N. Van Wessenhove: Single machine tardiness sequencing heuristics; *IEE Transactions*, 23: 346-354 (1991)
13. T. Stützle: An ant approach for the flow shop problem; in *Proc. of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT '98)*, Vol. 3, Verlag Mainz, Aachen, 1560-1564, 1998.
14. http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html.