Ant Colony Optimization for the Total Weighted Tardiness Problem

Matthijs den Besten[1,2], Thomas Stützle[2,3], and Marco Dorigo[2]

[1] University of Amsterdam, Department of Computer Science
Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands
[2] Université Libre de Bruxelles, IRIDIA,
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
[3] Darmstadt University of Technology, Intellectics Group,
Alexanderstr. 10, 64283 Darmstadt, Germany

Abstract. In this article we present an application of the Ant Colony Optimization (ACO) metaheuristic to the single machine total weighted tardiness problem. First, we briefly discuss the constructive phase of ACO in which a colony of artificial ants generates a set of feasible solutions. Then, we introduce some simple but very effective local search. Last, we combine the constructive phase with local search obtaining a novel ACO algorithm that uses a heterogeneous colony of ants and is highly effective in finding the best-known solutions on all instances of a widely used set of benchmark problems.

1 Introduction

In this paper we study the single machine total weighted tardiness problem (SMTWTP) [21], a scheduling problem that is known to be $\mathcal{NP}$-hard [16] and for which instances with more than 50 jobs can often not be solved to optimality with state-of-the-art branch & bound algorithms [1, 5].

In the SMTWTP n jobs have to be sequentially processed on a single machine. Each job j has a processing time p_j, a weight w_j, and a due date d_j associated, and the jobs become available for processing at time zero. The tardiness of a job j is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j in the current job sequence. The goal is to find a job sequence which minimizes the sum of the weighted tardiness given by $\sum_{i=1}^{n} w_i \cdot T_i$.

Because the SMTWTP is $\mathcal{NP}$-hard, several heuristic methods have been proposed for its solution. These include construction heuristics like the Earliest Due Date or the Apparent Urgency heuristics (see [22] for an overview) and metaheuristics like Simulated Annealing [18, 22], Tabu Search and Genetic Algorithms [5].

In this paper we present the application of the Ant Colony Optimization (ACO) metaheuristic [8, 9] to the SMTWTP. ACO has been introduced by Dorigo and colleagues as a new optimization paradigm which is inspired by the trail following behavior of real ant colonies. Algorithmic implementations of this metaheuristic have shown very promising results for the well known Traveling Salesman Problem [10, 25] and are currently among the best available algorithms for other hard problems like the Quadratic Assignment Problem [13, 17, 26], the Sequential Ordering Problem [11], Vehicle Routing Problems [12], and routing problems in highly dynamic environments [7].

procedure *ACO metaheuristic for static combinatorial problems*
  Set parameters, initialize pheromone trails
  while (termination condition not met) **do**
    ConstructSolutions
    ApplyLocalSearch % optional
    UpdateTrails
  end

Fig. 1. Algorithmic skeleton for the ACO metaheuristic applied to static problems.

The paper is structured as follows. In Section 2 we introduce ACO and Ant Colony System, the particular ACO algorithm applied here. Section 3 presents the local search algorithm we applied and Section 4 investigates the proposed ACO algorithm and gives final performance results. We conclude with some final remarks in Section 5.

2 Ant Colony Optimization

In ACO algorithms a colony of (artificial) ants iteratively constructs solutions to the problem under consideration using (artificial) pheromone trails which are associated with appropriately defined solution components and heuristic information. The ants only communicate indirectly by modifying the pheromone trails during the algorithm's execution. Because the constructed solutions need not be locally optimal with respect to small modifications, in many of the best performing ACO algorithms the ants additionally improve their solutions by applying a local search algorithm. Hence, most ACO algorithms for static combinatorial optimization problems[4] follow the particular algorithmic scheme given in Figure 1.

2.1 Solution construction

We apply Ant Colony System (ACS) [10], a particular ACO algorithm, to the SMTWTP. A feasible solution for the SMTWTP, also called a *sequence*, consists of a permutation of the jobs. When applied to the SMTWTP, each ant starts with an empty sequence and then iteratively appends an unscheduled job to the partial sequence constructed so far. In ACS jobs are scheduled as follows.

- With probability q, the unscheduled job j that maximizes $\tau_{ij}(t) \cdot \eta_{ij}^{\beta}$ is put in the current sequence at position i. Here, $\tau_{ij}(t)$ is the pheromone trail associated to the assignment of job j to position i, t indicates the dependence of the pheromone trail on the time, η_{ij} is the heuristic desirability of assigning job j to position i, and β is a parameter which determines the influence of the heuristic information.

[4] We call static those problems whose topology and costs do not change while they are being solved. The ACO metaheuristic can be applied also to dynamic problems in which topology and costs can change while solutions are built. An example is the *AntNet* algorithm that was applied to a routing problem for Internet-like networks [7].

- With probability $1 - q$, job j is chosen randomly with a probability given by

$$p_{ij} = \begin{cases} \frac{\tau_{ij}(t) \cdot \eta_{ij}^{\beta}}{\sum_{l \text{ not scheduled}} \tau_{il}(t) \cdot \eta_{il}^{\beta}} & \text{if job } j \text{ is not yet scheduled} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Hence, with a probability q the ant does the best decision as indicated by the pheromone trails and the heuristic information (exploitation), while with probability $1 - q$ it performs a biased exploration. The three following standard heuristics have been used to compute the heuristic information η_{ij} used by the ants:

- Earliest Due Date (EDD). This heuristic puts the jobs in non-decreasing order of the due dates d_j. In this case $\eta_{ij} = 1/d_j$.
- Modified Due Date (MDD). This heuristic puts the jobs in non-decreasing order of the modified due dates mdd_j [2] given by $mdd_j = \max\{C + p_j, d_j\}$, where C is the sum of the processing times of the already sequenced jobs. In this case $\eta_{ij} = 1/mdd_j$.
- Apparent Urgency (AU). This heuristic puts the jobs in non-decreasing order of the apparent urgency [20], given by $au_j = (w_j/p_j) \cdot \exp(-(\max\{d_j - C_j, 0\})/k\overline{p})$. Here, $\overline{p}$ is the average processing time of the remaining jobs, k is a parameter set as proposed in [22], and $\eta_{ij} = 1/au_j$.

2.2 Pheromone update

In ACS, two forms of pheromone update are applied. In the delayed pheromone trail update after each iteration pheromone trail is added to the components of the global-best solution, that is, the best sequence found so far. If in the global-best solution at iteration t job j is put on position i, then $\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t)$, where ρ, $0 < \rho \leq 1$, is a parameter representing the pheromone evaporation and $\Delta\tau_{ij}(t) = 1/T^*$, where T^* is the total weighted tardiness of the global-best solution.

Additionally, ACS applies a step-by-step pheromone update rule immediately after an ant has added a new job to the partial sequence. Here, pheromone trails are modified by the update rule $\tau_{ij} = (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0$, where ξ, $0 < \xi \leq 1$, and τ_0, which is chosen as a small value, are two parameters. The effect of the local updating rule is to make the decision of putting job j on position i less desirable for the other ants so that the exploration of different sequences is favored.

2.3 Other ACO approaches to scheduling problems

An application of ACO to the *unweighted* single machine total tardiness problem was presented in [2]. Although for this problem heuristic approaches appear not to be very interesting, because the total (unweighted) tardiness problem can be rather efficiently solved by enumerative methods [6], we verified that our approach substantially outperforms this early application. In the literature only few other applications of ACO algorithms to scheduling problems are reported. In [3] an application of Ant System, the first ACO algorithm, to the Job Shop Scheduling problem is reported. Yet, this approach obtained relatively poor computational results, probably because no local search was used to improve solutions. Much better performance is reported for the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System [25, 27] to the permutation Flow Shop Problem [24].

3 Local search for the SMTWTP

Local search for the SMTWTP starts from some initial sequence and repeatedly tries to improve the current sequence by replacing it with neighboring solutions. If in the neighborhood of the current sequence π a better sequence is found, it replaces π and the local search is continued from the new solution. The simplest local search algorithm, iterative descent, repeatedly applies these steps until no better neighboring sequence can be found and stops at the first local minimum encountered. Critical for the performance of the local search algorithm is the neighborhood structure chosen. For the SMTWTP we considered the following two neighborhood structures:

(1) exchanges of jobs placed at the ith and the jth position, $i \neq j$ (*interchange*)
(2) removal of the job at the ith position and insertion in the jth position (*insert*)

The implementation of these local search algorithms was sped-up using the techniques described in [4]. To achieve further improvements of the solution quality, we considered the concatenation of the iterative descent algorithms using the two different neighborhoods. This may be reasonable because a local optimum with respect to one neighborhood structure need not be a local optimum for the other one. In fact, the recent Variable Neighborhood Search metaheuristic [19] systematically applies the idea of changing neighborhoods in the search. The concatenation of two local search algorithms has also been applied in [23] for the permutation Flow Shop Problem. The concatenated local search algorithms, called Variable Neighborhood Descent (VND) in the following, will be denoted as *interchange+insert* and *insert+interchange*, depending on which neighborhood is searched first.

To evaluate local search we used a benchmark set of randomly generated instances, available via ORLIB at `http://www.ms.ic.ac.uk/info.html`. The benchmark set comprises instances with 40, 50, and 100 jobs. For the 40 and 50 job instances the optimal solutions are known, while for the 100 job instances only the best known solutions are given. The instances are generated randomly by drawing the processing time p_j for each job j randomly according to a uniform distribution of integers between 1 and 100 and assigning it a weight w_j randomly drawn from a uniform distribution over the integers between 1 and 10. The due dates are randomly drawn integers from the interval $[(1-\mathrm{TF}-\mathrm{RDD}/2)\cdot\sum p_i, (1-\mathrm{TF}+\mathrm{RDD}/2)\cdot\sum p_i]$, where TF, the tardiness factor, and RDD, the relative due date, are two parameters. There are five instances for each pair of TF and RDD from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. This makes three sets of 125 instances each. The tardiness factor and the relative due dates determine critically the difficulty of solving the instances. For example, we found that most of the instances with TF $= 0.2$ are solved after one single application of the local search procedure, while for larger TF, the instances were much harder to solve. All the experiments were run using a 450MHz Pentium III PC with 256 MB RAM. Programs were written in C++ and run under Red Hat Linux 6.1.

In Table 1 we give computational results for the three construction heuristics EDD, MDD, and AU, and for the proposed local search algorithms when starting from initial solutions generated by the three construction heuristics without local search. As indicated before, some of the instances are very easily solved. This can be noted in Table 1 observing that a large number of best known solutions are found by the construction

Table 1. Comparison of the local search effectiveness for the SMTWTP. Results on the 100 job instances without local search and using the interchange, the insert, the interchange+insert, and the insert+interchange local search. We give the average percentage deviation from the best known solutions (Δ_{avg}), the number of best-known solutions found (n_{opt}), and the average CPU time in seconds (t_{avg}) averaged over the 125 benchmark instances.

start	no local search			interchange			insert			inter+insert			insert+inter		
	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}
EDD	135	24	0.001	0.62	33	0.140	1.19	38	0.64	0.24	46	0.20	0.47	48	0.67
MDD	61	24	0.002	0.65	38	0.078	1.31	36	0.77	0.40	46	0.14	0.44	42	0.79
AU	21	21	0.008	0.92	28	0.040	0.56	42	0.26	0.59	46	0.10	0.21	49	0.27

heuristics without local search. Yet, when averaged over the whole benchmark set, the construction heuristics alone perform rather poorly and the application of a local search algorithm increases very strongly the average solution quality with the VND showing a significant improvement over the single neighborhood local search. Table 1 also shows that the initial solutions generated by AU are best combined with insert local search, while initial solutions from the EDD and MDD heuristics are best combined with interchange local search. Regarding computation time, the *interchange* local search is significantly faster than the *insert* local search. Yet, interestingly, the computation time of the VND is only marginally higher than the one taken by its first local search.

4 Combining ACS with local search

In ACO, as in many other metaheuristics, the performance critically depends on the appropriate combination of local search with the solution construction mechanism. We performed therefore a detailed analysis of the effectiveness of the coupling of ACS with the four local search algorithms on 10 hard SMTWTP instances from the 100 job benchmark set. For the experiments we used the following default parameter settings: 20 ants, $\beta = 2$, $\xi = 0.1$, $\rho = 0.1$, $q = 0.9$, and $\tau_0 = 1/(n \cdot T_{MDD})$, where T_{MDD} is the total weighted tardiness of the solution generated by the MDD heuristic and n is the number of jobs.[5]

4.1 Configuration of ACS for the SMTWTP

In the following we present the results using run-time distributions (RTDs) which give the cumulative empirically observed probability of finding an optimal solution as a function of the CPU time [15, 25]. Here, for each instance 25 runs have been performed using the MDD heuristic information (the same experiments were also run using the AU heuristic). In Figure 2 are given the run-time distributions for the $14th$ (left plot in Figure 2) and the $67th$ instance (right plot in Figure 2) of the 125 instances of the 100 job benchmark set (on the other eight instances the relative performance of the local search algorithms was similar). From the RTDs we could make the following interesting observations. First, the VND algorithms *interchange+insert* and *insert+interchange* performed typically better than a single local search in the first neighborhood (*interchange*

[5] Analogous parameter settings were found to yield good performance in earlier studies with ACS on the Traveling Salesman Problem [10] and the Sequential Ordering Problem [11].

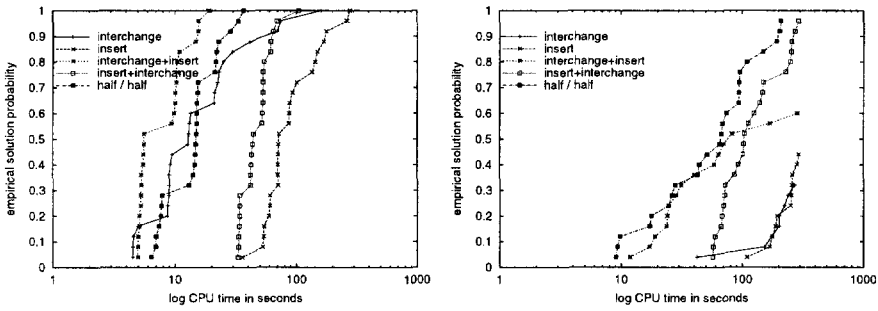

Fig. 2. Comparison for combinations of the different local search algorithms with ACS based on the empirical cumulative probability distribution for finding the best-known solution measured over 25 independent runs. We give the run-time distributions for reaching the best known solution value on the $14th$ (left) and the $67th$ instance (right) of the 125 available 100 job instances.

and *insert*, respectively). Hence, the higher solution quality obtained by VND clearly compensates the small run-time disadvantage. Second, which of the two local search algorithms (*interchange+insert* or *insert+interchange*) should be run first is strongly instance dependent. For example, on the $14th$ instance the *interchange+insert* VND clearly dominates the *insert+interchange* VND, as noted from the fact that for all computation times it achieves a much higher probability of finding optimal solutions. On the other side, on the $67th$ instance the *interchange+insert* VND achieves larger solution probabilities only for short computation times; for longer run-times the ACS using *insert+interchange* VND is significantly better.

Having noticed the instance dependence of the appropriate choice for the local search algorithm, we addressed this problem considering an ACS algorithm using a heterogeneous colony of ants, in which half of the ants apply *interchange+insert* and the other half apply *insert+interchange* VND. Using the heterogeneous colony, we obtained the RTDs indicated with *half/half* in Figure 2 and, as can be observed, the heterogeneous colony for all CPU-times obtains a larger solution probability than the worse of the two homogeneous colonies using VND; interestingly, on the $67th$ instance, the heterogeneous ant colony performed even better than the best homogeneous colony. We have observed this phenomenon also on some of the other instances.

When closer examining the solution construction, we noted that if a job j is not put on the position i for which $\tau_{ij}(t) \cdot \eta_{ij}^{\beta}$ is maximal, often it is put at one of the last positions in the sequence. Because the absolute position of a job is important, we introduced additional candidate lists [24, 10]. In particular, the candidate set is defined dynamically during the solution construction and comprises the first still not sequenced $cand$ jobs of the global-best solution π^*. The next job is then chosen among those of the candidate set in the case of a probabilistic choice according to Equation 1. We found that with $cand = 20$ a reasonably good performance is obtained.

Additionally, we analyzed the number of ants which should be used in ACS to achieve a good trade-off between the probability of finding best-known solutions and the necessary run-time to do so. In general, best behavior has been obtained with a num-

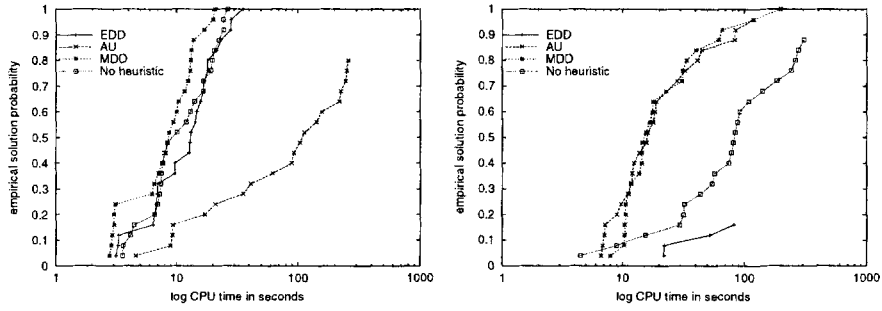

Fig. 3. Comparison on the influence of the heuristic information on ACS results. We give the RTDs for reaching the best known solution value on the $14th$ (left) and the $67th$ instance (right) of the 125 available 100 job instances.

ber of ants between 5 and 10. For a too small number of ants, some optimal solutions are found rather fast, but with increasing run-time often ACS with a larger number of ants achieves higher probabilities of finding optimal solutions. For the following experiments we used 10 ants.

As a next step, we analyzed the influence of the heuristic information on the final results. An important difference between static and dynamic heuristic information is that static information does not depend on the partial solution constructed by the ant while dynamic information does. When using static information, like the EDD-based heuristic information, the values of $\tau_{ij}(t) \cdot \eta_{ij}^{\beta}$ can be precomputed and need to be only occasionally updated. With dynamic heuristic information this is typically not possible and in this case the solution construction incurs a higher computational cost. Yet, this may be compensated by a higher accurateness of the dynamic heuristic information.

In Figure 3 we give RTDs for the two previously examined instances using the three different heuristics and not using heuristic information at all. The observed RTDs show that it is instance dependent which heuristic information yields best results. For example, while on the $14th$ instance the AU-based heuristic information showed to be even worse than not using any heuristic information at all, on the $67th$ instance the AU-based heuristic information performed best together with MDD. In general, we found that the advantage of the AU-based heuristic information increased with increasing range of due dates (RDD) and generally performed best for larger RDD. Only on instances with small RDD the MDD-based heuristic information gave significantly better results. Therefore, in ACS we use MDD-based heuristic information if RDD $\leq$ 0.3 and otherwise the AU-based heuristic information.

4.2 Final Results

In final experiments we tested ACS on the SMTWTP benchmark instances from ORLIB. Because many of the instances could already be solved by one single local search we felt that the only real challenge would be to find the best known solutions on *all* instances. To account for this fact, we run ACS at least 25 times on each instance and we set the maximal CPU time to 1200 seconds, so that ACS could find for every instance

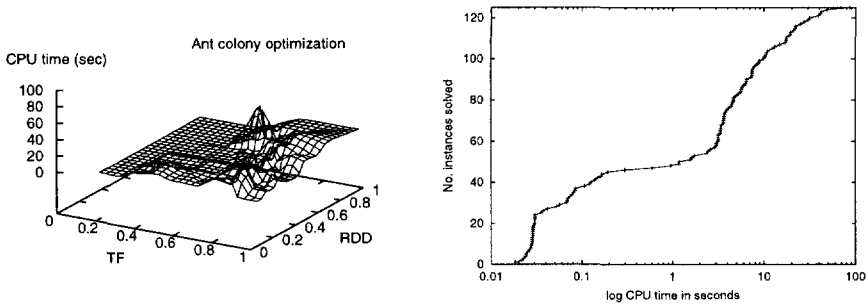

Fig. 4. Final study of the performance of ACS on the 100 job instances. On the left is given the average computation time to find optimal solutions in dependence of the relative due date (RDD) and the tardiness factor (TF), while on the right is given the cumulative number of instances solved in dependence of the average computation time (measured over 25 runs on each single instance) to solve them.

Table 2. We give some basic statistics on the distribution of the average computation times to solve instances of the three problem sets. We indicate the number of jobs, the average time (averaged over the 125 instances) to solve the benchmark set (t_{avg}) and the standard deviation (σ_t), the average time to solve the easiest and the hardest instance (t_{min} and t_{max}, respectively), and the quantils of the average time to solve a given percentage of the instances. Q_x indicates the average time to solve $x\%$ of the benchmark instances.

No. jobs	t_{avg}	σ_t	t_{min}	t_{max}	Q_{10}	Q_{25}	Q_{50}	Q_{75}	Q_{90}
40	0.088	0.021	0.004	1.72	0.004	0.008	0.013	0.122	0.215
50	0.32	0.869	0.006	10.74	0.007	0.016	0.031	0.353	0.606
100	6.99	7.019	0.018	86.26	0.028	0.070	3.190	7.499	19.30

the best known solutions in each single run. In Figure 4 and Table 2 we give indicative statistics on the average time taken by ACS to solve the instances.

We found that a large part of the benchmark set was solved in short computation time. For example, on the 100 job instances 113 out of 125 instances could be solved in less than 20 seconds on average. Only a small part of the instances, most of them with a TF around 0.6 and large RDD, took somewhat more time. A closer examination showed that the hardest instances occur at a point where roughly half of the jobs are late and half of the jobs non-late. In part, the increase of the computation time on such instances is also due to the fact that the speed-up techniques applied in the local search are not anymore as effective as on instances with many late or non-late jobs.

ACS appears to perform significantly better than most other previously proposed algorithms for the SMTWTP. For example, ACS finds always the best-known solutions for the 100 job instances, while the best performing Tabu Search algorithm in [5] could only find 103 of the best-known solutions at that time (some of those instances were later improved). The only algorithm reaching a similar performance as ACS appears to be iterated dynasearch (ID) [4]. ID found on average 123.2 times the best-known solutions (the same best-known solutions to which the Tabu Search was compared). Regarding the computation time, ACS may take slightly larger computation time. However, with ACS we could solve all benchmark instances in every single run.

5 Conclusions

In this paper we have presented ACS-SMTWTP, an effective ACO algorithm for the single machine total weighted tardiness problem. Our algorithm could find, for all known benchmark instances available in the ORLIB, the optimal or best-known solutions within reasonable computation times. Moreover, it was able to find the optimal solution in every run if enough computation time was given. Our analysis also showed that the hardest instances occur at specific values of the tardiness factor and of the relative due date, the parameters which were used in the randomized instance generation suggesting the existence of a kind of phase transition for the SMTWTP like those observed in many other combinatorial optimization problems [14]. There are several important reasons for the very good performance of the ACO approach. Among them are that we use a very effective local search algorithm and additionally a heterogeneous colony of ants which increased the robustness of the algorithm.

In the future we will test our ACS algorithm on larger SMTWTP instances and will extend our successful algorithm to a wider range of single-machine scheduling problems. Additionally, our results using a heterogeneous colony of ants suggest that a similar approach may also yield improvements for ACO algorithms in other applications where a strong instance dependence of the best algorithm configuration has been noted. Similarly, a self-tuning ACO algorithm which uses search feedback to adjust its configuration to the particular instance currently being solved may be interesting.

References

1. T. S. Abdul-Razaq, C. N. Potts, and L. N. Van Wassenhove. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, 26:235–253, 1990.
2. A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss. An ant colony optimization approach for the single machine total tardiness problem. In *Proc. of CEC'99*, pages 1445–1450. IEEE Press, Piscataway, NJ, 1999.
3. A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for Job-Shop Scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.
4. R. K. Congram, C. N. Potts, and S. L. Van de Velde. An iterated dynasearch algorithm for the single–machine total weighted tardiness scheduling problem. Technical report, Faculty of Mathematical Studies, University of Southampton, December 1998.
5. H. A. J. Crauwels, C. N. Potts, and L. N. Van Wassenhove. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10(3):341–350, 1998.
6. F. Della Croce, R. Tadei, P. Baracco, and A. Grosso. A new decomposition approach for the single machine total tardiness scheduling problem. *Journal of the Operational Research Society*, 49:1101–1106, 1998.
7. G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.
8. M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, 1999.

9. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
10. M. Dorigo and L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
11. L. M. Gambardella and M. Dorigo. Ant Colony System hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3), 2000.
12. L. M. Gambardella, È. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw Hill, 1999.
13. L. M. Gambardella, È. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50:167–176, 1999.
14. T. Hogg, B. A. Huberman, and C. P. Williams (Guest Editors). Special volume on frontiers in problem solving: Phase transitions and complexity. *Artificial Intelligence*, 81(1–2), 1996.
15. H. H. Hoos and T. Stützle. Evaluating Las Vegas algorithms — pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence 1998*, pages 238–245. Morgan Kaufmann Publishers, 1998.
16. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problem. In P. L. Hammer, E. L. Johnson, B. H. Korte, and Nemhauser G. L., editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 343–362. North-Holland, Amsterdam, NL, 1977.
17. V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.
18. H. Matsuo, C. J. Suh, and R. S. Sullivan. A controlled search simulated annealing method for the single machine weighted tardiness problem. Working paper 87-12-2, Department of Management, University of Texas at Austin, TX, 1987.
19. N. Mladenović and P. Hansen. Variable Neighborhood Search. *Computers & Operations Research*, 24:1097–1100, 1997.
20. T. E. Morton, R. M. Rachamadugu, and A. Vepsalainen. Accurate myopic heuristics for tardiness scheduling. GSIA Working Paper 36-83-84, Carnegie–Mellon University, PA, 1984.
21. M. Pinedo. *Scheduling — Theory, Algorithms, and Systems*. Prentice Hall, 1995.
22. C. N. Potts and L. N. Van Wassenhove. Single machine tardiness sequencing heuristics. *IIE Transactions*, 23:346–354, 1991.
23. C. R. Reeves. Landscapes, operators and heuristic search. To appear in *Annals of Operations Research*, 2000.
24. T. Stützle. An Ant Approach to 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, 1997.
25. T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications*. PhD thesis, Darmstadt University of Technology, Department of Computer Science, 1998.
26. T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 33–50. McGraw Hill, 1999.
27. T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.