Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computers & Industrial Engineering 48 (2005) 789–797
computers & industrial engineering
www.elsevier.com/locate/dsw

Two ant-colony algorithms for minimizing total flowtime in permutation flowshops

Chandrasekharan Rajendran[a], Hans Ziegler[b,*]

[a]*Department of Management Studies, Indian Institute of Technology Madras, Chennai 600 036, India*
[b]*Faculty of Business Administration and Economics, Chair for Production, Operations and Logistics Management, University of Passau, Innstrasse 39, 94032 Passau, Germany*

Abstract

The problem of scheduling in flowshops with the objective of minimizing total flowtime is studied. For solving the problem two ant-colony algorithms are proposed and analyzed. The first algorithm refers to some extent to ideas by Stuetzle [Stuetzle, T. (1998). An ant approach for the flow shop problem. In: *Proceedings of the sixth European Congress on intelligent techniques and soft computing (EUFIT '98)* (Vol. 3) (pp. 1560–1564). Aachen: Verlag Mainz] and Merkle and Middendorf [Merkle, D., & Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In: *Proceedings of the EvoWorkshops 2000, lecture notes in computer science 1803* (pp. 287–296). Berlin: Springer]. The second algorithm is newly developed. The proposed ant-colony algorithms have been applied to 90 benchmark problems taken from Taillard [Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278–285]. A comparison of the solutions yielded by the ant-colony algorithms with the best heuristic solutions known for the benchmark problems up to now, as published in extensive studies by Liu and Reeves [Liu, J., & Reeves, C.R. (2001). Constructive and composite heuristic solutions to the $P//\Sigma C_i$ scheduling problem. *European Journal of Operational Research*, *132*, 439–452, and Rajendran and Ziegler [Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, *155*, 426–438], shows that the presented ant-colony algorithms are better, on an average, than the heuristics analyzed by Liu and Reeves and Rajendran and Ziegler.

Keywords: Flowshop; Scheduling; Total flowtime; Ant-colony algorithm

* Corresponding author. Fax: +49 851 509 2452.
E-mail addresses: craj@iitm.ac.in (C. Rajendran), ziegler@uni-passau.de (H. Ziegler).

0360-8352/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2004.12.009

1. Introduction

For solving flowshop scheduling problems with different objectives many exact algorithms and problem-specific traditional heuristics have been proposed (e.g. Campbell, Dudek, & Smith, 1970; Gelders & Sambandam, 1978; Ho, 1995; Ignall & Schrage, 1965; Johnson, 1954; Liu & Reeves, 2001; Miyazaki & Nishiyama, 1980; Miyazaki, Nishiyama, & Hashimoto, 1978; Nawaz, Enscore, & Ham, 1983; Rajendran, 1993; Rajendran, 1995; Rajendran & Ziegler, 1997; Woo & Yim, 1998). In addition, metaheuristics such as genetic algorithms, simulated annealing and tabu search have been developed (e.g. Ben-Daya & Al-Fawzan, 1998; Ishibuchi, Misaki, & Tanaka, 1995; Nowicki & Smutnicki, 1996; Widmer & Hertz, 1989). In recent times, attempts are being made to solve combinatorial optimization problems by making use of ant-colony-optimization algorithms (ACO algorithms). For solving scheduling problems ACO algorithms have been proposed only recently (e.g. MMAS by Stuetzle (1998), and PACO by Rajendran and Ziegler (2004) for minimizing the makespan/total flowtime in flowshops, and Merkle and Middendorf (2000) for minimizing total weighted tardiness for the single machine problem).

In this paper, the problem of scheduling in flowshops with the objective of minimizing total flowtime by using ant-colony algorithms is investigated. The first algorithm presented relates to ideas of Stuetzle (1998) and Merkle and Middendorf (2000), and the second one is based on new ideas. We evaluate the ACO algorithms in a comparison with the best heuristic solutions reported by Liu and Reeves (2001), and Rajendran and Ziegler (2004) for 90 benchmark flowshop scheduling problems taken from Taillard (1993).

2. The permutation flowshop scheduling problem

The flowshop scheduling problem consists in scheduling n jobs with given processing times on m machines, where the sequence of processing a job on all machines is identical for each job. Without loss of generality the route of the jobs is equal to 1, 2,…, m. It is assumed that each job can be processed on only one machine at a time and that each machine can process only one job at a time. Furthermore, the operations are not preemptable, the jobs are available for processing at time 0 and the setup times are sequence independent. In studying flowshop scheduling problems, it is a common assumption that the sequence in which each machine processes all jobs is identical on all machines (permutation flowshop). A schedule of this type is called a permutation schedule and is defined by a complete sequence of all jobs. We consider only permutation schedules in this paper.

Assuming that each operation is to be performed as soon as possible, for a given sequence of jobs the completion or finishing times of the operations can be found as follows. Let t_{ij} be the processing time of job i on machine j, n be the total number of jobs to be scheduled and m be the total number of machines in the flowshop. σ denotes the partial sequence of jobs already scheduled, $c(\sigma, j)$ denotes the completion time of partial sequence σ on machine j and $c(\sigma i, j)$ denotes the completion time of job i on machine j when job i is appended to partial sequence σ. The completion times of each job i on the machines are calculated by the recursive equations: $c(\sigma i, j) = \max\{c(\sigma, j);\ c(\sigma i, j-1)\} + t_{ij}, \quad j = 1, \ldots, m$, where $c(\varnothing, j) = 0$ for all j and $c(\sigma i, 0) = 0$ for all i. The completion time or flowtime of job i, C_i, is given by $C_i = c(\sigma i, m)$. When all jobs are scheduled, the total flowtime F is obtained by $F = \sum_{i=1}^{n} C_i$. In this paper, the objective of minimizing total flowtime is considered.

3. Description of the ant-colony algorithms

3.1. General structure of the algorithms

The main idea in ant-colony algorithms is to mimic the pheromone trail used by real ants searching for feed as a medium for communication and feedback. The pioneering work has been done by Dorigo (1992), an introduction to the ACO algorithms has been dealt with in Dorigo, Maniezzo, and Colorni (1996). Basically, ACO algorithms are population-based, cooperative search procedures. They make use of simple agents called ants that iteratively construct solutions to the problem considered. The construction of solutions is guided by (artificial) pheromone trails and problem-specific heuristic information. In the context of combinatorial optimization problems, pheromones indicate the intensity of ant trails with respect to solution components, and such trails are determined on the basis of the contribution to the objective function. An individual ant constructs a complete solution by starting with a null solution and iteratively adding solution components, using the trail intensities for making its choices, until a complete solution is built. Usually an improvement scheme (local search) is applied to a solution constructed this way. After the construction and possibly improvement of a complete solution, every ant gives feedback on the solution by depositing pheromone on each solution component (i.e. updating trail intensity). In addition old pheromone evaporates to a certain extent. Typically, solution components which are part of better solutions (or are used by ants over many iterations) will hold a higher amount of pheromone, and hence, such solution components are more likely to be used by the ants in future iterations of the ACO algorithm. In each iteration of an ACO algorithm one can use many ants in parallel or only one ant. At present the prevailing implementation of most ACO algorithms is using only one ant in each iteration.

The general structure of ACO algorithms can be described as follows:

```
Step l: Initialize the pheromone trails and the parameters.
Step 2: While (termination condition is not met) do:
            construct a solution;
            improve the solution by local search;
            update the pheromone trail intensities
               and in case of an improved solution update the best solution found
               and the corresponding objective value as well.
Step 3: Return the best solution found.
```

The trails form a kind of adaptive memory of previously found solutions and are modified at the end of each iteration. In the context of applying ACO algorithms to scheduling problems, the trail intensities represent the desire of setting a certain job in a certain position of a sequence. Hence a pheromone value is stored and updated in each iteration for each possible position k in the sequence for each job i. Therefore, n^2 pheromone values have to be stored. Although more than one ant can be used in each iteration, the ACO algorithms proposed in this paper make use of only one ant.

3.2. ACO1

The first algorithm, ACO1, refers to some extent to ideas of MMAS proposed by Stuetzle (1998), and the ant-colony algorithm of Merkle and Middendorf (2000) who considered the single-machine

total-weighted-tardiness problem. The parameter setting of ACO1 follows the MMAS of Stuetzle (1998), and M-MMAS of Rajendran and Ziegler (2004). In the procedure for selecting a job to be appended to a partial ant-sequence the summation rule developed by Merkle and Middendorf (2000) has been incorporated. In addition, a different procedure for generating the seed sequence for ACO1 and a new technique for local search (job-index-based local search) is used.

3.2.1. Initialization of the pheromone trails and the parameters

Let τ_{ik} the trail intensity of setting job i in position k of a sequence. For initializing the trail intensities an initial sequence is necessary. This sequence is generated by using Rajendran's (1993) heuristic. First the jobs are arranged in the ascending order of their $\sum_{j=1}^{m}(m-j+1)\times t_{ij}$ values. Then an improvement scheme proposed by Nawaz et al. (1983) is applied. To the sequence found this way a job-index-based local search procedure (see Section 3.2.2) is applied three times. The final sequence thus obtained is the seed sequence for the ACO.

According to Stuetzle (1998) the parameters of ACO1 are chosen as follows. Using the seed sequence, limits $\tau_{\max}$ and $\tau_{\min}$ for the trail intensities are calculated by $\tau_{\max} = 1/((1-\rho)\times F_{\text{best}})$ and $\tau_{\min}=\tau_{\max}/5$, where ρ denotes the persistence of trail ($1-\rho$ being the evaporation rate) and F_{best} the best objective value obtained so far, i.e. the one of the seed sequence. ρ is set to 0.75 in all iterations. The initial trail intensities are chosen as $\tau_{ik}=\tau_{\max}$ for all i and all k.

3.2.2. Construction of a solution

Starting from a null sequence, ACO algorithms make use of trail intensities in order to determine the job to be appended next in position k, $k=1,2,\ldots,n$. As indicated earlier, the trail intensity τ_{ik} denotes the 'desire' of placing job i in position k of a sequence. Even though this trail intensity changes with respect to every iteration in the ant-colony algorithm, the iteration counter is omitted for the sake of simplicity of presentation. An ant starts constructing a sequence by choosing a job for the first position, followed by the choice of an unscheduled job for the second position, and so on. A dummy job 0 is introduced on which an ant is set initially, and the construction of partial sequences begins, thereby leading to the build-up of a complete sequence, called an ant-sequence. In the case of ACO1, the following procedure is used to choose an unscheduled job i probabilistically for position k:

Set $T_{ik}=\sum_{q=1}^{k}\tau_{iq}$.
Sample a uniform random number u in the range [0, 1].
If $u\leq(n-4)/n$
  then among the first five unscheduled jobs as present in the best sequence obtained so far, choose the job with the maximum value of T_{ik};
  else select job i from the set of the first five unscheduled jobs as present in the best sequence obtained so far by sampling from the probability distribution $p_{ik}=(T_{ik}/\sum_{l}T_{lk})$, where job l belongs to the set of the first five unscheduled jobs, as present in the best sequence obtained so far, and p_{ik} denotes the probability of choosing job i for position k.
  When there are less than five jobs unscheduled, then all such jobs are considered.

The rationale behind the summation of trail intensities from the first to the kth position is that the choice of the job for appending is based on the pheromone values up to position k, instead of being based on the pheromone value with respect to position k. This gives a better estimate of the desirability of placing a job

in position k, as against the actual pheromone value with respect to position k. In other words, the sum of pheromone values of job i from position 1 up to position k indicates the 'need' or 'desire' to schedule job i not later than position k. In case of selecting job i from the set of the first five unscheduled jobs as present in the best sequence randomly, the probabilities are chosen proportional to the cumulative pheromone values T_{ik}. Selecting the job for appending on basis of the summation rule, i.e. using T_{ik} instead of using τ_{ik} (as used by Stuetzle, 1998), leads to a high exploration of jobs with high values of τ_{ik} up to position k in the present best solution, and the restriction to select between the first five unscheduled jobs as present in the best sequence obtained so far leads to a good performing neighborhood.

Having thus generated a complete sequence of jobs (ant-sequence), a newly proposed job-index-based local search procedure is applied three times in order to improve the solution. Let $[k]$ denote the index of the job at position k of the current seed sequence, then the local search procedure can be described as follows:

For $i = 1(1)n$:
  For $k = 1(1)n$:
    If $[k] \neq i$ then insert job i in position k of the current seed sequence and adjust the sequence accordingly by not changing the relative position of the other jobs;
    calculate the total flowtime of the modified sequence.
  Choose the best sequence among such $n-1$ modified sequences. If the total flowtime is improved, then replace the current seed sequence by the best one found.

The ant-sequence is used as initial seed sequence. A good feature of this local search procedure is that it is not guided or biased by the job ordering in any sequence, and hence, the search may not get entrapped in local optima early. We have experimented with some different improvement schemes and found the one presented to be the best one. Furthermore, applying the improvement scheme more than three times yielded only a very slight additional improvement. Hence, taking into account the additional computational effort, applying the local search three times is to be preferred.

3.2.3. Updating of trail intensities

Updating of the trail intensities is based on the solution present after the application of the job-index-based local search procedure on the ant-sequence. Let the objective function value of this improved sequence found in Section 3.2.2 be denoted by F_{new}, then the trail intensities are up-dated as follows:

If job i is placed in position k, then $\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}} + (1/F_{\text{new}})$; else $\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}}$.

In case of $\tau_{ik}^{\text{new}} > \tau_{\max}$ or $\tau_{ik}^{\text{new}} < \tau_{\min}$, the trail intensity τ_{ik}^{new} is set to $\tau_{\max}$ or $\tau_{\min}$, respectively.

The idea behind the setting of trail intensity is that positions which are often occupied by certain jobs receive a 'higher amount of pheromone', and hence those jobs will be placed in the corresponding positions with higher probability.

If the sequence generated according to Section 3.2.2 is superior to the best sequence that has been obtained so far, the best sequence and the best objective function value F_{best} are updated. Furthermore, using F_{best} the limits $\tau_{\max}$ and $\tau_{\min}$ are updated as well and the new limits have to be obeyed by the updated trail intensities τ_{ik}^{new} for all i and k.

Various experiments performed have shown that the generation of 40 ant sequences yielded the best trade-off between performance of ACO1 and computational time. Therefore, the termination condition has been chosen to be 40 iterations.

3.3. ACO2

ACO2 is quite different from ACO1 in each of the components of the algorithm.

3.3.1. Initialization of the pheromone trails and the parameters

Initially, all τ_{ik} are set to $(1/F_{\text{best}})$ and ρ is set to 0.75 throughout, where F_{best} is the total flowtime of the seed sequence determined as described in Section 3.2.1. On the trail intensities no limits are imposed.

3.3.2. Construction of a solution

In ACO2, the following procedure is used to choose an unscheduled job i for position k:

Sample a uniform random number u in the range [0, l].
If $u \leq 0.4$
then choose the first unscheduled job as present in the best sequence obtained so far;
else
if $u \leq 0.8$
then set $T_{ik} = \sum_{q=1}^{k} \tau_{iq}$;
among the set of the first five unscheduled jobs, as present in the best sequence obtained so far, choose the job with the maximum value of T_{ik};
else select job i from the same set of five unscheduled jobs by sampling from the probability distribution $p_{ik} = (T_{ik}/\sum_l T_{lk})$, where job l belongs to the set of the first five unscheduled jobs, as present in the best sequence obtained so far.
Note that when there are less than five jobs unscheduled, then all such unscheduled jobs are considered.

A complete sequence is generated accordingly. We have experimented with different probability ranges for the three cases and found those presented to be the best. The rationale behind the probability ranges chosen is that the choice between the first unscheduled job in the present best sequence and the one from the first five unscheduled jobs in the present best sequence having the best value of T_{ik} is done with equal probability and that the probabilistic choice from the first five unscheduled jobs in the present best sequence is done with half of the probability for each of the other two possibilities.

The sequence constructed is subjected to the job-index-based local search procedure described in Section 3.2.2 three times in order to improve the solution. The total flowtime of the sequence present after the application of the improvement procedure is denoted by F_{new}.

3.3.3. Updating of trail intensities

Based on the solution present after the application of the job-index-based local search procedure, the trail intensities are updated as follows:

If job i is placed in position k, then $\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}} + (1/(\text{diff}_i \times F_{\text{new}}))$; else $\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}}$;
with $\text{diff}_i = (|\text{position of job } i \text{ in the best sequence obtained so far} - k| + 1)^{1/2}$.

diff_i is a measure for the distance between the position of job i in the best sequence obtained so far and its position in the sequence generated in the current iteration. The reason for the above setting for updating the trail intensities is that jobs occupying in the generated ant-sequence a position closer to

their respective position in the best sequence obtained so far should get their trail intensities increased by a larger value than jobs occupying a position farther from their respective position in the best sequence.

If the sequence generated according to Section 3.3.2 is superior to the best sequence that has been obtained so far, the best sequence and the best objective function value are up dated.

As in ACO1, the termination condition for ACO2 has been chosen to be 40 iterations. The best heuristic sequence thus obtained is finally subjected to the job-index based swap scheme, applied only once (as in the case of PACO).

4. Performance analysis of the algorithms

Liu and Reeves (2001) developed a couple of new heuristics and compared these with a number of existing heuristics (e.g. the heuristics by Ho, 1995; Rajendran & Ziegler, 1997; Woo & Yim, 1998). They considered the benchmark problems of Taillard (1993) and reported the best heuristic solutions with respect to total flowtime objective. It is to be noted that no single heuristic has emerged to be the best for all benchmark problems. Rajendran and Ziegler (2004) proposed an ant-colony algorithm, called PACO. In order to compare the solutions yielded by the ACO1 and ACO2 algorithm with the best heuristic solutions reported by Liu and Reeves, and those by PACO, we use the relative percentage increase in total flowtime. For a given problem, let the total flowtime of the best-heuristic solution, among all solutions reported by Liu and Reeves, be denoted by $F1$, and the total flowtime yielded by PACO, ACO1 and ACO2 be denoted by $F2$, $F3$, and $F4$, respectively. Then the relative percentage increase in total flowtime of the solution yielded by approach a is given by $(Fa - \min\{Fb;\ b = 1, 2, 3, 4\}) \times 100/\min\{Fb;\ b = 1, 2, 3, 4\}$.

The mean values over 10 different problems of a given size ($m \times n$) are reported in Table 1. In addition, the maximum values of relative percentage increase in total flowtime are stated. It is evident from the table that the ACO1 and the ACO2 algorithm yield solutions of superior quality, as against

Table 1
Relative performance of the heuristic procedures, measured in terms of mean and maximum relative percentage increase in total flowtime with respect to the best heuristic solution

m	n	BEST(LR)		PACO		ACO1		ACO2	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
5	20	1.214	2.128	0.309	0.848	0.052	0.187	0.206	0.766
	50	0.771	1.423	0.170	0.789	0.353	0.635	0.174	0.595
	100	0.205	0.756	0.278	0.809	0.160	0.435	0.199	0.668
10	20	1.394	2.170	0.285	0.681	0.011	0.105	0.395	0.826
	50	1.437	2.238	0.203	1.121	0.452	1.167	0.430	1.047
	100	0.921	1.671	0.231	0.684	0.516	0.998	0.088	0.637
20	20	1.155	1.863	0.121	0.619	0.051	0.352	0.311	1.083
	50	1.739	3.280	0.126	0.659	0.431	0.867	0.219	0.838
	100	1.653	3.068	0.170	0.573	0.401	1.346	0.109	0.489

Sample size in every problem set is 10, generated by the procedure from Taillard (1993). BEST(LR) refers to the best performing heuristic as investigated by Liu and Reeves (2001), and PACO refers to the algorithm by Rajendran and Ziegler (2004). ACO1 and ACO2, respectively, solved all problems in less than 1 h on a Pentium 3 computer with 800 MHz using FORTRAN and the operating system Windows NT 4.0.

the best-heuristic solutions reported by Liu and Reeves, on an overall basis. Furthermore, for larger-sized problems ACO2 is to be preferred to ACO1 and PACO. ACO1 achieved an improvement of the total flowtime up to 2.74% and ACO2 up to 3.47% in comparison to the best heuristic solution in Liu and Reeves.

5. Summary

Of late, attempts are being made to solve combinatorial optimization problems by making use of ant-colony-optimization algorithms. Compared to other metaheuristics such as genetic algorithms, simulated annealing and tabu search, relatively few attempts have been made to solve scheduling problems using ant-colony algorithms. In this paper, the problem of scheduling in flowshops with the objective of minimizing total flowtime has been investigated. Two ant-colony algorithms have been proposed and evaluated by making use of benchmark problems. It has been found that the ant-colony algorithms yield solutions of superior quality as against the best known heuristic solutions reported in recent research studies. The results of this study encourage the development of ant-colony algorithms for various scheduling problems with different objectives.

Acknowledgements

The first author gratefully acknowledges the Research Fellowship of Alexander-von-Humboldt Foundation for carrying out this work. The authors are thankful to the reviewers and the Editor for their suggestions to improve the earlier versions of the paper.

References

Ben-Daya, M., & Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, *109*, 88–95.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*, *16*, B630–B637.

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms (in Italian). PhD Thesis. Milano, Italy: Dipartimento di Elettronica, Politecnico di Milano.

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

Gelders, L. F., & Sambandam, N. (1978). Four simple heuristics for scheduling a flowshop. *International Journal of Production Research*, *16*, 221–231.

Ho, J. C. (1995). Flowshop sequencing with mean flow time objective. *European Journal of Operational Research*, *81*, 571–578.

Ignall, E., & Schrage, L. (1965). Application of the branch-and-bound technique to some flowshop scheduling problems. *Operations Research*, *13*, 400–412.

Ishibuchi, H., Misaki, S., & Tanaka, H. (1995). Modified simulated annealing algorithms for the flow shop sequencing problems. *European Journal of Operational Research*, *81*, 388–398.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules. *Naval Research Logistics Quarterly*, *1*, 61–68.

Liu, J., & Reeves, C. R. (2001). Constructive and composite heuristic solutions to the P//ΣC_i scheduling problem. *European Journal of Operational Research*, *132*, 439–452.

Merkle, D., & Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In: *Proceedings of the EvoWorkshops 2000, lecture notes in computer science 1803* (pp. 287–296). Berlin: Springer.

Miyazaki, S., & Nishiyama, N. (1980). Analysis for minimizing weighted mean flowtime in flowshop scheduling. *Journal of the Operations Research Society of Japan*, *23*, 118–132.

Miyazaki, S., Nishiyama, N., & Hashimoto, F. (1978). An adjacent pairwise approach to the mean flowtime scheduling problem. *Journal of the Operations Research Society of Japan*, *21*, 287–299.

Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *OMEGA*, *11*, 91–95.

Nowicki, E., & Smutnicki, C. (1996). A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, *91*, 160–175.

Rajendran, C. (1993). Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*, *29*, 65–73.

Rajendran, C. (1995). Heuristics for scheduling in flowshop with multiple objectives. *European Journal of Operational Research*, *82*, 540–555.

Rajendran, C., & Ziegler, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research*, *103*, 129–138.

Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, *155*, 426–438.

Stuetzle, T. (1998). An ant approach for the flow shop problem. In: *Proceedings of the sixth European Congress on intelligent techniques and soft computing (EUFIT '98)* (Vol. 3) (pp. 1560–1564). Aachen: Verlag Mainz.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278–285.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flowshop sequencing problem. *European Journal of Operational Research*, *41*, 186–193.

Woo, H. S., & Yim, D. S. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers and Operations Research*, *25*, 175–182.