Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 155 (2004) 426–438
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs

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

[a] *Industrial Engineering and Management Division, Department of Humanities and Social Sciences, Indian Institute of Technology Madras, Chennai 600 036, India*
[b] *Faculty of Business Administration and Economics, Department of Operations, Production and Logistics Management, University of Passau, D-94032 Passau, Germany*

Received 18 October 2001; accepted 18 November 2002

Abstract

The problem of scheduling in permutation flowshops is considered with the objective of minimizing the makespan, followed by the consideration of minimization of total flowtime of jobs. Two ant-colony optimization algorithms are proposed and analyzed for solving the permutation flowshop scheduling problem. The first algorithm extends the ideas of the ant-colony algorithm by Stuetzle [Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT '98), vol. 3, Verlag Mainz, Aachen, Germany, 1998, p. 1560], called max–min ant system (MMAS), by incorporating the summation rule suggested by Merkle and Middendorf [Proceedings of the EvoWorkshops 2000, Lecture Notes in Computer Science No. 1803, Springer-Verlag, Berlin, 2000, p. 287] and a newly proposed local search technique. The second ant-colony algorithm is newly developed. The proposed ant-colony algorithms have been applied to 90 benchmark problems taken from Taillard [European Journal of Operational Research 64 (1993) 278]. First, a comparison of the solutions yielded by the MMAS and the two ant-colony algorithms developed in this paper, with the heuristic solutions given by Taillard [European Journal of Operational Research 64 (1993) 278] is undertaken with respect to the minimization of makespan. The comparison shows that the two proposed ant-colony algorithms perform better, on an average, than the MMAS. Subsequently, by considering the objective of minimizing the total flowtime of jobs, a comparison of solutions yielded by the proposed ant-colony algorithms with the best heuristic solutions known for the benchmark problems, as published in an extensive study by Liu and Reeves [European Journal of Operational Research 132 (2001) 439], is carried out. The comparison shows that the proposed ant-colony algorithms are clearly superior to the heuristics analyzed by Liu and Reeves. For 83 out of 90 problems considered, better solutions have been found by the two proposed ant-colony algorithms, as compared to the solutions reported by Liu and Reeves.

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

1. Introduction

Many exact and heuristic algorithms have been proposed over the years for solving the static permutation flowshop scheduling problem with the

* Corresponding author. Tel.: +91-44-22578431; fax: +91-44-22578430.
E-mail address: craj@iitm.ac.in (C. Rajendran).

0377-2217/$ - see front matter.
doi:10.1016/S0377-2217(02)00908-6

objectives of minimizing makespan and total flowtime of jobs, considered either separately or simultaneously (e.g. Johnson, 1954; Ignall and Schrage, 1965; Campbell et al., 1970; Gelders and Sambandam, 1978; Miyazaki et al., 1978; Miyazaki and Nishiyama, 1980; Nawaz et al., 1983; Rajendran, 1993, 1995; Ho, 1995; Wang et al., 1997; Woo and Yim, 1998; Liu and Reeves, 2001). In addition, metaheuristics such as genetic algorithms, simulated annealing and tabu search are used to solve flowshop scheduling problems (e.g. Widmer and Hertz, 1989; Ishibuchi et al., 1995; Nowicki and Smutnicki, 1996; Ben-Daya and Al-Fawzan, 1998).

In recent times, attempts are being made to solve combinatorial optimization problems by making use of ant-colony optimization (ACO) algorithms (or simply, ant-colony or ACO algorithms). The pioneering work has been done by Dorigo (1992), and an introduction to the ACO algorithms has been dealt with in Dorigo et al. (1996). Attempts have been made recently to solve scheduling problems using ACO algorithms (e.g. Stuetzle, 1998 dealing with permutation flowshop scheduling problem with the objective of minimizing the makespan, and Merkle and Middendorf, 2000 considering the single-machine scheduling problem with the objective of minimizing the sum of weighted tardiness of jobs).

In this paper, we investigate the problem of scheduling in permutation flowshops by using ACO algorithms, first with the consideration of minimizing the makespan, followed by the consideration of minimizing the sum of total flowtime of jobs. Two ant-colony algorithms are proposed in this paper. The first algorithm incorporates the concept of summation rule (presented by Merkle and Middendorf, 2000), a modification with respect to the selection of the job to be appended to the partial ant-sequence, and a new local search technique in the max–min ant system (MMAS) which is an ant-colony algorithm developed by Stuetzle (1998). The second proposed ant-colony algorithm is based on new ideas that are developed in the current work. By considering the objective of minimizing the makespan, we evaluate the ACO algorithms in comparison with the upper bound values of makespan presented by Taillard (1993) for the 90 benchmark permutation flowshop scheduling problems taken from Taillard (1993). Next, we consider the objective of minimizing the total flowtime of jobs and evaluate the solutions yielded by the two proposed ACO algorithms relative to the best heuristic solutions reported by Liu and Reeves (2001) for the same 90 benchmark permutation flowshop problems.

2. Formulation of the static permutation flowshop scheduling problem

The permutation 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 and unidirectional for each job. 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 also consider only permutation sequences in this work.

Let

t_{ij} be the processing time of job i on machine j;
n be the total number of jobs to be scheduled;
m be the total number of machines in the flowshop;
σ be the ordered set of jobs already scheduled, out of n jobs; partial sequence;
$q(\sigma, j)$ be the completion time of partial sequence σ on machine j (i.e. the release time of machine j after processing all jobs in partial sequence σ);
$q(\sigma i, j)$ be the completion time of job i on machine j, when the job is appended to partial sequence σ.

For calculating the start and completion times of jobs on machines in permutation flowshops, recursive equations are used as follows.

Initialize $q(\sigma i, 0)$, the completion time of job i on machine 0, equal to zero. This time indicates the time of availability of a job in the flowshop, and it is equal to 0 for all jobs in case of static flowshops.

For $j = 1$ to m do

$$q(\sigma i, j) = \max\{q(\sigma, j); q(\sigma i, j-1)\} + t_{ij}. \quad (1)$$

The flowtime of job i, C_i is given by

$$C_i = q(\sigma i, m). \quad (2)$$

When all jobs are scheduled, the total flowtime F, and the makespan M are obtained as follows:

$$F = \sum_{i=1}^{n} C_i,$$

and

$$M = \max\{C_i, i = 1, 2, \ldots, n\}. \quad (3)$$

It is to be noted that $q(\phi, j)$ is equal to 0 for all j, where ϕ denotes a null schedule.

3. Description of the proposed ACO algorithms

3.1. General structure of ant-colony algorithms

The main idea in ant-colony algorithms is to mimic the pheromone trail used by real ants as a medium for communication and feedback among ants. Basically, the ACO algorithm is a population-based, cooperative search procedure that is derived from the behavior of real ants. ACO algorithms make use of simple agents called *ants* that iteratively construct solutions to combinatorial optimization problems. The solution generation or construction by ants is guided by (artificial) pheromone trails and problem-specific heuristic information. ACO algorithms can be applied to combinatorial optimization problems by defining solution components which the ants use to iteratively construct solutions and in the process, the ants deposit pheromone. Basically, in the context of combinatorial optimization problems, pheromones indicate the intensity of ant-trails with respect to solution components, and such intensities are determined on the basis of the influence or contribution of each solution component with respect to the objective function. An individual ant constructs a complete solution by starting with a null solution and iteratively adding solution components until a complete solution is constructed. After the construction of a complete solution, every ant gives feedback on the solution by depositing pheromone (i.e., updating trail intensity) on each solution component. Typically, solution components which are part of better solutions or used by ants over many iterations will receive 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. This is achieved by additionally making use of pheromone evaporation in updating trail intensities (Stuetzle and Hoos, 2000).

It is to be noted that one can make use of only one ant in every iteration of the ACO algorithm, instead of using many ants in parallel in one iteration. In the former case, the number of iterations in the algorithm may increase, while the latter case may result in increased computational complexity. The consideration of parallel ants in one iteration is highly relevant in the context of parallel-processor computing systems because the updation of pheromone trails due to one ant can be reflected on the intensity of trails of other ants working in parallel. On the other hand, the consideration of one single ant renders the task of implementing the ACO algorithm on a single-process computing system easy and less complex. In addition, a complete solution that has been constructed by a single ant can be subjected to an improvement or local search scheme to unearth possibly the best solution in the neighborhood. Such a consideration of a single ant in ACO algorithms is quite common (e.g. Stuetzle, 1998). In the current work, the proposed ant-colony algorithms make use of a single ant in every iteration.

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

Step 1: Initialize the pheromone trails and parameters.
Step 2: While (termination condition is not met) do the following:
construct a solution;
improve the solution by local search;
update the pheromone trail or trail intensities.
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 application of ACO algorithms to scheduling problems, τ_{ik} denotes the trail intensity (or desire) of setting job i in position k of a sequence. It is to be noted that for every job i for any possible position k, a pheromone value is stored and updated in each iteration of the ACO algorithm. Hence there are (number of jobs)^2 such values of τ_{ik}.

3.2. Description of the first proposed ant-colony algorithm (M-MMAS)

The first algorithm, called M-MMAS, proposed in this paper extends the ant-colony algorithm of Stuetzle (1998), called MMAS, by incorporating the summation rule developed by Merkle and Middendorf (2000) for the single-machine total-weighted-tardiness problem, and by modifying the procedure for the selection of the job to be appended to the partial ant-sequence. In addition, the M-MMAS makes use of a new local search technique, called job-index-based local search (see Section 3.2.2 for details of this technique), instead of the modified first-move strategy used by Stuetzle.

3.2.1. Initialization of parameters in the M-MMAS

For initializing the trail intensities, an initial sequence is necessary. The initial sequence is generated depending upon the objective function under consideration. In this paper the Nawaz, Enscore and Ham (NEH) (1983) heuristic is used to generate an initial sequence for the objective of minimizing the makespan, and Rajendran's (1993) heuristic is used to generate an initial sequence for the objective of minimizing the total flowtime of jobs in permutation flowshops. Both these heuristics are very simple to be implemented and fast to be executed. The resulting sequence from one of these heuristics, depending upon the objective function under consideration, is subjected to the proposed job-index-based local search technique, applied three times (see Section 3.2.2). The final sequence thus obtained is the seed sequence to the M-MMAS. We compute the limits $\tau_{\max}$ and $\tau_{\min}$ for the trail intensities τ_{ik} by setting $\tau_{\max} = 1/((1-\rho)Z_{\text{best}})$ and $\tau_{\min} = \tau_{\max}/5$, where ρ denotes the persistence of trail ($1-\rho$ being the evaporation rate), and Z_{best} denotes the best objective-function value obtained so far. Initially, Z_{best} equals the objective-function value yielded by 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 k.

3.2.2. Construction of an ant-sequence and its improvement by a local search scheme

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, where $1 \leqslant k \leqslant n$. As indicated earlier, the trail intensity, τ_{ik}, denotes the 'desire' of placing job i in position k in 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, an ant-sequence. In the case of the M-MMAS, the following procedure is used to choose an unscheduled job, say job i, probabilistically for position k:

$$\text{Set } T_{ik} = \sum_{q=1}^{k} \tau_{iq}. \tag{4}$$

Sample a uniform random number u in the range [0, 1].
If $u \leqslant (n-4)/n$
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
job i is selected, from the set of first five unscheduled jobs as present in the best sequence obtained so far, for position k by sampling from the following probability distribution:

$$p_{ik} = \left(T_{ik} / \sum_{l} T_{lk} \right), \tag{5}$$

where job l belongs to the set of 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.

The rationale behind the summation of trail intensities 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. Moreover, the summation value of pheromones indicates 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 up to position k indicates the 'need' or 'desire' to schedule job i not later than position k. It is to be noted that when the uniform random number, u, is less than or equal to $(n-4)/n$, we choose the job with the maximum value of T_{ik} among the first five unscheduled jobs as present in the best sequence obtained so far, whereas in the MMAS the job with the maximum value of T_{ik} among all unscheduled jobs is chosen.

Having thus generated a complete ant-sequence or a complete sequence of jobs, 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 proposed 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 positions of the other jobs;
      calculate the value of objective function of the modified sequence.
      Choose the best sequence among such $(n-1)$ modified sequences. If the objective-function value is improved, then replace the current sequence by the best one found.

The ant-sequence is taken as the initial current 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 a couple of local search or improvement schemes such as the modified first-move strategy by Stuetzle (1998) and the overall-seed-sequence-based local search scheme by Rajendran and Ziegler (1997a,b, 1999). From the experimental analysis and its results, we have found that the proposed local search technique, namely, job-index-based local search procedure, performs better than these local search procedures in terms of enhancing the quality of the ant-sequence. The details are not presented here for the sake of restricting the length of the paper.

3.2.3. Updating of trail intensities

Updating of the trail intensities is based on the sequence obtained after the three-time application of the job-index-based local search procedure on the ant-sequence. Let the objective-function value of this sequence be denoted by Z_{current}. The trail intensities are updated as follows:

$$\begin{aligned} \tau_{ik}^{\text{new}} &= \rho \times \tau_{ik}^{\text{old}} + (1/Z_{\text{current}}), \\ &\qquad \text{if job } i \text{ is placed in position } k \\ &\qquad \text{in the generated sequence,} \\ &= \rho \times \tau_{ik}^{\text{old}}, \ \text{otherwise.} \end{aligned} \tag{6}$$

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. Furthermore, evaporation of pheromone reduces the trail intensity of a job with respect to a position which is only seldom occupied by the job.

If the sequence obtained after the three-time application of the job-index-based local search procedure on the ant-sequence is superior to the best sequence that has been obtained so far, then the best sequence, Z_{best}, τ_{max} and τ_{min} are updated accordingly. It is to be noted that in case of updating the limits τ_{max} and τ_{min}, these new limits are immediately applied to all trail intensities τ_{ik}^{new}.

In the M-MMAS, we generate 40 ant-sequences with the possible improvement of every ant-sequence through the three-time application of the job-index-based local search technique, and we finally obtain the best heuristic sequence. We have ensured that the M-MMAS requires almost the same computational time as that of the MMAS (as implemented by us) yielding results comparable to those presented by Stuetzle (1998). We have observed that each of MMAS and M-MMAS requires less than one hour of computational time on a Pentium-3 computer with 800 MHz using FORTRAN for solving the 90 benchmark problems from Taillard (1993).

3.3. Development of a new ACO algorithm (PACO)

We now present the development of the proposed ant-colony algorithm, called the PACO. The PACO is quite different from MMAS in each of the various components of the algorithm.

3.3.1. Initialization of parameters

The seed sequence for the ant-colony algorithm is obtained in a way similar to the M-MMAS, with the objective-function value set to Z_{best}. This seed sequence obtained after the three-time application of the job-index-based local search procedure on the solution yielded by the NEH (1983) heuristic (if the objective is to minimize the makespan), or on the solution yielded by Rajendran's (1993) heuristic (if the objective is to minimize the total flowtime of jobs), is indeed of good quality. Hence, we suggest to have differential setting of the τ_{ik}s initially, instead of the same or uniform setting of the τ_{ik}s, in the PACO as follows:

$$\begin{aligned}
\text{Set } \tau_{ik} &= (1/Z_{best}), \\
&\quad \text{if } (|\text{position of job } i \text{ in the seed} \\
&\quad \text{sequence to the PACO} - k| + 1) \leqslant n/4; \\
&= (1/(2 \times Z_{best})), \\
&\quad \text{if } n/4 < (|\text{position of job } i \text{ in the seed} \\
&\quad \text{sequence to the PACO} - k| + 1) \leqslant n/2; \\
&= (1/(4 \times Z_{best})), \text{ otherwise.}
\end{aligned} \tag{7}$$

The rationale behind this initial differential setting of the τ_{ik}s is that the seed solution to the PACO being good, those positions that are close to the position of job i in the seed sequence should be associated with larger values of the τ_{ik}s than those positions that are away from the position of job i in the seed sequence. In other words, the influence of a good seed sequence is better reflected in such a differential setting of the τ_{ik}s, as opposed to the same setting of the τ_{ik}s, with respect to each position k for job i.

Further, no limits (such as τ_{max} and τ_{min}) are imposed on the τ_{ik}s in the PACO. However, ρ is set to 0.75 in all iterations, as in the case of M-MMAS.

3.3.2. Construction of an ant-sequence and its improvement by the proposed local search

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

Set $T_{ik} = \sum_{q=1}^{k} \tau_{iq}$ and sample a uniform random number u in the range $[0, 1]$.
If $u \leqslant 0.4$
then
the first unscheduled job as present in the best sequence obtained so far is chosen;
else
  if $u \leqslant 0.8$
  then
  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

job i is selected from the same set of five unscheduled jobs for position k as a result of sampling from the following probability distribution:

$$p_{ik} = \left(T_{ik} \Big/ \sum_{l} T_{lk} \right),$$

where job l belongs to the set of 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 ant-sequence is generated accordingly. The resulting sequence is subjected to the job-index-based local search scheme three times to improve the solution. Let the objective-function value of this resultant sequence be denoted by Z_{current}.

The rationale behind the selection of the job to be scheduled next is that the choice is governed between the best sequence and the best value of T_{ik} with equal probability, and the probabilistic choice of the job is done with half of the probability of going in for the first unscheduled job found in the best sequence. Moreover, the experimentation with different probability ranges has shown the proposed range to perform the best.

3.3.3. Updating of trail intensities

In the PACO, updating of the trail intensities is based not only on the resultant sequence obtained after the three-time application of the job-index-based local search procedure on the ant-sequence, but also on the relative distance between a given position and the position of job i in the resultant sequence. The trail intensities are updated as follows.

Let h be the position of job i in the resultant sequence.
If $n \leqslant 40$
then

$$\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}} + (1/(diff \times Z_{\text{current}})), \quad \text{if } |h-k| \text{ is} \leqslant 1;$$
$$= \rho \times \tau_{ik}^{\text{old}}, \text{ otherwise;}$$

else

$$\tau_{ik}^{\text{new}} = \rho \times \tau_{ik}^{\text{old}} + (1/(diff \times Z_{\text{current}})), \quad \text{if } |h-k| \leqslant 2;$$
$$= \rho \times \tau_{ik}^{\text{old}}, \text{ otherwise.}$$

In the above, *diff* is defined as follows:

$$diff = (|\text{position of job } i \text{ in the best sequence obtained so far} - k| + 1)^{1/2}. \tag{8}$$

The above differential setting for those positions close to the position of job i in the resultant sequence is based on the premise that the trail intensities of such positions close to the position of job i should be updated in the same way, as opposed to those positions that are away from the position of job i in the resultant sequence. Of course, this differential setting is also dependent upon the size of flowshop problem, determined primarily by n, the number of jobs. The reasons for the consideration of *diff* in updating trail intensities are that all jobs occupying their respective positions in the resultant sequence should not have their trail intensities increased by the same value (as done normally in ant-colony algorithms), and that the jobs occupying positions in the resultant sequence closer to their respective positions in the best sequence obtained so far should get their trail intensities increased by larger values than the jobs occupying positions farther from their respective positions in the best sequence.

If the sequence obtained after the three-time application of the job-index-based local search procedure on the ant-sequence is superior to the best sequence that has been obtained so far, then the best sequence and Z_{best} are updated accordingly.

In the PACO, we generate 40 ant-sequences (with the possible improvement of every ant-sequence through the three-time application of the job-index-based local search technique), and hence, we obtain the best heuristic sequence. The best heuristic sequence thus obtained is finally subjected to a job-index-based swap scheme, applied only once. In the case of job-index-based swap scheme, we first consider job 1 (i.e., job with

index 1 and not the job in position 1 in the sequence) and swap it with every other job, taken one at a time. The best-swap sequence is compared with the seed sequence, and if such a swap improves the seed sequence, then the seed sequence is updated corresponding to the best-swap sequence; else the seed sequence is retained. The procedure is repeated by considering the swapping of job 2, followed by that of job 3, and so on, up to job n. The sequence thus obtained is the final sequence yielded by the PACO.

It is to be noted that we have undertaken a sensitivity analysis of performance for the PACO by varying different parameters such as those involved in the choice of the job to be appended next, and the updation of trail intensities. As for the permutation flowshop scheduling problem, we have found that the prescribed settings in Sections 3.3.1, 3.3.2, 3.3.3 have resulted in the best performance of the PACO. The complete details are not reported for the sake of concise presentation.

4. Performance analysis of the M-MMAS and PACO

We first present the performance evaluation of the M-MMAS and PACO with respect to minimization of the makespan. The test problems for evaluating the two proposed ant-colony algorithms are generated by following the procedure given by Taillard (1993) for generating benchmark permutation flowshop scheduling problems. These test problems have varying sizes, with the number of jobs varying from 20 to 100, and the number of machines varying from 5 to 20. The problems are solved by the M-MMAS and PACO. The values of makespan given by the M-MMAS and the PACO are relatively evaluated against the upper bound or heuristic values for makespan reported by Taillard (1993) for the benchmark problems. The mean values of such percentage relative increases in makespan yielded by the proposed ant-colony algorithms, relative to the best makespan values reported by Taillard, are noted for every problem size, defined by $(n \times m)$. The results of evaluation are presented in Table 1. In addition, we also report the percentage relative increase in makespan yielded by the MMAS. Overall, it is observed from the results that the M-MMAS and the PACO perform better than the MMAS. The main reasons for such a superior performance of the M-MMAS are due to the use of summation rule and job-index-based local search technique. It is also evident that the PACO performs better, on an average, than the M-MMAS, especially in the case of large-sized problems. This better performance is mainly due to the effectiveness of differential initial setting and differential updation of trail intensities for different positions with respect to a given job. In addition, the choice of the job to be appended next to the existing partial ant-sequence is done in the PACO by making use of the best sequence obtained so far, apart from considering the trail intensities. These factors assume greater significance in enhancing the performance of the ant-colony algorithm in the case of relatively large-sized flowshop

Table 1
Relative performance of three heuristic procedures for the large-sized problems, in comparison with the benchmark solutions given by Taillard (1993), for the makespan-objective

n	m	Mean relative percentage increase in makespan		
		MMAS	M-MMAS	PACO
20	5	0.408	0.762	0.704
	10	0.591	0.890	0.843
	20	0.410	0.721	0.720
50	5	0.145	0.144	0.090
	10	2.193	1.118	0.746
	20	2.475	2.013	1.855
100	5	0.196	0.084	0.072
	10	0.928	0.451	0.404
	20	2.238	1.030	0.985

Notes: 1. Sample size in every problem set is 10 and the total number of problems is 90, generated by using the procedure given by Taillard (1993).
2. A larger value of an entry indicates an inferior performance, relative to the tabu search method of Taillard (1993).
3. MMAS denotes the ant-colony algorithm of Stuetzle, and M-MMAS and PACO refer to the proposed ant-colony algorithms.
4. The computational-time requirement of heuristics under evaluation is adjusted to be the same by altering the number of sequences generated in a heuristic, and all problems referred to above have been solved by any heuristic in less than one hour on a Pentium-3 computer with 800 MHz using FORTRAN.

problems than in the case of relatively small-sized problems.

It is evident that the best values for makespan given by Taillard are better than the heuristic values yielded by the proposed ant-colony algorithms. The primary reason for such a performance is that we have restricted the computational time in the case of the proposed ant-colony algorithms. To explain further, let us consider the number of solutions enumerated by Taillard to arrive at the best-solution values specified in his work. For a 50-job, 20-machine permutation flowshop problem, the total number of iterations was 5×10^4 for each of the three replications (or initial solutions or resolutions) in Taillard's approach. When we follow the best-move strategy in generating solutions in the neighborhood of a sequence in a given iteration, we generate n sequences. In the proposed ant-colony algorithms we generate 40 ant-sequences and the three-time application of the local search procedure on each ant-sequence generates $3 \times n \times (n-1)$ sequences. It means that the best-solution values reported by Taillard were obtained after generating a large number of sequences, when compared to the total number of sequences generated in the proposed ant-colony algorithms. Apart from this factor of computational effort, it is also evident that the number of local searches required for a marginal decrease in makespan is quite high, thereby rendering the task of enhancing the performance of ant-colony algorithms in permutation flowshop scheduling with the makespan-objective quite difficult.

For evaluating the performance of the proposed ant-colony algorithms with respect to the total-flowtime objective, the results of a recent study by Liu and Reeves (2001) are referred to. Liu and Reeves developed a couple of new heuristics, and compared their heuristics with a number of existing heuristics such as those by Ho (1995), Rajendran and Ziegler (1997b), Wang et al. (1997) and Woo and Yim (1998). They considered the benchmark problems of Taillard (1993), and reported the best-heuristic solutions for these benchmark problems with respect to the total-flowtime objective. It is to be noted that no single heuristic, considered by Liu and Reeves, has emerged to be the best for all benchmark problems. In order to compare the solutions yielded by the M-MMAS and PACO with the best-heuristic solutions reported by Liu and Reeves, we proceed as follows. It is to be noted that we need to use the fast heuristic by Rajendran (1993), followed by the three-time implementation of the proposed local search scheme, to obtain the initial sequence to the ant-colony algorithm with the objective of minimizing total flowtime of jobs.

Let the best-heuristic solution, among all heuristics under consideration by Liu and Reeves, and the solutions yielded by the M-MMAS and PACO for a given problem be denoted by F_1, F_2 and F_3 respectively. These solutions are relatively evaluated as follows:

Percentage relative increase in total flowtime of the solution yielded by approach i

$$= (F_i - \min\{F_k,\ k = 1, 2 \text{ and } 3\}) \times 100/(\min\{F_k,\ k = 1, 2 \text{ and } 3\}). \quad (9)$$

The mean values are noted over 10 different problems of a given size ($n \times m$) and are reported in Table 2. In addition, the maximum values of percentage relative increase in total flowtime with respect to different procedures are also reported in the table.

It is evident from the table that the M-MMAS and PACO yield solutions of superior quality, as against the best-heuristic solutions reported by Liu and Reeves, on an overall basis. We also observe that the PACO performs better than the M-MMAS in the case of relatively large-sized problems than in the case of relatively small-sized problems. Such a difference in the performance has already been observed with respect to the makespan-objective, and reasons have been discussed as well. We also observe that the performance of the ant-colony algorithms is better in the case of total-flowtime minimization than in the case of makespan minimization. The reason is that there exist different sequences with different total-flowtime values, even though their makespan values may be the same or almost the same. In fact, even for a two-machine flowshop problem, it has been found that the variations in total flowtime of different sequences are quite large even though their makespan values

Table 2
Relative performance of three heuristic procedures for the large-sized problems, measured in terms of mean and maximum percentage relative increase in total flowtime with respect to the best heuristic solution

n	m	Relative percentage increase in total flowtime					
		BES (LR)		M-MMAS		PACO	
		Mean	Maximum	Mean	Maximum	Mean	Maximum
20	5	1.183	2.036	0.021	0.121	0.278	0.848
	10	1.393	2.170	0.010	0.105	0.284	0.680
	20	1.155	1.863	0.051	0.352	0.120	0.619
50	5	0.691	1.108	0.274	0.621	0.090	0.400
	10	1.377	2.377	0.392	1.167	0.144	1.120
	20	1.611	3.280	0.304	0.648	0	0
100	5	0.179	0.499	0.134	0.435	0.252	0.809
	10	0.755	1.671	0.351	0.998	0.065	0.428
	20	1.548	3.015	0.297	1.345	0.066	0.476

Notes: 1. Sample size in every problem set is 10 and the total number of problems is 90, generated by using the procedure given by Taillard (1993).
2. A larger value of an entry indicates an inferior performance, relative to the best performing heuristic.
3. BES (LR) refers to the best performing heuristic, among many heuristics investigated by Liu and Reeves (2001), and M-MMAS and PACO refer to the proposed ant-colony algorithms.

are the same (see Rajendran, 1992, for details). Hence, it appears that the permutation flowshop scheduling problem with the makespan-objective is 'harder' to solve than the flowshop scheduling problem with the total-flowtime objective with the use of ant-colony algorithms.

We also report the absolute values of total flowtime of jobs for various benchmark problems that have been yielded by different approaches for the benefit of future researchers who may like to come up with new heuristics and compare their results with our findings (see Table 3 for details). In fact, a similar presentation of results by Liu and Reeves has eased our burden in the sense that we have not programmed again the heuristics considered by them in our study.

5. Summary

Of late, attempts are being made to solve combinatorial optimization problems by making use of ACO algorithms. Compared to other meta-heuristics 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, we have investigated the problem of scheduling in permutation flowshops by using ant-colony algorithms. We have proposed two ant-colony algorithms, with the first algorithm (M-MMAS) being an extended version of an existing ant-colony algorithm with the consideration of a new local search technique and an existing methodology for the use of sum of trail intensities. The second algorithm, called the PACO, has been newly developed in this study. First, the effectiveness of the proposed ant-colony algorithms is evaluated by considering the benchmark problems and upper bound values for makespan, given by Taillard. It has found that both the proposed ant-colony algorithms perform better than the existing ant-colony algorithm. Subsequently, the effectiveness of the proposed ant-colony algorithms has been evaluated by considering the objective of minimizing total flowtime and by comparing with the best heuristic solutions reported in a recent research study by Liu and Reeves. The performance evaluation shows that the proposedant-colony algorithms are clearly superior to the heuristics analyzed by Liu and Reeves. For 83 out of 90 problems considered, better solutions have been found by the two proposed

Table 3
Absolute total flowtime of jobs yielded by three heuristic procedures for the benchmark problems by Taillard (1993)

n	m	Total flowtime of jobs yielded by heuristics for the benchmark problems by Taillard		
		BES (LR)	M-MMAS	PACO
20	5	14 226	14 056	14 056
		15 446	15 151	15 214
		13 676	13 416	13 403
		15 750	15 486	15 505
		13 633	13 529	13 529
		13 265	13 139	13 123
		13 774	13 559	13 674
		13 968	13 968	14 042
		14 456	14 317	14 383
		13 036	12 968	13 021
20	10	21 207	20 980	20 958
		22 927	22 440	22 591
		20 072	19 833	19 968
		18 857	18 724	18 769
		18 939	18 644	18 749
		19 608	19 245	19 245
		18 723	18 376	18 377
		20 504	20 241	20 377
		20 561	20 330	20 330
		21 506	21 320	21 323
20	20	34 119	33 623	33 623
		31 918	31 604	31 597
		34 552	33 920	34 130
		32 159	31 698	31 753
		34 990	34 593	34 642
		32 734	32 637	32 594
		33 449	33 038	32 922
		32 611	32 444	32 533
		34 084	33 625	33 623
		32 537	32 317	32 317
50	5	65 663	65 768	65 546
		68 664	68 828	68 485
		64 378	64 166	64 149
		69 795	69 113	69 359
		70 841	70 331	70 154
		68 084	67 563	67 664
		67 186	67 014	66 600
		65 582	64 863	65 123
		63 968	63 735	63 483
		70 273	70 256	69 831
50	10	88 770	89 599	88 942
		85 600	83 612	84 549
		82 456	81 655	81 338
		89 356	87 924	88 014
		88 482	88 826	87 801
		89 602	88 394	88 269
		91 422	90 686	89 984
		89 549	88 595	88 281
		88 230	86 975	86 995
		90 787	89 470	89 238
50	20	129 095	127 348	126 962
		122 094	121 208	121 098
		121 379	118 051	117 524
		124 083	123 061	122 807
		122 158	119 920	119 221
		124 061	122 369	122 262
		126 363	125 609	125 351
		126 317	124 543	124 374
		125 318	124 059	123 646
		127 823	126 582	125 767
100	5	256 789	257 025	257 886
		245 609	246 612	246 326
		241 013	240 537	241 271
		231 365	230 480	230 376
		244 016	243 013	243 457
		235 793	236 225	236 409
		243 741	243 935	243 854
		235 171	234 813	234 579
		251 291	252 384	253 325
		247 491	246 261	246 750
100	10	306 375	305 004	305 376
		280 928	279 094	278 921
		296 927	297 177	294 239
		309 607	306 994	306 739
		291 731	290 493	289 676
		276 751	276 449	275 932
		288 199	286 545	284 846
		296 130	297 454	297 400
		312 175	309 664	307 043
		298 901	296 869	297 182
100	20	383 865	373 756	372 630
		383 976	383 614	381 124
		383 779	380 112	379 135
		384 854	380 201	380 765
		383 802	377 268	379 064
		387 962	381 510	380 464
		384 839	381 963	382 015
		397 264	393 617	393 075
		387 831	385 478	380 359
		394 861	387 948	388 060

Note: BES (LR) refers to the best performing heuristic, among many heuristics investigated by Liu and Reeves (2001), and M-MMAS and PACO refer to the proposed ant-colony algorithms.

ant-colony algorithms, as compared to the solutions reported by Liu and Reeves. It has also been

observed that the second proposed ant-colony algorithm (PACO) performs better than the first proposed ant-colony algorithm (M-MMAS) in the case of relatively large-sized permutation flowshop problems than in the case of relatively small-sized flowshop problems. This difference in the performance is attributed to the use of differential initial setting and differential updation of trail intensities in the PACO. It is hoped that more-efficient ant-colony algorithms will be developed for scheduling in various production systems such as flowshops, jobshops and cellular manufacturing systems with respect to different objectives.

Acknowledgements

The first author gratefully acknowledges the Research Fellowship of Alexander–von-Humboldt Foundation for carrying out this work in 2001. The revision of the work was undertaken when the first author was supported by the Foundation in 2002. The authors are thankful to the two referees for the suggestions and comments to improve the earlier version 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, Dipartimento di Elettronica, Politecnico di Milano, Italy.

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 flow-shop. 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 $P||\Sigma C_i$ 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. In: Lecture Notes in Computer Science, vol. 1803. Springer-Verlag, Berlin, pp. 287–296.

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 Operations Research Society of Japan 21, 287–299.

Nawaz, M., Enscore Jr., E.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., 1992. Two-machine flowshop scheduling problem with bicriteria. Journal of Operational Research Society 43, 871–884.

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., 1997a. Heuristics for scheduling in a flowshop with setup, processing and removal times separated. Production Planning and Control 8, 568–576.

Rajendran, C., Ziegler, H., 1997b. 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., 1999. Heuristics for scheduling in flowshops and flowline-based manufacturing cells to minimize the sum of weighted flowtime and weighted tardiness of jobs. Computers and Industrial Engineering 37, 671–690.

Stuetzle, T., 1998. An ant approach for the flow shop problem. In: Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT '98), vol. 3. Verlag Mainz, Aachen, Germany, pp. 1560–1564.

Stuetzle, T., Hoos, H.H., 2000. Max–min ant system. Future Generation Computer Systems 16, 889–914.

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

Wang, C., Chu, C., Proth, J.-M., 1997. Heuristic approaches for $n/m/F/\Sigma C_i$ scheduling problems. European Journal of Operational Research 96, 636–644.

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.