Available online at www.sciencedirect.com
SCIENCE DIRECT®
Int. J. Production Economics 101 (2006) 259–272
international journal of production economics
www.elsevier.com/locate/ijpe

An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops

Yuvraj Gajpal, Chandrasekharan Rajendran*

Department of Management Studies, Indian Institute of Technology Madras, Chennai 600 036, India

Received 11 July 2002; accepted 13 January 2005
Available online 5 March 2005

Abstract

The problem of scheduling in permutation flowshops with the objective of minimizing the completion-time variance of jobs is considered and solved by making use of ant-colony optimization (ACO) algorithms. ACO is an algorithmic approach, inspired by the foraging behavior of real ants, which can be applied to solve combinatorial optimization problems. A new ant-colony algorithm (NACO) has been developed in this paper to solve the flowshop scheduling problem. The objective is to minimize the completion-time variance of jobs. Two existing ant-colony algorithms and the proposed ant-colony algorithm have been compared with an existing heuristic for scheduling with the objective of minimizing the completion-time variance of jobs. It is found that the proposed ant-colony algorithm gives promising and better results, on an average, as compared to those solutions given by the existing ant-colony algorithms and the existing heuristic for the permutation flowshop scheduling problem under study.

Keywords: Scheduling; Flowshop; Completion-time variance; Ant-colony algorithm

1. Introduction

The problem of scheduling a set of jobs on one or several machines in a permutation flowshop to optimize a given objective function has been the subject of extensive research. Many exact and heuristic algorithms have been proposed over the years for solving the static permutation flowshop scheduling problem with the objectives of minimizing the 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; Woo and Yim, 1998; Liu and Reeves, 2001; Allahverdi and Aldowaisan,

*Corresponding author. Tel.: +91 44 22578444; fax: +91 44 22570509.
E-mail address: craj@iitm.ac.in (C. Rajendran).

0925-5273/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2005.01.003

2002; Chung et al., 2002). In addition, metaheuristic 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; Ben-Daya and Al-Fawzan, 1998; Chang et al., 2002).

Merten and Muller (1972) pointed out the appropriateness of considering non-regular measures of performance with respect to scheduling problems. They considered the problem of scheduling n jobs on a single machine so as to minimize the variance of job completion times, called the completion-time variance (CTV) problem. For the CTV problem with respect to a single-machine, Eilon and Chowdhury (1977) showed that the optimal sequence problem is V-shaped (i.e., the sequence in which all jobs prior to the shortest-time job are in descending order and all jobs after the shortest-time job are in ascending order of their process times). Kanet (1981), Bagchi et al. (1987), Vani and Raghavachari (1987), and Raghavachari (1988) also dealt with the problems of scheduling with non-regular penalty functions. Kubiak (1993) proved that the CTV problem is NP-hard and that no polynominal time algorithm exists to find an optimal solution for the CTV problem for a single-machine. Considering the problem of scheduling in permutation flowshops with the objective of minimizing the variance of completion times of jobs, Gowrishankar et al. (2001) developed a branch-and-bound algorithm for obtaining the best permutation sequence. In addition, they proposed a heuristic. The heuristic works as follows: (a) the given m-machine flowshop problem is reduced to a set of $(m-1)$ two-machine problems by using the heuristic principle of Campbell et al. (1970); (b) the jobs in every reduced two-machine problem are arranged in a V-shaped sequence; (c) the best sequence, among $(m-1)$ sequences, with respect to the minimization of CTV of jobs in the given m-machine flowshop, is chosen; and (d) the sequence is improved by an insertion scheme, followed by a repeated application of the adjacent pairwise interchange of jobs. It is being noted that the objective of minimizing the CTV serves to minimize the variations in resource consumption and utilization. Further, if all jobs have the same and common due-date, which is equal to the mean completion time of jobs, then the minimization of the CTV will lead to reduced mean absolute lateness and minimum sum of squared deviations of job-completion times from the common due-date.

In recent times, attempts are being made to solve the combinatorial optimization problems by making use of ant-colony-optimization algorithms (or simply, ant-colony or ACO algorithms). An introduction to the ACO algorithms has been dealt with by Dorigo et al. (1996). Basically, the ACO algorithm is a population-based, cooperative search procedure that is derived from the behavior of real ants. It makes use of simple agents called *ants* that iteratively construct solutions to combinatorial optimization problems. The generation (or construction) of solutions by ants is guided by (artificial) pheromone trails and problem-specific heuristic information. In the process, the ants 'deposit' pheromones. An individual ant constructs a complete solution by starting with the null solution and iteratively adding solution components until a complete solution is constructed. After the construction of the complete solution, every ant gives feedback on the solution by depositing pheromones (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 making use of an 'evaporation factor' (or the equivalent 'persistence of trail') in updating trail intensities. Attempts have been made recently to solve scheduling problems using ACO algorithms (e.g., Stuetzle (1998) for flowshop scheduling problems with the objective of minimizing the makespan; Merkle and Middendorf (2000) for single-machine scheduling problems with the objective of minimizing the total weighted tardiness of jobs; and Rajendran and Ziegler (2004) for flowshop scheduling problems with the objective of minimizing the makespan/total flowtime of jobs).

In this paper, we investigate the problem of scheduling in flowshops with the objective of minimizing completion-time variance, by using ant-colony algorithms. We propose a new ant-colony-optimization algorithm, and compare it with the existing ant-colony algorithms for scheduling by Stuetzle (1998) (called MMAS), and Rajendran and Ziegler (2004) (called PACO) (these algorithms being adapted in the current

study for scheduling to minimize the completion-time variance of jobs) and the existing heuristic reported by Gowrishankar et al. (2001). An extensive performance analysis of the heuristics under study is carried out by evaluating the completion-time variance of jobs in the benchmark flowshop problems that are generated by using the procedure suggested by Taillard (1993). It is to be noted that the flowshop problems generated by the procedure of Taillard are meant to serve as benchmark problems for flowshop scheduling with respect to the makespan objective. However, it is common to use the same set of flowshop problems as benchmark problems for other objectives as well (see Liu and Reeves, 2001; Rajendran and Ziegler, 2004).

2. Problem formulation of 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 further assumed that the sequence in which each machine processes all jobs is identical on all machines. 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 study. We also assume that in the process of minimizing the CTV, idle times are not inserted while developing schedules (or timetables) for jobs in a given permutation sequence. The reason is that the objectives of minimizing the sum of machine idle times (including inserted idle times) and minimizing the CTV are basically conflicting in nature. Hence, we restrict ourselves to the problem of minimizing the CTV with no inserted idle times. In fact, it is quite common to find in the literature on flowshop scheduling the assumption that idle times are not inserted while developing schedules. In other words, in this paper we consider the problem of minimizing the CTV of jobs, with the consideration of non-delay schedules.

We use the following terminology:

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

The completion time of σi on machine j can be found by the following recursive equation:

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

where $q(\Phi,j) = 0$ and $q(\sigma,0) = 0$, for all σ and j, with Φ denoting a null schedule.

The flowtime of job i, C_i, is given by

$$C_i = q(\sigma i, m). \tag{2}$$

The completion time variance of jobs, CTV, in a complete sequence is obtained as follows:

$$\text{CTV} = \sum_{i=1}^{n} (C_i - \bar{C})^2/n, \tag{3}$$

where

$$\overline{C} = \left(\sum_{i=1}^{n} C_i\right) \Big/ n. \tag{4}$$

3. Description and analysis of an existing ACO algorithm for flowshop scheduling

The first ACO algorithm for scheduling in flowshops to minimize makespan, developed by Stuetzle (1998), is now presented, followed by a discussion of the local search procedure used in the algorithm. The algorithm is called Max–Min Ant System (MMAS). The complete details of the algorithm are presented in view of the fact that ACO algorithms are relatively less known, as against simulated annealing or genetic algorithms, with respect to the application to scheduling problems.

The basic steps of the MMAS are presented below:

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 intensity, denoted τ_{ip}, where $\tau_{\max} \geqslant \tau_{ip} \geqslant \tau_{\min}$.

Step 3: Return the best solution found.

In the context of application of the ACO algorithm to scheduling problems, τ_{ip} denotes the trail intensity (or desire) of setting job i in position p of a sequence. These trails form a kind of adaptive memory of previously found solutions, and are modified at the end of every iteration. It is to be noted that, for every job i, for any possible position p, a pheromone value is stored and updated in each iteration of the ACO algorithm. Hence, there are n^2 such values of τ_{ip}. In order to ensure that the trail intensities do not go beyond certain limits during the process of updating the trail, Stuetzle introduced the maximum and minimum values for τ_{ip}, denoted by $\tau_{\max}$ and $\tau_{\min}$, respectively.

3.1. Initialization of parameters in the MM Ant System

Initially, all τ_{ip}'s are set equal to $\tau_{\max}$, where $\tau_{\max}$ is set equal to $1/((1-\rho) \times Z_{\text{best}})$, and $\tau_{\min}$ is set equal to $\tau_{\max}/5$. Note that ρ denotes the persistence of the trail, and is set to 0.75. It is also to be noted that $\tau_{\max} \geqslant \tau_{ip} \geqslant \tau_{\min}$ is always maintained in the search process of the MMAS. The term 'Z_{best}' refers to the best value of objective function that has been obtained so far. The heuristic by Nawaz et al. (1983) (called NEH heuristic) is used to obtain a solution to the given flowshop scheduling problem with the objective of the minimizing makespan (in this paper we use the heuristic with the consideration of the objective of minimizing the CTV of jobs). The solution is improved by the local search, called the modified first-move strategy, that has been proposed by Stuetzle. The objective-function value of the solution yielded by this local search procedure is initially set to Z_{best}, and hence all τ_{ip}'s are initialized. The mechanics of the modified first-move strategy is explained below with a simple numerical example for the ease of understanding. In this paper, we refer to this local search procedure as the position-based local search procedure.

Suppose we have {5-4-3-2-1} as the seed solution or sequence. Call it the current seed sequence.

Do the following local-search insertion procedure K times (until there is no improvement in the local search):

{

Step 1: Choose the job found in the first position of the current seed sequence, i.e., job 5. Insert it in all possible positions, thereby getting the sequence {4-5-3-2-1}, {4-3-5-2-1}, {4-3-2-5-1} and {4-3-2-1-5}. Choose the best sequence with respect to the objective under consideration; say, sequence {4-5-3-2-1} is chosen. If it is better than the current seed sequence {5-4-3-2-1}, then update the current seed sequence as {4-5-3-2-1}; else {5-4-3-2-1} is retained as the current seed sequence. Assume that {4-5-3-2-1} is chosen.

Step 2: Choose the job found in the second position of the current seed sequence, and proceed with its possible insertions. In this case job 5 is chosen again.

Step 3: Likewise, proceed until the job in the last position of the current seed sequence is checked for possible improvement.

}

The number of iterations of this local search procedure (indicated by K) will obviously vary, depending on the sequence that is improved upon. A possible deficiency of this local search procedure is that a job may be checked more than once for possible improvement, and hence there is a possibility of wasted opportunities or the lost check of different jobs being evaluated. In other words, we may expect the number of iterations (indicated by K) to go up to reach good local optima. This deficiency may have negative impact on the performance of the MMAS in case of execution-time restrictions on heuristics.

3.2. Generation of an ant-sequence

Starting from a null sequence, ant-colony algorithm makes use of trail intensities in order to determine the job to be appended in position p, where $1 \leqslant p \leqslant n$, and n refers to the number of jobs to be scheduled. Even though the trail intensity τ_{ip} 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 by the ant, called an ant-sequence. In the case of MMAS, the following procedure is used to choose an unscheduled job, say job i, probabilistically for position p.

Sample a uniform random number u in the range [0,1].
If $u \leqslant (n-4)/n$
then
  among the jobs that are not yet scheduled, choose the job with the maximum value of τ_{ip};
else
  job i is selected, from the set of first five unscheduled jobs as present in the best sequence obtained so far, for position p by sampling from the following probability distribution:

$$\varepsilon_{ip} = \left(\tau_{ip} \Big/ \sum_{i''} \tau_{i''p} \right), \tag{5}$$

  where job i'' 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.

3.3. Updating of trail intensities

An ant-sequence is subjected to the position-based local search procedure to enhance the quality of solution. Let the objective-function value of this improved sequence be denoted by Z_{current}. Subsequently, trail intensities are updated as follows:

$$\tau_{ip}^{\text{new}} = \begin{cases} \rho \times \tau_{ip}^{\text{old}} + (1/Z_{\text{current}}) & \text{if job } i \text{ is placed in position } p \text{ in the generated sequence;} \\ \rho \times \tau_{ip}^{\text{old}} & \text{otherwise,} \end{cases} \quad (6)$$

where ρ is set equal to 0.75.

Update the best sequence and Z_{best}, if the generated sequence is superior to the best sequence that has been obtained so far.

4. Proposed (or) new ACO algorithm (NACO algorithm)

The salient features of the NACO algorithm are presented, followed by the detailed discussion of the algorithm.

Step 1: Obtain an initial solution Σ^* by using NEH heuristic and with the consideration of the minimization of CTV of jobs (see Appendix A for details).

Step 2: Improve the initial solution by using the proposed random-job-insertion local search procedure, which is executed three times (i.e., by setting $K = 3$) (see Section 4.1.1 for details). The improved sequence is taken as the seed sequence for the NACO algorithm. Call this sequence Σ^S.

Step 3: Initialize the pheromone trails and parameters.

Step 4: Do the following 40 times (i.e., the generation of 40 ant-sequences):
- generate an ant-sequence by using the principles of the NACO algorithm;
- if the ant-sequence generated is the same as a previous ant-sequence, generate another ant-sequence until it is different from the previously generated ant-sequences;
- improve the solution by executing the random-job-insertion local search thrice (i.e., by setting $K = 3$);
- update the pheromone trail or trail intensity, denoted τ_{ip}.

Step 5: Improve the best solution found in Step 4, by using random-job-insertion local search procedure, which is executed three times (i.e., by setting $K = 3$).

Step 6: Return the solution found.

We now present the details of the NACO algorithm.

4.1. Description of local search procedure used in NACO algorithm

In the NACO, we propose and use *random-job-insertion local search* (RJILS) procedure to improve the initial sequence, every ant-sequence generated and the final solution by executing the local search procedure three times (i.e., $K = 3$).

4.1.1. Random-job-insertion local search

We propose a local search procedure on this paper, called the RJILS procedure. This local search procedure is best explained with a simple numerical example.

Suppose we have {5-4-3-2-1} as the seed sequence. Call it the current seed sequence.

Do the following local-search insertion procedure K times:

{
Step 0: Initialize a set of n distinct jobs, say, {1,2,3,4,5} denoted by Ω.
Step 1: Randomly choose a job, say, job 3, from Ω. Insert it in all possible positions of the current seed sequence, thereby getting sequences {3-5-4-2-1}, {5-3-4-2-1}, {5-4-2-3-1} and {5-4-2-1-3}. Choose the best sequence with respect to the objective under consideration; say, sequence {5-4-2-3-1} is chosen. If it is better than the current seed sequence {5-4-3-2-1}, then update the current seed sequence as {5-4-2-3-1}; else {5-4-3-2-1} is retained as the current seed sequence. Assume that sequence {5-4-2-3-1} is chosen. Remove the chosen job (i.e., job 3) from Ω.
Step 2: Randomly choose another job from Ω and proceed with its possible insertions in the current seed sequence. For example, if job 5 is chosen, we get sequences {4-5-2-3-1}, {4-2-5-3-1}, {4-2-3-5-1} and {4-2-3-1-5}. Compare the best among these sequences with the current seed sequence {5-4-2-3-1}, and update the current seed sequence, if necessary. Remove job 5 from Ω.
Step 3: Likewise, proceed until all n jobs are checked (i.e., all jobs are removed from Ω) for possible improvement of the current seed sequence.
}

It is to be noted that sequence Σ^*, obtained by the NEH heuristic, is taken as the seed sequence and subjected to the random-job insertion based local search by executing the search three times (i.e., $K = 3$). The final sequence thus obtained is taken as the seed sequence for the proposed ant-colony algorithm. Call this sequence Σ^S.

4.2. Initialization of parameters in the NACO algorithm

Initially, all τ_{ip}'s are set to $(1/\mathrm{CTV}_{\mathrm{best}})$, and ρ is set to 0.75. It is to be noted that the value of $\mathrm{CTV}_{\mathrm{best}}$ is initialized to the CTV of the seed solution to the NACO algorithm, i.e., CTV of sequence Σ^S that is obtained, as described in Section 4.1. The NACO algorithm does not use any upper and lower limits with respect to trail intensities of jobs (i.e., $\tau_{\max}$ and $\tau_{\min}$), as compared to the MMAS.

4.3. Generation of an ant-sequence

Starting from a null sequence, the NACO algorithm makes use of trail intensities in order to determine the job to be appended next in position p, where $1 \leqslant p \leqslant n$, and n refers to the number of jobs to be scheduled. In the case of the NACO algorithm, the following procedure is used to choose an unscheduled job, say job i, probabilistically for position p.

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
    set $T_{ip} = \sum_{q=1}^{p} \tau_{iq}$; among the set of first five unscheduled jobs as present in the best sequence obtained so far, choose the job with the maximum value of T_{ip};
  else

job i is selected, from the set of first five unscheduled jobs as present in the best sequence obtained so far, for position p by sampling from the following probability distribution:

$$\varepsilon_{ip} = \left(T_{ip} \Big/ \sum_{i''} T_{i''p} \right), \tag{7}$$

where job i'' 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 sequence is generated accordingly.

The rationale behind the summation of trail intensities (namely, obtaining T_{ip}) is that the choice of the job for appending is based on the pheromone values up to position p, instead of being based on the pheromone value with respect to position p. Moreover, the summation value of pheromones indicates a better estimate of the desirability of placing a job in position p, as against the use of the actual pheromone value with respect to position p. In other words, the sum of pheromone values of job i up to position p indicates the 'need' or 'desire' to schedule job i not later than position p. This approach of summing up of trail intensities has been proposed by Merkle and Middendorf (2000) in the case of single-machine total-weighted-tardiness problem.

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_{ip} 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.

4.4. Updating of trail intensities

Every ant-sequence is subjected to the random-job-insertion local search procedure (with $K = 3$) to enhance the quality of solution. Let the objective function (i.e., CTV) of this improved sequence be denoted by $\mathrm{CTV}_{\mathrm{current}}$. Update the best sequence and $\mathrm{CTV}_{\mathrm{best}}$, if $\mathrm{CTV}_{\mathrm{current}}$ is less than $\mathrm{CTV}_{\mathrm{best}}$. Subsequently, trail intensities are updated as follows:

$$\tau_{ip}^{\mathrm{new}} = \begin{cases} \rho \times \tau_{ip}^{\mathrm{old}} + (1/(diff \times \mathrm{CTV}_{\mathrm{current}})) & \text{if job } i \text{ is placed in position } p \text{ in the generated sequence;} \\ \rho \times \tau_{ip}^{\mathrm{old}} & \text{otherwise;} \end{cases} \tag{8}$$

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

The reasons for the above setting are that all jobs occupying their respective positions in the generated sequence should not have their trail intensities increased by the same value, and that the jobs occupying in the generated ant-sequence the positions 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 the positions farther from their respective positions in the best sequence. It is envisaged that such a differential updating of trail intensities of different jobs with respect to their corresponding positions in the best ant-sequence obtained so far will enhance the performance of ant-colony algorithms.

It is to be noted that the value of K (i.e., number of times any local search procedure is performed on the generated ant-sequences) is set to 3 in the NACO, PACO and MMAS, in our current study.

We now present a comparison of the NACO and the ant-colony algorithm of Rajendran and Ziegler (2004), called the PACO. One of the differences is in the use of the local search scheme. While the

ant-colony algorithm of Rajendran and Ziegler uses the job-index-based local search scheme (JILS), the NACO uses the RJILS. Basically, the JILS chooses the job on the basis of its index for insertion in different positions of the current seed sequence for possible improvement, while the RJILS chooses the job at random for insertion. Such a random choice is intended to avoid any possible bias in the choice of jobs for insertion, thereby enhancing the chances of getting out of local minima. In addition, it is possible that while using the JILS, the solution may not improve after the first or the second application of the JILS, and hence the consideration of the same sequence of jobs for insertion will not result in any improvement in the subsequent application of the JILS. This deficiency is not likely to occur in the application of the RJILS. Apart from the local search scheme, in terms of basic mechanics of the ant-colony algorithm, the NACO sets initially all τ_{ip}'s to the same value, namely, $(1/\mathrm{CTV}_{\mathrm{best}})$, as opposed to differential setting of τ_{ip}'s in the case of the PACO. In addition, after obtaining an ant-sequence and its improvement through the local search scheme, the NACO updates the value of τ_{ip} only with respect to the position of job i in the improved sequence, while the PACO updates the value of τ_{ip} with respect to more than one position of job i.

5. Performance analysis of the algorithms

The following heuristic procedures are evaluated with respect to the minimization of CTV of jobs in flowshops:

(1) the NACO algorithm;
(2) the MMAS adapted in the current study to minimize the CTV of jobs;
(3) the PACO adapted in the current study to minimize the CTV of jobs; and
(4) the existing heuristic by Gowrishankar et al. (2001): this heuristic is termed as EH in this current study.

In addition, we consider the following variants of the above heuristics.

(1) In view of the fact that we propose a new ant-colony algorithm, coupled with a new local search (namely, RJILS), and that the existing ant-colony algorithm of Stuetzle makes use of a different local search scheme (namely, position-based local search), we have evaluated the proposed ant-colony algorithm (with its basic mechanics) with the incorporation of position-based local search. We call this variant $\mathrm{NACO}^{\mathrm{p}}$. Basically, we make use of this variant to examine the relative effectiveness of the mechanics of the proposed ant-colony algorithm and the mechanics of the MMAS, with the local search scheme remaining the same. In addition, we have evaluated the proposed ant-colony algorithm (with its basic mechanics) with the incorporation of JILS used by Rajendran and Ziegler in the PACO (also see Rajendran and Ziegler (2005) for a related work). We call this variant $\mathrm{NACO}^{\mathrm{j}}$. Basically, we make use of this variant to examine the relative effectiveness of the mechanics of the proposed ant-colony algorithm and the mechanics of the PACO, with the local search scheme remaining the same. Such variants will help us to bring out the superior performance that can solely be attributed to the mechanics of the proposed ant-colony algorithm.

(2) Further, we have also considered the possible improvement in the existing heuristic, namely EH by Gowrishankar et al. (2001) by subjecting the heuristic sequence yielded by the EH to the RJILS (without any restriction on the value of K). In other words, we apply the proposed local search until there is no improvement in the quality of solution. We call this variant EH^{r}. Such a variant will help us to improve the performance of the EH to the extent possible, through the use of the proposed RJILS.

The benchmark flowshop scheduling problems of Taillard (1993) are considered and solved with respect to the minimization of the CTV, by making use of the above-mentioned five heuristic procedures.

We have calculated the relative percentage increase (RPI) in the CTV of jobs yielded by heuristic approach i as follows:

$$\mathrm{RPI}_i = (\mathrm{CTV}_i - \min\{\mathrm{CTV}_k; k = 1,2,3,4,5,6 \text{ and } 7\}) \times 100 / \min\{\mathrm{CTV}_k; k = 1,2,3,4,5,6 \text{ and } 7\}. \quad (9)$$

In the above, $k = 1$ refers to the EH, $k = 2$ refers to the EH^{r}, $k = 3$ refers to the MMAS, $k = 4$ refers to the PACO, $k = 5$ refers to the $\mathrm{NACO}^{\mathrm{p}}$, $k = 6$ refers to the $\mathrm{NACO}^{\mathrm{j}}$, and $k = 7$ refers to the NACO.

The mean RPI with respect to every heuristic approach for a given problem of size $(n \times m)$, over 10 problems, is calculated and reported in Table 1. In addition, we have also reported the maximum relative percentage increase in the CTV of jobs (corresponding to every heuristic) among 10 problems of size $(n \times m)$. The CPU time requirements for the main heuristic procedures are given in Table 2. Even though the ant-colony algorithms require more computational effort, their performance in terms of minimizing the CTV of jobs is very good and emerges far superior to that of the existing heuristic. We also provide the complete details of the CTV values yielded by the best heuristic in Table 3. Such details can serve as benchmark values for future researchers to compare their heuristic approaches and evaluate the effectiveness. Moreover, the researchers are saved from the burden of coding our heuristic approach!

An inspection of Table 1 reveals, on an average, especially in the case of large-sized problems, that the NACO emerges to be the best, and that, on the whole, the mechanics of the proposed ant-colony algorithm appears to be good (see the performance of the NACO, $\mathrm{NACO}^{\mathrm{j}}$, $\mathrm{NACO}^{\mathrm{p}}$ and PACO in the table). The superiority of the proposed local search can be noted by comparing the columns in Table 1 corresponding to the $\mathrm{NACO}^{\mathrm{p}}$, $\mathrm{NACO}^{\mathrm{j}}$ and NACO. However, on an average in the case of small-sized problems (with jobs up to 20), the MMAS performs the best, as also seen in the work by Rajendran and Ziegler (2004). This is so because the MMAS makes use of the term '$((n-4)/n)$' in the construction of an ant-sequence which works well for small values of n. The results of the evaluation of different algorithms demonstrate the superiority of the mechanics of the new ant-colony algorithm and the RJILS.

Table 1
Relative percentage increase in CTV of jobs yielded by heuristics for the benchmark problems of Taillard (1993)

n	m	EH		EH^{r}		MMAS		PACO		$\mathrm{NACO}^{\mathrm{p}}$		$\mathrm{NACO}^{\mathrm{j}}$		NACO	
		Mean	Max.	Mean	Max.	Mean	Max.	Mean	Max.	Mean	Max.	Mean	Max.	Mean	Max.
20	5	4.33	9.55	4.05	9.55	0.21	1.97	1.36	3.45	1.35	2.57	1.32	4.08	0.72	1.77
20	10	9.11	19.83	9.11	19.83	0.79	6.43	2.04	5.41	2.31	4.17	1.51	3.50	1.02	3.32
20	20	8.56	23.78	8.01	23.78	1.62	5.58	2.42	5.88	1.98	4.89	1.73	6.29	2.07	5.72
50	5	5.68	9.07	5.39	8.41	1.34	3.45	0.48	2.88	1.20	3.14	1.36	4.01	1.18	3.75
50	10	10.43	13.99	8.73	13.64	1.90	7.63	1.26	3.34	1.00	2.95	1.07	4.34	0.87	2.81
50	20	11.69	18.41	11.08	18.41	2.20	4.81	1.76	5.22	1.62	4.58	1.39	5.04	0.84	3.49
100	5	6.69	9.64	6.31	9.08	1.09	2.17	1.14	2.47	1.28	2.95	0.47	1.87	0.29	0.94
100	10	9.42	13.38	8.50	11.80	1.21	5.11	1.22	2.69	1.10	3.29	1.14	2.34	0.31	1.77
100	20	10.18	14.77	9.13	13.69	2.52	4.45	0.71	3.60	1.54	4.20	0.55	1.52	0.94	3.90

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) EH refers to the existing heuristic for solving the CTV-problem; EH^{r} refers to the existing heuristic, supplemented by the random-job insertion local search; MMAS refers to the ant-colony algorithm by Stuetzle (1998); PACO refers to the ant-colony algorithm by Rajendran and Ziegler (2004); $\mathrm{NACO}^{\mathrm{p}}$ refers to the proposed ant-colony algorithm with the use of position-based local search; $\mathrm{NACO}^{\mathrm{j}}$ denotes the proposed ant-colony algorithm with the use of job-index-based local search; and NACO refers to the proposed ant-colony algorithm with the use of random-job insertion local search.

Table 2
Mean CPU time (in s) required for the 3 procedures for solving the large-sized problems on a PC with Pentium-2 processor

n	m	Mean CPU time (in s)		
		EH	MMAS/PACO	NACO
20	5	<1	2	2
	10	<1	5	5
	20	<1	9	9
50	5	1	48	48
	10	1	81	81
	20	4	146	148
100	5	15	381	384
	10	18	652	652
	20	31	1199	1213

Table 3
CTV of jobs yielded by heuristics for the benchmark problems of Taillard (1993)

n	m	1	2	3	4	5	6	7	8	9	10
20	5	73040.55[5]	90885.27[3]	53894.49[3]	89822.05[6]	72350.55[3]	71665.73[3]	69088.45[3]	70214.31[3]	73329.22[3]	52580.03[4]
	10	89959.73[6,7]	93316.45[7]	67862.71[3,4,7]	63789.09[3]	74915.04[3]	59322.02[7]	89726.74[3]	73162.19[3]	91665.55[3]	79721.11[3]
	20	106760.70[3]	79883.23[5]	94117.25[3]	91197.39[3,6]	97539.41[6]	97419.73[7]	105212.60[7]	92997.55[3]	92481.50[4]	98381.60[6]
50	5	337797.00[4]	424159.00[6]	340703.00[4]	382174.00[7]	399393.90[4]	410108.60[4]	384700.20[4]	365121.50[4]	307528.40[4]	393417.60[5]
	10	412169.00[7]	376213.80[6]	375092.40[6]	427881.80[4]	418558.80[5]	400570.40[7]	424867.00[6]	411500.20[4]	376418.40[3]	422959.40[7]
	20	555325.40[7]	490080.60[7]	476917.90[7]	503854.00[7]	489436.30[7]	440341.20[6]	508770.70[7]	503631.80[5]	481857.40[5]	512969.30[4]
100	5	1587393.00[6]	1499100.00[6]	1415901.00[7]	1283034.00[3]	1470733.00[7]	1333450.00[7]	1433848.00[3]	1374820.00[5]	1595026.00[6]	1461685.00[4]
	10	1789760.00[6]	1480867.00[6]	1663430.00[5]	1812091.00[3]	1550735.00[7]	1413509.00[7]	1465169.00[7]	1686253.00[7]	1794633.00[7]	1861396.00[4]
	20	1712887.00[4]	1851755.00[6]	1815847.00[7]	1765004.00[7]	1802465.00[5]	1959520.00[4]	1840811.00[4]	1908259.00[4]	1877671.00[4]	1821639.00[6]

Note: Superscript value shows the heuristic approach that has yielded the corresponding CTV value, where 1 refers to the EH (existing heuristic for solving the CTV-problem), 2 refers to the EH^r (existing heuristic, supplemented by the random-job insertion local search), 3 refers to the MMAS (ant-colony algorithm by Stuetzle (1998)), 4 refers to the PACO by Rajendran and Ziegler (2004), 5 refers to the $NACO^p$ (proposed ant-colony algorithm with the use of position-based local search), 6 refers to the $NACO^j$ (proposed ant-colony algorithm with the use of job-index based local search), and 7 refers to the NACO (proposed ant-colony algorithm with the use of random-job-insertion local search).

In order to provide an idea of the convergence of the NACO to the best solution, the best value of the CTV up to a given number of ant-sequences is observed for every problem. Basically, we observe the process of convergence of the NACO. We provide a sample figure (see Fig. 1) corresponding to a problem with the problem size (50×20).

6. Summary

While many solution procedures have been developed over many years for solving the flowshop scheduling problems, only few attempts have been made for developing solution procedures for the flowshop problem with the objective of minimizing completion-time variance. This study has addressed the problem of scheduling in the flowshop with the objective of minimizing completion-time variance by

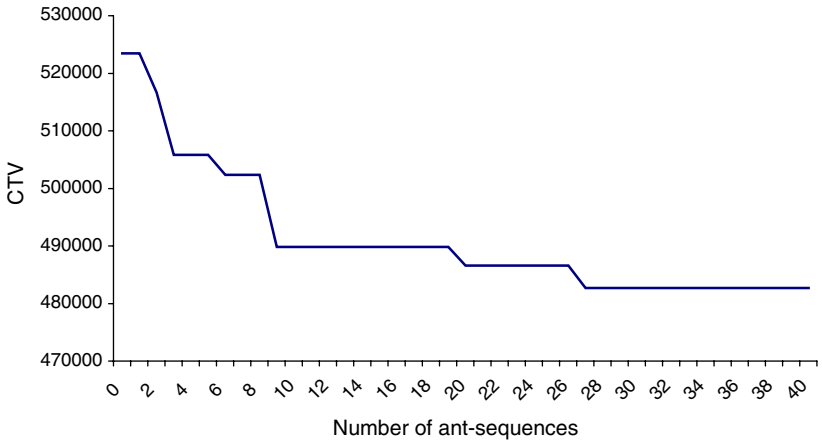

Fig. 1. Convergence process of the proposed ant-colony-algorithm.
Note: (1) The best value of the CTV up to a given number of generated ant-sequences is shown in the above figure. (2) The above figure shows the convergence process for a problem with the problem size (50×20).

making use of the ant-colony-optimization algorithm that involves a population-based, cooperative search procedure process derived from the behavior of real ants. 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. A new ant-colony algorithm (NACO) has been developed in this paper to solve the flowshop scheduling problem with the objective of minimizing the completion-time variance of jobs. The existing ant-colony and the proposed ant-colony algorithms have been compared with the existing heuristic for the flowshop scheduling problem with the objective of minimizing the completion-time variance. An extensive performance analysis of these heuristics has been carried out by evaluating the completion-time variance of jobs yielded by the procedures under study. It is found that the proposed ant-colony algorithm is giving promising and better results, on an average, as compared to those solutions given by the existing ant-colony algorithms and the existing heuristic for the permutation flowshop scheduling problem under study.

Acknowledgement

The authors are thankful to the two reviewers for their suggestions and comments to improve the three earlier versions of the paper. The second author gratefully acknowledges the discussion with Professor Hans Ziegler, University of Passau, Germany.

Appendix A

NEH heuristic:

The NEH heuristic, as adapted in the current study with the CTV-objective, works as follows.
Phase 1:
Arrange the job in descending order of $\sum_{j=1}^{m} t_{ij.}$,
where t_{ij} denotes the process time of job i on machine j.

Phase 2:
Improve the sequence obtained from Phase 1 by the insertion scheme explained below.
Suppose we have {5-4-3-2-1} as the seed sequence obtained from Phase 1. Call this sequence the overall seed sequence.

Do the following local-search procedure once:

{

Step 1: Choose the job found in the first and second positions, i.e., jobs 5 and 4. Form the two possible sequences, i.e., {5-4} and {4-5}. Choose the better partial sequence with respect to the objective under consideration, i.e. CTV. Assume that {4-5} is chosen as the current partial sequence.

Step 2: Consider the job found in the third position of the overall seed sequence, i.e., job 3. Insert the job in all positions of the current partial sequence, and hence obtain {3-4-5}, {4-3-5} and {4-5-3}. Choose the best partial sequence, say {4-3-5}, and update the current partial sequence as {4-3-5}.

Step 3: Likewise, proceed until the job found in the last position of the overall seed sequence is inserted in all possible positions of the current partial sequence, and the best complete sequence is obtained.

Step 4: Compare the sequence obtained from Step 3 and the overall seed sequence (i.e., {5-4-3-2-1} in this case) with respect to the CTV. Return the better solution.

}

References

Allahverdi, A., Aldowaisan, T., 2002. New heuristics to minimize total completion time in *m*-machine flowshops. International Journal of Production Economics 77, 71–83.

Bagchi, U., Sullivan, R.S., Chang, Y., 1987. Minimizing mean squared deviation of completion times about a common due date. Management Science 33, 894–906.

Ben-Daya, M., Al-Fawzan, M., 1998. A tabu search approach for the flowshop 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 sequence problem. Management Science 16, B630–B637.

Chang, P.-C., Hsieh, J.-C., Lin, S.-G., 2002. The development of gradual-priority weighting approach for the multi-objective flowshop scheduling problem. International Journal of Production Economics 79, 171–183.

Chung, C.-S., Flynn, J., Kirca, O., 2002. A branch and bound algorithm to minimize the total flow time for *m*-machine permutation flowshop problems. International Journal of Production Economics 79, 185–196.

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.

Eilon, S., Chowdhury, I.G., 1977. Minimizing waiting time variance in the single machine problem. Management Science 23, 567–575.

Gelders, L.F., Sambandam, N., 1978. Four simple heuristics for scheduling a flow-shop. International Journal of Production Research 16, 221–231.

Gowrishankar, K., Rajendran, C., Srinivasan, G., 2001. Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date. European Journal of Operational Research 132, 643–665.

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 flowshop 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.

Kanet, J.J., 1981. Minimising variance of flow time in single machine system. Management Science 27, 1453–1458.

Kubiak, W., 1993. Completion time variance minimization on a single machine is difficult. Operations Research Letters 14, 49–59.

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, Lecture Notes in Computer Science, Springer, vol. 1803, Springer, Berlin, pp. 287–296.

Merten, A.G., Muller, A.E., 1972. Variance minimization single machine sequencing problems. Management Science 18, 518–528.

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 Jr., E.E., Ham, I., 1983. A heuristic algorithm for the *m*-machine, *n*-job flowshop sequencing problem. OMEGA 11, 91–95.

Raghavachari, M., 1988. Scheduling problems with non-regular penalty functions—a review. Operations Research 25, 144–164.

Rajendran, C., 1993. Heuristics for scheduling in a flowshop to minimize total flowtime. International Journal of Production Economics 29, 65–73.

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

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.

Rajendran, C., Ziegler, H., 2005. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. Computers & Industrial Engineering Forthcoming.

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

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

Vani, V., Raghavachari, M., 1987. Deterministic and random single machine sequencing with variance minimization. Operations Research 35, 111–120.

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.