Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times

C Gagné[1], WL Price[2]* and M Gravel[1]
[1]*Université du Québec, Chicoutimi, Québec, Canada; and* [2]*Université Laval, Québec, Canada*

We compare several heuristics for solving a single machine scheduling problem. In the operating situation modelled, setup times are sequence-dependent and the objective is to minimize total tardiness. We describe an Ant Colony Optimization (ACO) algorithm having a new feature using look-ahead information in the transition rule. This feature shows an improvement in performance. A comparison with a genetic algorithm, a simulated annealing approach, a local search method and a branch-and-bound algorithm indicates that the ACO that we describe is competitive and has a certain advantage for larger problems.
Journal of the Operational Research Society (2002) **53**, 895–906. doi:10.1057/palgrave.jors.2601390

Keywords: scheduling; metaheuristic; ant colony optimization; single machine; total tardiness; sequence-dependent setups

Introduction

Industrial production scheduling constitutes a fertile field for both researchers and practitioners of operational research. The interest in this field is generated not only by the problem-solving challenge that it offers but also by the practical results that can be achieved. However, researchers such as McCarthy and Liu[1] and McKay and Wiers[2] have remarked on the sometimes wide gap between the theoretical problems treated and those met in practice.

The development of efficient solution procedures for the scheduling of jobs in a casting centre belonging to a Canadian multinational firm is the principal aim of the research reported in this paper. The authors have previously reported[3,4] details of successful work in this area of metaheuristics. The aim of this paper is to describe the performance of some extensions to the ant colony optimization (ACO) algorithm[5,6] which has already demonstrated its usefulness in this industrial situation and is presently incorporated in software used by the firm. This paper presents a comparison with a genetic algorithm, a simulated annealing approach, a local search method and a branch-and-bound algorithm which indicates that the ACO that we describe is competitive and has a certain advantage for larger problems.

In the casting centre, holding furnaces contain molten metal ready to be poured. They may require certain draining and cleaning operations of varying durations between the casting of two successive jobs for different metal alloys. These operations may be seen as the setup operations dealt with in the literature. We seek a schedule for current released jobs that takes into account these sequence-dependent setup times as well as multiple objectives. We validate the performance of the new elements that we have introduced by solving a known problem from the literature, the single machine problem with sequence-dependent setups, and we use published benchmark problems[7] in our comparisons. This allows us to compare our results with those previously published for various metaheuristics.

*Correspondence: WL Price, Faculté des Sciences de l'administration, Université Laval, Québec, G1K 7P4 Canada.
E-mail: wilson.price@fsa.ulaval.ca

The single machine scheduling problem

It is possible to achieve interesting practical results through the study of single machine shops. For example, there may be a bottleneck machine that strongly influences performance and which allows a more complex shop to be adequately modelled as a single machine.

In the scheduling of a single machine shop, the minimization of total tardiness, an objective that seeks to improve customer service, has proved useful. Du and Leung[8] have shown that this problem is NP-hard. Wisner and Siferd[9] find that meeting target delivery dates is the most important scheduling objective and indicate that 58% of production planners actively seek to meet delivery dates.

Solution methods for the single machine scheduling problem with constant setup times are abundant in the literature. Koulamas[10] reviews the total tardiness minimization problem and points out that many solution approaches are available for the single machine problem. He proposes a classification of existing methods as *optimal* or *heuristic*.

The first class includes dynamic programming, branch and bound and hybrids including both. The class of heuristic methods can be further broken down to sub-classes including constructive algorithms, local search methods and decomposition methods. Results indicate that local search and decomposition approaches are generally more effective than construction heuristics.

Two major theoretical developments concern the single machine scheduling tardiness minimization problem. Emmons[11] developed the dominance condition and solved two particular cases with constant setups. Lawler,[12] among others, wrote on subject of the decomposition principle. These contributions allowed the development of optimal solution procedures but they also inspired the construction of various heuristics.

In their 1999 paper, Bauer *et al*[13] present an ACO procedure for solving the single machine total tardiness minimization problem and report good results. In 2000, den Besten *et al*[14] solved the single machine total weighted tardiness minimization problem using a heterogeneous colony of ants and they too indicate the success of their approach. Merkle and Middendorf[15] describe an ACO algorithm having an innovative pheromone evaluation rule for solving this same problem.

Scheduling with sequence-dependent setup times

Adding the characteristic of sequence-dependent setup times increases the difficulty of the problem of minimizing total tardiness on a single machine. This characteristic invalidates the dominance condition as well as the decomposition principle.[16]

The importance of explicitly treating sequence-dependent setups in production scheduling has been presented a number of times in the literature. In particular Wilbrecht and Prescott[17] state that this is particularly true where production equipment is being used close to its capacity levels. Wortman[18] states that the efficient management of production capacity requires the consideration of setup times.

Many practical industrial situations require the explicit consideration of setups and the development of appropriate scheduling tools. For example, Pinedo[19] describes a manufacturing plant making paper bags where setups are required when the type of bag changes. The duration of a setup depends on the similarity of the bags made in the preceding lot. A similar situation was observed in the plastics industry by Das *et al*[20] and Franca *et al.*[21] The printing industry also has setups that are sequence dependent because various cleaning operations are required when the print colours are changed.[22] The aluminium industry has casting operations where setups, mainly affecting the holding furnaces, are required between the casting of different alloys.[3] The textile, pharmaceutical, chemical and metallurgical industries present other practical examples where sequence-dependent setups are frequently observed.

In 1999, two reviews of problems with sequence-dependent setups were published. The authors proposed different classifications of the field but arrive at similar conclusions. Allahverdi *et al*[23] identify gaps in existing research, including in the underlying theoretical underpinnings and in the treatment of multiple objectives. They state that few papers have treated the many industrial applications having sequence-dependent setups in combination with the objective of meeting delivery dates. Their general conclusion is that scheduling where sequence-dependent setups are required is a fertile area for further research. Yang and Liao[24] observe that there are few comparisons of the solution methods developed for this problem. They make the same observation concerning the applications of the various methods available in practical situations.

The literature is not extensive for the single machine scheduling problem with sequence-dependent setup times where the objective is to meet delivery dates or to reduce tardiness, however Lee *et al*[25] have proposed the Apparent Tardiness Cost with Setups (ATCS) dispatching rule for minimizing weighted tardiness. Among other authors who have treated this problem, we find Ragatz[26] who proposed a branch-and-bound algorithm for the exact solution of smaller instances. A genetic algorithm and a local improvement method were proposed by Rubin and Ragatz[16] while Tan and Narasimhan[27] used simulated annealing. Finally, Tan *et al*[7] present a comparison of these four approaches and conclude, following a statistical analysis, that the local improvement method offers better performance than simulated annealing, which in turn is better than branch-and-bound. In this comparison, the genetic algorithm had the worst performance.

Let us now consider the formal definition of the problem of scheduling a single machine having sequence-dependent setup times where the objective is the minimization of total tardiness.[16] Let there be n jobs to produce, all released at time zero, and which must be completed without interruption on a single machine. Each job j has as attributes its production duration p_j, its delivery date d_j, and its setup time s_{ij}, which is incurred when job j is undertaken following job i in a job sequence Q. We define $Q = \{Q(0), Q(1), \ldots, Q(n)\}$ as the job sequence where $Q(j)$ is the subscript of the jth job in the sequence and where $Q(0) = 0$. The machine is continuously available through the planning period and can process only one job at a time. Once a job is started it must be completed without interruption. The end time of job j is expressed as:

$$c_{Q(j)} = \sum_{x=1}^{j} [s_{Q(x-1)Q(x)} + p_{Q(x)}]$$

The tardiness for this same job j is expressed as:

$$t_{Q(j)} = \max\{0, c_{Q(j)} - d_{Q(j)}\}$$

The objective to minimize is the total tardiness T for the set of jobs to be produced and is expressed as:

$$T_Q = \sum_{j=1}^{n} t_{Q(j)}$$

In this paper we present an ACO algorithm which has certain advantages for this case. In particular, it allows us to use various elements of information about the problem to better direct the search for good solutions. Moreover, the basic algorithm has already demonstrated its capacity to perform well in comparison to other metaheuristics in an industrial setting.[4] Our objective was to improve on this performance on the one hand by including extensions suggested in the literature, and on the other through the use of new elements that use broader information in the transition rule. The validation of the effectiveness of this metaheuristic in solving the single machine scheduling problem described will be presented in terms of the solution quality and the computation times. We compare our results to those reported in the recent study of Tan *et al.*[7]

An ACO metaheuristic for the single machine scheduling problem with sequence-dependent setup times

The basic ant colony algorithm

The pioneering work of Colorni *et al*[5] and Dorigo[6] was inspired by studies of the behaviour of ants.[28] Ants communicate through *pheromones* deposited on the ground in varying intensities as they move about. As more ants use the same path, more pheromone is deposited. Ants tend to follow these pheromone trails and in this manner, they communicate with each other as to the location of food sources. When an obstacle blocks an existing path, some ants will avoid it on the right side, others on the left side. Those having found the shorter path will rejoin the previous pheromone trail more quickly. This results in a more rapid buildup of pheromone on the shorter path, and still more ants will be attracted to it. In this way the favoured path from a nest to a food source tends to that having the shortest distance. The solution of the travelling salesman problem (TSP) was one of the first applications of the ACO.

Various extensions to the basic TSP algorithm were proposed, notably by Dorigo and Gambardella.[29] The improvements include a modified transition rule called the *pseudo-random-proportional rule*, global and local trail updating rules, the use of local improvement rules based on the *restricted 3-opt method*[30] and the use of a restricted candidate list as is common practice in large scale problems.[31] These extensions have been included in the algorithm proposed in this article. In other papers, previous authors such as Colorni *et al*,[5] Dorigo *et al*,[32] and Dorigo and Di Caro[33] provide details concerning the operation of the algorithm and choices of the parameter values. The algorithm described in this section is referred to in the literature as the *ACS-TSP*.[34]

An augmented ACO for the single machine scheduling problem with sequence-dependent setup times

The TSP may be used as a starting point in modelling the single machine scheduling problem where the objective is to minimize the total time taken to process all jobs (the makespan). However, where total tardiness minimization is the objective and where setup times are sequence-dependent, the problem is more difficult. Processing and setup times may still be used in a manner analogous to inter-city distances in the TSP, but they can only serve as an indirect guide in total tardiness minimization because they are frequently quite unrelated to due dates. In addition, the existence of sequence-dependent setup times reduces the efficiency of local search methods because it requires a complete re-evaluation of the objective function rather than an adjustment for the local effects.

Figure 1 outlines our procedure and we will now address the details of each step.

Step 1: *Pheromone initialization*

At each period, given that the previous job added to the sequence was job i, the ant chooses the next job to append by considering, among other factors, the pheromone trail intensity $\tau_{ij}(t)$. At the initial iteration, the trail intensity $\tau_{ij}(0)$ for all job pairs (i,j) is initialized to a small positive quantity τ_0. Subsequently, the pheromone trail will contain information based on the solution quality and the number of times ants chose to make jobs (i,j) adjacent.

Step 2: *Main loop*

In the main loop, each of the m ants constructs a sequence of n jobs. This loop is executed for $t_{\max}$ cycles.

Step 2.1: *Set initial job*

For each job sequence constructed, in order to maintain continuity, the last job produced during the previous period is taken to be the initial job. This job determines the nature of the setup required before the first job of the current period may be produced. This setup is taken into account when the fitness, L_k, of a job sequence k is determined. The fitness is the total tardiness of the job sequence with respect to the due dates of each of the jobs.

Step 2.2: *Each ant builds a job sequence*

Pseudo-random-proportional transition rule. From an existing partial job sequence, an ant must choose the next job to append. As trail intensity $\tau_{ij}(t)$ increases, the probability that adjacent job pair (i,j) will again be chosen by another ant also increases, but a compromise is made between the trail intensity and other elements. This choice is made using a pseudo-random-proportional

```
/* STEP 1 : PHEROMONE INITIALIZATION */
For each pair of jobs (i,j) do
        Set an initial value $\tau_{ij}(0) = \tau_0$
End For
Let $T_+$ be the best job sequence found so far and let $L_+$ be its total tardiness

/*STEP 2 : MAIN LOOP */
For $t = 1$ to $t_{max}$ do
  /* STEP 2.1 : SET INITIAL JOB */
  For $k = 1$ to $m$ do
        Set the initial job for ant $k$
        Store this information in $Tabu_k$
  End For
  /* STEP 2.2 : BUILD A JOB SEQUENCE FOR EACH ANT */
  Let $T_*$ be the best job sequence found in the current cycle and let $L_*$ be its total tardiness
  For $i = 1$ to $n$ do
        For $k = 1$ to $m$ do
          Choose the next job $j, j \notin Tabu_k$, among the $cl$ candidate jobs according to:
              $$j = \begin{cases} \arg\max_{l \notin Tabu_k} \left\{ [\tau_{il}(t)]^{\alpha} \cdot \left[\frac{1}{\bar{s}_{il}}\right]^{\beta} \cdot \left[\frac{1}{\bar{m}_{il}}\right]^{\delta} \cdot \left[\frac{1}{\bar{B}_{il}}\right]^{\phi} \right\} & \text{if } q \le q_0 \\ J & \text{if } q > q_0 \end{cases} \quad (1)$$
              where $J$ is chosen according to the probability :
              $$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot \left[\frac{1}{\bar{s}_{ij}}\right]^{\beta} \cdot \left[\frac{1}{\bar{m}_{ij}}\right]^{\delta} \cdot \left[\frac{1}{\bar{B}_{ij}}\right]^{\phi}}{\sum_{l \notin Tabu_k} [\tau_{il}(t)]^{\alpha} \cdot \left[\frac{1}{\bar{s}_{il}}\right]^{\beta} \cdot \left[\frac{1}{\bar{m}_{il}}\right]^{\delta} \cdot \left[\frac{1}{\bar{B}_{il}}\right]^{\phi}} \quad (2)$$
              Store this information in $Tabu_k$
              Local update of pheromone trail for chosen job pair $(i,j)$:
              $\tau_{ij}(t) = \rho_l \cdot \tau_{ij}(t) + (1-\rho_l) \cdot \Delta\tau_{ij}$ where $\Delta\tau_{ij} = \tau_0$     (3)
        End For
  End For
  /* STEP 2.3 : EVALUATION OF SOLUTIONS AND IMPROVEMENT METHOD */
  For $k = 1$ to $m$ do
        Compute the total tardiness $L_k(t)$ for the job sequence $T_k(t)$ produced by ant $k$
        Apply local improvement method for the sequence $T_k(t)$ and recompute $L_k$
        If an improved sequence is found then update $T_+$ and $L_+$
  End For
  /* STEP 2.4 : GLOBAL UPDATE OF PHEROMONE TRAIL */
  For each adjacent job pair $(i,j) \in T_*$ do
        Update the pheromone trail according to:
        $\tau_{ij}(t+1) = \rho_g \cdot \tau_{ij}(t) + (1-\rho_g) \cdot \Delta\tau_{ij}(t)$ where $\Delta\tau_{ij}(t) = 1/L_*$     (4)
  End For
  Empty all $Tabu_k$
End For
Print the best job sequence $T_+$ and its total tardiness $L_+$
```

Figure 1 The ACO algorithm for the single machine scheduling total tardiness minimization problem with sequence-dependent setup times.

transition rule expressed in Equations (1) and (2) of Figure 1. In Equation (1), q is a random number and q_0 is a parameter; both are between 0 and 1. Parameter q_0 determines the relative importance of the *exploitation* of existing information and the *exploration* for new solutions. Equation (1) states that the next job will chosen either by a greedy rule (where $q \leqslant q_0$) or by a probabilistic rule (Equation (2)) where $q > q_0$. Equation (2) describes the biased exploration probability rule $p_{ij}^k(t)$ appropriate for the scheduling problem that we treat.

Visibility. For this scheduling problem, two matrices are used to represent the two elements that influence total tardiness: the setup times and the slack times. These two matrices represent the visibility information analogous to the distance matrix in the TSP.

The setup time matrix is of dimension $[n+1, n]$. Note that the last job of the previous period is represented by an additional row. The elements of this matrix are the relative setup times where $\bar{s}_{ij} = s_{ij}/\text{Max}\, s_{ij}$.

The slack time matrix favours respect of due dates and the minimization of total tardiness. It represents the relative slack if job j is produced immediately following job i. The slack of a job is defined as $m_{ij} = \max\{(d_j - p_j - s_{ij}), 0\}$ and the second matrix is composed of elements $\overline{m}_{ij} = m_{ij}/\text{Max}\, m_{ij}$.

We use the relative values for the two matrices of setup times and slack times to facilitate the trade-offs that the algorithm will seek to make between these two elements. The parameters β and δ allow control over the behaviour of the algorithm and the weight that is given to one or the other of these elements in the makeup of the distance measure used in the algorithm.

Selection lists. It is necessary to ensure that a job that has already been placed in the sequence being constructed is not again selected. For each ant, a *Tabu* list of those jobs that cannot again be selected is therefore maintained.

The use of a *candidate list* limits the number of choices for the next job to place in the sequence. In the scheduling problem treated, the candidate list is composed of those jobs having the smallest slack times. The calculation of the slack for job j, where $j \notin Tabu$, following job i, is done according to the definition of m_{ij}. The list of jobs $\{j\}$ is then sorted in increasing order of slack and the first cl of these become the candidate list for job i.

Look-ahead information. Usually, in the transition rule, the selection of the next order to place in the sequence is the result of a compromise between the trail, information on past choices, and the visibility, information derived from the problem parameters. We propose the addition of look-ahead information about the potential of the current partial solution. Michel and Middendorf[35] propose a different form of look-ahead function which works by evaluating the trail intensity resulting from a choice. Closer to our work is Maniezzo[36] who uses a lower bound to estimate the quality of a partial solution and this value replaces the visibility in the application of the transition rule. In our procedure, we compute a look-ahead term using the lower bound and add it to the other information in the transition rule.

The lower bound on the total tardiness of the partial solution, including the job j presumed to be the candidate chosen following job i, is calculated according to the following: $B_{ij} = T_{Q_h} + T_{Q_h'}$. We take the actual value, T_{Q_h}, generated by the current partial solution, Q_h, and add the consequences of one of the candidate choices as well as a lower bound, $T_{Q_h'}$, on the remaining uncompleted portion, Q_h'. The computation of the lower bound for the uncompleted portion follows the method used in the branch-and-bound computations of Ragatz,[26] and later in Tan *et al.*[7]

Equations (1) and (2) related to the transition rule are again modified by the addition of a new term, $\bar{B}_{ij}$, which represents the look-ahead information. Relative values are again used to obtain a normalized measure and the parameter ϕ controls the weight of this information. The lower bound for a partial solution including the candidate job j is calculated as follows: $\bar{B}_{ij} = B_{ij}/\text{Max}\, B_{ij}$. We have carried out numerical experiments to study the performance of this new element in the ACO algorithm and our results are presented in the next section.

Local trail update. To avoid premature convergence, a local trail update is introduced. This updating effects a temporary reduction in the quantity of pheromone for a given adjacent job pair so as to discourage the next ant from choosing the same adjacent job pair during the same cycle. When an ant selects a job, a local update is made to the trail for that job pair according to Equation (3) of Figure 1. In this equation, ρ_ℓ $(0 < \rho_\ell < 1)$ is a parameter that determines the rate of reduction of the pheromone level. In this case $\Delta\tau_{ij} = \tau_0$ which is the initial trail, a small positive quantity. The pheromone reduction is small but sufficient to lower the attractiveness of job pair (i,j). If a good solution including (i,j) is found in the cycle, the global trail update will again increase the pheromone level for job pair (i,j). The exploration of new solutions during a sequence is thus encouraged.

Step 2.3: *Evaluation of solutions and local improvement methods*

We have included two local improvement methods in the ACO algorithm. The first of these two methods is the restricted 3-opt method described by Dorigo and Gambardella.[29] This method has the characteristic of not inverting the complete job sequence, which is important where setup times are sequence dependent. The second method, called Random Search Pairwise Interchange

(RSPI), proceeds to invert each pair of adjacent jobs in turn. This method was used by Rubin and Ragatz[16] either alone or in conjunction with a metaheuristic. The results, of Tan *et al*[7] indicate that the RSPI produces better solution quality than simulated annealing, branch-and-bound and the genetic algorithm if sufficient computation time is allowed.

One of these local improvement methods is applied to the path found by each ant. The choice of method is made by a random (fair coin toss) choice. The restricted 3-opt method, in which one passes only once through the list of adjacent pairs, is applied in order to reduce the computation times, which is particularly important if the problem size is large.

Step 2.4: *Global update of pheromone trail*

The quantity of pheromone to be added to the job pair (i,j) is therefore proportional to the quality of the sequence obtained by each ant. The pheromone trail is updated at the end of a cycle, but only for the job pairs (i,j) in the best solution found in the cycle L_*. Equation (4) of Figure 1 is used for the global trail update. In this equation (4), ρ_g $(0 < \rho_g < 1)$ plays the same role as ρ in the basic algorithm. This change allows both for the evaporation of the trail deposited at the end of previous cycles and the additions to be made at the current cycle. The original authors suggest that the combination of this global trail update along with the new transition rule will improve convergence of the ACO.

In most cases, the values of the parameters in the ACO algorithm follow the suggestions given by Dorigo and Gambardella[29] and in the remaining cases, the values have been determined through direct numerical experimentation. The trail is initialized to the value, $\tau_0 = (n \cdot L_{nn})^{-1}$ where n is the size of the problem and L_{nn} is the result of a solution obtained either randomly or through some simple heuristic. The remaining parameters have been assigned the following values: $\rho_g = \rho_\ell = 0.9$, $m = 10$, $q_0 = 0.9$ and $cl = 0.3 \cdot n$ with a lower bound equal to 10. Further the maximum number of cycles without improvement of the best known solution is fixed at 50. Note that the values of α, β, δ and ϕ have been adjusted empirically according to the problem characteristics (processing time variance, tardiness factor, due dates range) and take values between 1 and 4. For example, where the due-dates are such that tardiness factor is high, more weight is given to the slack matrix.

The next section presents the ACO that the authors have designed for the solution of the scheduling problem with sequence-dependent setup times.

Numerical experiments and results

The numerical experiments use the set of 32 test problems devised by Rubin and Ragatz[16] and available on the Internet at (http://mgt.bus.msu.edu/datafiles.htm). The problem set is divided by size into four groups of problems having 15, 25, 35 and 45 jobs, respectively. Moreover, the eight problems of each group were derived from a $2 \times 2 \times 2$ experimental plan related to three characteristics each of which may be adjusted to one of two levels. These characteristics are the processing time variance (PTV), the tardiness factor (TF) and the due dates range (DDR). The operation times of the jobs are normally distributed with an arbitrary mean of 100 units and the setup times are uniformly distributed with a mean of 9.5 units.

A first comparison seeks to evaluate the impact on solution quality of the look-ahead information that we have introduced into the transition rule. We present the results of two different versions of the ACO for the 32-problem set. One version is as illustrated in Figure 1 while the second does not include look-ahead information but is identical in all other respects. Table 1 shows the percentage difference with respect to the best solution found by the branch-and-bound algorithm for the best ACO solution, the median ACO value and the worst ACO solution. The branch-and-bound results are drawn from Tan *et al.*[7] Those branch-and-bound results that are optimal are indicated by an asterisk, and in other cases computation was stopped after a maximum of five million nodes were generated. The first columns of the table show the problem characteristics. For each algorithm, the times shown are the averages of the 20 solution runs.

From Table 1, we note that, in general, the look-ahead information has allowed an improvement in solution quality. Six exceptions are indicated by a double asterisk in the table. In five of these six cases the quality drop is less than 1%. In the remaining case, problem #705, the median value of the gap with the branch-and-bound solution has dropped from 3 to 5.6%.

There has been an increase in computation times due to the use of look-ahead information. Generally, as Table 1 shows, solution times have increased by less than a factor of two, although in some instances the increase is higher. Currently, look-ahead information is computed every time the transition rule is used, but we feel that a more selective use of this information would allow us to retain the solution quality improvements while imposing a lesser computational penalty.

We analysed our results to determine whether the quality improvements observed have statistical significance. Unfortunately, when many of the results are identical, as is the case here for many problems, the usual statistical tests are unhelpful because of the lack of variance in the data being analysed. However, Conover and Iman's[37] result allows us to use a parametric t-test on the average ranking for problems where, for each algorithm, less than 80% of the solutions obtained are identical.

Our procedure is as follows:

- Each problem is solved 20 times with each of the two algorithms.

Table 1 Experimental ACO results for 32 problems with and without look-ahead information

						ACO without look-ahead information				*ACO including look-ahead information*			
						% to B&B				*% to B&B*			
Problem	*# Jobs*	*PTV*	*TF*	*DDR*	*B&B*	*Best*	*Median*	*Worst*	*Average run time (s)*†	*Best*	*Median*	*Worst*	Average run *time (s)*†
Prob401	15	L	L	N	90*	0.0	2.2	15.6	0.25	0.0	4.4	7.8	1.25
Prob402	15	L	L	W	0*	0.0	0.0	0.0	0‡	0.0	0.0	0.0	0.05
Prob403	15	L	M	N	3418*	0.0	1.9	2.8	1.15	0.0	1.1	2.1	1.45
Prob404	15	L	M	W	1067*	0.0	0.0	4.1	0.20	0.0	0.0	0.0	1.35
Prob405	15	H	L	N	0*	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0‡
Prob406	15	H	L	W	0*	0.0	0.0	0.0	0‡	0.0	0.0	0.0	0‡
Prob407	15	H	M	N	1861*	0.0	0.0	6.4	0.90	0.0	0.0	1.1	1.45
Prob408	15	H	M	W	5660*	0.0	1.4	2.5	1.00	0.0	1.1	1.5	1.45
Prob501	25	L	L	N	264	−0.4	1.1	4.5	5.70	−1.1	0.8	1.9	7.15
Prob502	25	L	L	W	0*	0.0	0.0	0.0	0.10	0.0	0.0	0.0	0.15
Prob503	25	L	M	N	3511	−0.2	0.3	1.9	3.50	−0.4	0.3	0.9	7.80
Prob504	25	L	M	W	0*	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.20
Prob505	25	H	L	N	0*	0.0	0.0	0.0	0.10	0.0	0.0	0.0	0.10
Prob506	25	H	L	W	0*	0.0	0.0	0.0	0.10	0.0	0.0	0.0	0.10
Prob507	25	H	M	N	7225	0.6	2.2	3.4	7.60	0.7**	1.8	3.7**	9.80
Prob508	25	H	M	W	2067	−3.4	12.2	24.0	6.85	−5.9	5.9	14.2	8.55
Prob601	35	L	L	N	30	−40.0	−3.3	40.0	16.25	−46.7	−13.3	6.7	29.75
Prob602	35	L	L	W	0*	0.0	0.0	0.0	0.35	0.0	0.0	0.0	0.40
Prob603	35	L	M	N	17774	0.0	0.6	2.2	21.75	−0.5	0.1	0.3	32.20
Prob604	35	L	M	W	19277	−0.1	0.5	1.2	23.20	−0.8	0.3	1.3**	32.15
Prob605	35	H	L	N	291	−13.4	−4.8	7.9	13.74	−15.1	−8.8	−1.0	30.95
Prob606	35	H	L	W	0*	0.0	0.0	0.0	0.30	0.0	0.0	0.0	0.35
Prob607	35	H	M	N	13274	−1.9	−0.4	2.2	23.50	−1.4**	−1.6	0.6	27.90
Prob608	35	H	M	W	6704	−21.8	−17.8	−13.5	0.40	−29.4	−34.8	−20.1	33.00
Prob701	45	L	L	N	116	−8.6	−3.4	19.0	32.60	−11.2	−5.6	0.0	83.15
Prob702	45	L	L	W	0*	0.0	0.0	0.0	0.90	0.0	0.0	0.0	0.95
Prob703	45	L	M	N	27097	−1.0	−0.2	0.5	14.55	−1.6	−1.0	−0.5	91.75
Prob704	45	L	M	W	15941	−2.3	−0.2	1.0	36.25	−2.8	−1.1	−0.3	89.15
Prob705	45	H	L	N	234	−0.6	3.0	25.6	61.05	−5.1**	5.6**	15.4	77.55
Prob706	45	H	L	W	0*	0.0	0.0	0.0	0.80	0.0	0.0	0.0	0.90
Prob707	45	H	M	N	25070	−2.0	−1.0	−0.1	6.30	−4.2	−3.3	−2.9	78.55
Prob708	45	H	M	W	24123	−2.2	−0.4	0.5	47.56	−3.2**	−1.7	−0.6	84.70

Legend: B&B = branch-and-bound method; PTV = processing time variance; TF = tardiness factor; DDR = due date range; L = low; H = high; M = moderate; N = narrow; W = wide.
*Indicates optimum solution.
**Look-ahead information has not improved solution quality.
†Intel Pentium III personal computer (733 MHz; 256 MB RAM).
‡Solution times less than 1 s for each of the 20 runs for a problem.

- If 16 (80%) or more of the solutions found with either algorithm are identical, the problem is not considered in the comparison. Thus, 14 of the 32 original problems were rejected.
- For each of the 18 remaining problems, the 20 results for each algorithm are combined into one table and ranked from best to worst. Thus there are a total of 360 cases for each algorithm version.
- For each algorithm, the average rank is calculated and compared.

Table 2 presents the results of this statistical test which allows us to draw the conclusion that the version of the algorithm that includes look-ahead information offers solutions of better quality. The 18 problems considered by this test are listed in column 1 of Table 1 in bold characters.

These results have encouraged us to conclude that using look-ahead information contributes significantly to improving solution quality and we therefore proceeded to compare this version of the ACO algorithm to the genetic, simulated annealing and RSPI local improvement heuristics and the branch-and-bound algorithm presented in Tan *et al.*[7] Each of the problems was solved 20 times both in the results quoted from the work of Tan *et al*[7] and in our ACO algorithm experiments. We worked on two different computers. In an attempt to reproduce as closely as possible the experimental

Table 2 t-Test for a comparison of two mean ranks for the 18 problems considered: two samples with different variances. The significance level for the test is 5%

	Without look-ahead information		*Including look-ahead information*
Mean rank	24.189		16.811
Variance	137.741		98.591
Number of cases	360		360
t-value		9.356	
2-tailed prob.		0.000	

conditions of these latter authors who worked on an Intel Pentium 90 MHz machine, we obtained an Intel Pentium 100 Mhz computer. We were, however, unable to compare other machine characteristics such as the available RAM, and we realize that our comparisons can only be approximate. The ACO algorithm is coded in the C language. Note that the algorithms compared in Tan *et al*[7] were coded in different languages.

Table 3 presents the details of this comparison in a form similar to that of Table 1 and shows the same three performance indicators. Tan *et al*[7] presented a statistical analysis showing that the RSPI method performed better than the other approaches tried and so it is used in the comparisons with our results that follow.

The values of the ACO parameters α, β, δ, and ϕ were set to identical values for all problems having the same characteristics. These parameters were adjusted following empirical tests on the more realistic larger-sized problems. The results of a number of tests allow us to conjecture that the ACO algorithm could have performed better for the small problems had we specifically adjusted the parameters for them.

In Table 3, the results identified by the symbol '+' are those where the ACO is better than RSPI. Those indicated by the symbol '++' show where RSPI has produced a better result. We note that the smaller problems having 15 and 25 jobs are more easily solved by the RSPI. The ACO algorithm has a better solution in only three such cases as compared to 16 for the RSPI. For the 35-job problem group, performance of the two methods is about the same. For the larger problems, the ACO performs 12 times better compared to the RSPI method, which is six times better. We observe that the ACO algorithm has a relative advantage for the larger problems.

A statistical analysis allowed the confirmation of the summary conclusions. Tan *et al*[7] graciously furnished us with their detailed results to allow this analysis to be made. As in the statistical analysis of Table 2, the problems for which the two heuristics produced 80% or more identical results were ignored to avoid a preponderance of ties. The 14 problems considered are indicated in column one of Table 3 in bold characters. Table 4 shows the statistical analyses on the average rank grouped by problem size. Because the problems of 15 jobs showed little variation in results among methods they were not considered in this analysis. Part (a) of Table 4 shows that the RSPI method has the better performance for problems of 25 jobs, part (b) indicates that the two heuristics offer equal performance and part (c) shows that for the larger problems, the ACO algorithm is superior. In summary, the statistical analysis confirms our earlier remarks.

The number of times that a heuristic is superior to another does not, however, convey the degree of difference in performance. For this reason, the average percentage difference for those cases where one method is better than the other has been calculated, using the median results. The RSPI heuristic shows a 2.87% lead over the ACO in cases where it is better, but the ACO has a 13.89% lead over the RSPI in those cases where the ACO is better. We note, therefore, that using the ACO can produce a much better result than the RSPI but it is not likely to be, even in the worst case, much worse than the RSPI result.

Particular cases in the results in Table 3 have been noted:

- the RSPI has better performance for the eighth problem of each group, that problem where PTV is high, TF is moderate and DDR is large;
- the largest difference in results is observed for problem #508 between the median and the worst case results.

It is likely that these differences could be lessened were the parameters of the ACO adjusted to the characteristics of these specific problems.

In other problems the ACO algorithm produces much better results:

- for the first and fifth problems (those with low TF and close DDR) in the 35 and 45 job problem sets;
- for the odd numbered 45-job problems of Table 3 (indicated by a double asterisk);
- for all problems having closely spaced due-dates, the ACO algorithm is better for all three performance measures;

We note that the ACO algorithm uses specific information concerning the setups and the due dates in the form of the two distance matrices and it is, no doubt, this information that allows a greater measure of success in meeting the due dates. Finally the second and the sixth problems of each group proved easy for both solution methods.

Table 5 presents further statistical t-test on the mean rank for those problems having the same PTF, TF and DDR characteristics. Again those problems for which 80% of the results are identical have been set aside to avoid a preponderance of ties. The second and sixth problems were also set aside since they were easily solved to optimality by both approaches. Analyses (a), (b) and (d) of Table 5 show that the ACO algorithm is superior for the first, third and fifth problems of each job-size group. Analyses (c), (e) and (f) of

Table 3 Comparison of the performance of the ACO algorithm with the methods reported in Tan *et al*:[7] branch-and-bound (B&B), genetic algorithm (GA), simulated annealing (SA), and the RSPI local improvement method

						GA (Rubin and Ragatz[16])			*SA (Tan and Narasimhan[27])*			*RSPI (Rubin and Ragatz[16])*				*ACO*			
						% to B&B			*% to B&B*			*% to B&B*				*% to B&B*			
Problem	*# Jobs*	*PTV*	*TF*	*DDK*	*B&B*	*Best*	*Median*	*Worst*	*Best*	*Median*	*Worst*	*Best*	*Medium*	*Worst*	*Average run time (s)*†	*Best*	*Medium*	*Worst*	*Average run time (s)*‡
Prob401	15	L	L	N	90*	0.0	4.4	4.4	0.0	3.3	7.8	0.0	0.0^{++}	3.3^{++}	360	0.0	4.4	7.8	11.80
Prob402	15	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.35
Prob403	15	L	M	N	3418*	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0^{++}	0.2^{++}	360	0.0	1.1	2.1	13.50
Prob404	15	L	M	W	1067*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	12.7	360	0.0	0.0	0.0^{+}	11.05
Prob405	15	H	L	N	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.35
Prob406	15	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.30
Prob407	15	H	M	N	1861*	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.4^{++}	360	0.0	0.0	1.1	13.75
Prob408	15	H	M	W	5660*	0.0	0.0	0.9	0.0	0.0	0.6	0.0	0.0^{++}	0.9^{++}	360	0.0	1.1	1.5	13.20
Prob501	25	L	L	N	264	0.0	1.5	3.8	0.8	1.9	4.2	0.8	0.8	1.5^{++}	960	−1.1^{+}	0.8	1.9	85.90
Prob502	25	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.70
Prob503	25	L	M	N	3511	−0.4	0.2	0.9	−10.4	−9.9	−9.6	−0.4	−0.4^{++}	−0.4^{++}	960	−0.4	0.3	0.9	88.65
Prob504	25	L	M	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.50
Prob505	25	H	L	N	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	–$^{\#}$	960	0.0	0.0	0.0^{+}	1.60
Prob506	25	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.60
Prob507	25	H	M	N	7225	2.1	6.1	9.6	0.0	1.1	2.4	0.0^{++}	0.1^{++}	0.2^{++}	960	0.7	1.8	3.7	126.90
Prob508	25	H	M	W	2067	−5.9	−5.9	−1.5	−7.4	−7.4	−3.1	−7.4^{++}	−7.4^{++}	−7.4^{++}	960	−5.9	5.9	14.2	102.90
Prob601	35	L	L	N	30	76.7	150.0	193.3	20.0	43.3	96.7	20.0	31.7	46.7	1800	−46.7^{+}	−13.3^{+}	6.7^{+}	386.50
Prob602	35	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1800	0.0	0.0	0.0	3.75
Prob603	35	L	M	N	17774	−0.7	0.4	2.2	0.1	0.8	1.4	0.1	0.2	0.7	1800	−0.5^{+}	0.1^{+}	0.3^{+}	776.65
Prob604	35	L	M	W	19277	0.2	1.0	2.6	−0.2	0.7	2.8	−0.2	−0.1^{++}	0.1^{++}	1800	−0.8^{+}	0.3	1.3	382.05
Prob605	35	H	L	N	291	13.7	37.3	56.7	−6.2	−1.2	8.9	−6.2	−4.1	−2.1^{++}	1800	−15.1	−8.8^{+}	−1.0	413.10
Prob606	35	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1800	0.0	0.0	0.0	3.65
Prob607	35	H	M	N	13274	5.0	6.6	7.6	−1.7	−0.1	1.7	−1.7^{++}	−0.8^{++}	−0.2^{++}	1800	−1.4	−0.1	0.6	715.45
Prob608	35	H	M	W	6704	−29.0	−28.6	−26.7	−29.4	−29.0	−26.9	−29.4	−29.2^{++}	−29.0^{++}	1800	−29.4	−24.8	−20.1	880.35
Prob701	45	L	L	N	116	57.8	82.8	118.1	1.7	29.3	40.5	1.7	22.4	28.4	3600	−11.2^{+}	−5.6^{+}	0.0^{+}	1197.95
Prob702	45	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3600	0.0	0.0	0.0	11.05
Prob703	45	L	M	N	27097	−0.1	1.5	2.5	−1.3	0.2	1.3	−1.3	−0.2	0.1	3600	−1.6^{+}	−1.0^{+}	−0.5^{+}	2638.25
Prob704	45	L	M	W	15941	−2.4	−1.6	1.0	−3.3	−1.2	1.0	−3.3^{++}	−2.7^{++}	1.8^{++}	3600	−2.8	−1.1	−0.3	1886.65
Prob705	45	H	L	N	234	53.4	89.3	114.5	8.5	20.5	49.1	8.5	18.8	23.1	3600	−5.1^{+}	5.6^{+}	15.4^{+}	1168.85
Prob706	45	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3600	0.0	0.0	0.0	9.60
Prob707	45	H	M	N	25070	−1.1	1.8	6.3	−3.4	−2.5	−0.8	−3.4	−2.9	−2.6	3600	4.2^{+}	−3.3^{+}	−2.9^{+}	2524.65
Prob708	45	H	M	W	24123	2.8	7.2	10.1	−4.0	−3.2	−2.0	−4.0^{++}	−3.9^{++}	−3.3^{++}	3600	−3.2	−1.7	−0.6	2336.10

Legend: B&B = branch-and-bound method; PTV = processing time variance; TF = tardiness factor; DDR = due date range; L = low; H = high; M = moderate; N = narrow; W = wide.
*Indicates optimum solution.
$^{+}$ACO is better than RSPI.
$^{++}$RSPI is better than ACO.
$^{\#}$Divides by 0, the worst solution is 6.0.
†Pentium 90 MHz personal computer. ‡Pentium 100 MHz personal computer.

Table 4 *t*-Test for a comparison of two mean ranks for the 14 problems considered: two samples with equal variances. (a) 2 problems of 25 jobs; (b) 6 problems of 35 jobs; and (c) 6 problems of 45 jobs. The significance level for these tests is 5%

	(a): 25 job problems		*(b): 35 job problems*		*(c): 45 job problems*	
	ACO	*RSPI*	*ACO*	*RSPI*	*ACO*	*RSPI*
Mean rank	23.85	17.15	21.23	19.77	17.28	23.73
Variance	138.90	95.86	151.78	114.97	125.42	121.99
Number of cases	40	40	120	120	120	120
t-value	2.766		0.984		−4.492	
2-tailed prob.	0.007		0.326		0.000	

Table 5 show the RSPI to perform better for the fourth, seventh and eight problems of each group. In each *t*-test on mean ranking, a variance comparison test was also used before.

We have already indicated that comparisons of solution times are difficult not only because of the nature of the computers used but also because of the nature of the algorithms. Tan *et al*[7] have used fixed solution times of 6, 16, 30 and 60 min for problems of size 15, 25, 35 and 45, jobs respectively. The ACO algorithm uses quite different stopping criteria. Despite these differences and to further our understanding of the results, we show the average computing time required by the ACO algorithm in the final column of Table 3.

In the measure that the results of the two sets of numerical experiments can be compared, we note that the ACO algorithm does not approach the solution times of the RSPI even in the worst cases. For example, the longest solution time for solution of 45-job problems, when run on a 100 MHz Pentium computer, is about 44 min as compared to the 60 min allowed the RSPI. When run on a current computer, solution time for problems of this size was under two minutes as shown in the final column of Table 1.

Table 6 presents a statistical analysis similar to the previous ones concerning the equality of mean ranking for the set of 14 problems retained. Neither algorithm was shown to dominate the other in this test.

Table 6 *t*-Test on mean ranking for the 14 problems considered: two samples with equal variances. The significance level for these tests is 5%

	ACO	*RSPI*
Mean rank	19.911	21.089
Variance	143.616	120.432
Number of cases	280	280
t-value	−1.214	
2-tailed prob.	0.225	

In summary, our statistical tests show that for the Rubin and Ragatz[16] problem set the ACO algorithm is competitive in solution quality and has shorter computation times than the best performing method tried by Tan *et al.*[7] The RSPI seems to perform somewhat better on small problems while the ACO algorithm is better for the larger problems.

A new set of larger test problems, available from the authors, were generated randomly using a procedure defined by Ragatz.[26] Each problem of 55, 65, 75 and 85 jobs, respectively was solved 20 times by the ACO algorithm on a current computer having an Intel Pentium III 733 MHz processor with 256 MB RAM, running under Windows 2000. Table 7 presents the results in the format of the relevant previous tables and adds the mean and standard deviation of the results obtained. We offer these results to provide a basis for future comparisons of larger-sized problems more typical of real order books.

Conclusions

The single machine scheduling problem with sequence-dependent setup times for total tardiness minimization is NP-hard, but we have found that the ACO algorithm that we have presented is able to efficiently solve problems of a size encountered in industrial situations.

The modifications and adaptations of the original ACO algorithm that we propose have contributed to the performance of this method in the cases studied. The use of look-ahead information in the selection of the next job during the

Table 5 (a)–(f): *t*-Test for a comparison of two mean ranks for problems having same characteristics (the *n*th problem of each group). Note that the second and the sixth problems are not included in this analysis because a lack of variances. The tests (a), (b), (e) and (f) have two samples with equal variances and the tests (c) (d) have two samples with unequal variances. The significance level for these tests is 5%

	(a)		*(b)*		*(c)*		*(d)*		*(e)*		*(f)*	
	ACO	*RSPI*	*ACO*	*RSPI*	*ACO*	*RSPI*	*ACO*	*RSPI*	*ACO*	*RSPI*	*ACO*	*RSPI*
Mean rank	12.73	28.27	13.88	27.13	27.30	13.70	13.48	27.53	23.75	17.25	30.00	11.00
Variance	83.01	55.61	70.25	113.01	120.88	57.33	115.44	55.46	131.71	116.29	53.82	33.53
Number of cases	60	60	40	40	40	40	40	40	60	60	40	40
t-value	−10.219		−6.190		6.443		−6.797		3.197		12.858	
2-tailed prob.	0.000		0.000		0.000		0.000		0.002		0.000	

Table 7 Results for larger test problems (55, 65, 75 and 85 jobs)

					ACO					
Problem	*# Jobs*	*PTV*	*TF*	*DDR*	*Best*	*Median*	*Worst*	*Average*	*Std dev.*	*Average run* time (s)*
Prob551	55	L	L	N	212	237.5	273	241.3	18.6	167.60
Prob552	55	L	L	W	0	0.0	0	0.0	0.0	2.25
Prob553	55	L	M	N	40 828	41104.0	41 303	41110.9	116.8	227.50
Prob554	55	L	M	W	15 091	15576.0	16 423	15621.3	320.9	221.20
Prob555	55	H	L	N	0	0.0	12	2.1	3.8	100.90
Prob556	55	H	L	W	0	0.0	0	0.0	0.0	2.10
Prob557	55	H	M	N	36 489	37357.5	37 973	37308.6	374.4	163.15
Prob558	55	H	M	W	20 624	21417.0	22 457	21386.7	427.1	236.30
Prob651	65	L	L	N	295	317.5	350	319.1	16.3	354.45
Prob652	65	L	L	W	0	0.0	0	0.0	0.0	4.00
Prob653	65	L	M	N	57 779	58249.5	58 653	58266.8	223.3	440.70
Prob654	65	L	M	W	34 468	35399.0	36 107	35365.4	419.0	466.65
Prob655	65	H	L	N	13	24.5	38	25.3	6.2	347.45
Prob656	65	H	L	W	0	0.0	0	0.0	0.0	3.95
Prob657	65	H	M	N	56 246	57037.5	57 825	57027.5	470.0	352.65
Prob658	65	H	M	W	29 308	30099.5	31 074	30155.9	524.2	349.90
Prob751	75	L	L	N	63	313.0	368	311.6	23.2	610.20
Prob752	75	L	L	W	0	0.0	0	0.0	0.0	6.55
Prob753	75	L	M	N	78 211	78541.5	79 088	78604.1	233.7	738.85
Prob754	75	L	M	W	35 826	37592.0	38 333	37514.3	492.5	537.10
Prob755	75	H	L	N	0	0.0	0	0.0	0.0	7.00
Prob756	75	H	L	W	0	0.0	0	0.0	0.0	7.85
Prob757	75	H	M	N	61 513	62201.0	63 284	62216.8	442.4	564.55
Prob758	75	H	M	W	40 277	42271.0	42 964	42018.5	806.4	719.70
Prob851	85	L	L	N	453	515.5	557	511.0	30.2	883.80
Prob852	85	L	L	W	0	0.0	0	0.0	0.0	10.55
Prob853	85	L	M	N	98 540	98957.0	99 250	98949.0	212.8	1075.75
Prob854	85	L	M	W	80 693	81785.5	82 728	81702.6	567.0	1301.25
Prob855	85	H	L	N	333	374.5	409	373.5	21.7	971.00
Prob856	85	H	L	W	0	0.0	0	0.0	0.0	10.70
Prob857	85	H	M	N	89 654	90574.5	91 447	90569.2	501.8	905.60
Prob858	85	H	M	W	77 919	79368.5	80 612	79299.5	689.4	1057.80

Legend: B&B = branch-and-bound method; PTV = processing time variance; TF = tardiness factor; DDR = due date range; L = low; H = high; M = moderate; N = narrow; W = wide.
*Intel Pentium III personal computer (733 MHz; 256 MB RAM).

construction of a solution has been shown, with the support of statistical tests, to improve solution quality. The extra computational load can be an obstacle to use of this information in an industrial context. It is our opinion, however, that it will be possible to obtain much of the benefit of the look-ahead information without undue computational penalty by a parsimonious use of this computation in the transition rule. We have found it useful to add a second distance matrix so that we may explicitly represent both the setup times and due dates.

The ACO algorithm described here has performed competitively with the best results obtained by Tan *et al*[7] for other heuristics. Overall, the solutions obtained by the ACO algorithm for larger problems are of equal or better quality and the computational times are appreciably lower. For the smaller problems we did not find this advantage to hold, but such problems are, in our experience, less representative of those found in industry. We have added several larger cases to the problem set and we will supply them to other authors upon request for further numerical experiments.

Our results have encouraged us to incorporate the ACO algorithm presented in our practical work on the scheduling of aluminium casting.

Acknowledgements– This research was financially supported by the FCAR (Québec) and NSERC (Canada) granting councils. The authors also wish to thank Professor Paul A. Rubin and his colleagues for having generously furnished details of their own results and for their valuable comments.

References

1 MacCarthy BL and Liu J (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *Int J Prod Res* **31**: 59–79.

2 McKay KN and Wiers VCS (1999). Unifying the theory and practice of production scheduling. *J Manuf Systems* **18**: 241–255.

3 Gravel M, Price WL and Gagné C (2000). Scheduling jobs in a Alcan aluminium factory using a genetic algorithm. *Int J Prod Res* **38**: 3031–3041.

4 Gravel M, Price WL and Gagné C (2002). Scheduling continuous casting of aluminum using a multiple-objective ant colony optimization metaheuristic. *Eur J Opl Res* (in press).

5 Colorni A, Dorigo M and Maniezzo V (1991). Distributed optimization by ant-colonies. In: Verela F and Bourgine P (eds). *Proceedings of the First European Conference on Artificial Life (ECAL'91)*. MIT Press: Cambridge MA, pp 134–142.

6 Dorigo M (1992). *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy.

7 Tan KC, Narasimhan R, Rubin PA and Ragatz GL (2000). A comparison of four methods for minimizing total tardiness on a single processor with sequence-dependent setup times. *Omega* **28**: 313–326.

8 Du J and Leung JY (1990). Minimizing total tardiness on one machine is NP-hard. *Math Opns Res* **15**: 483–494.

9 Wisner JD and Siferd SP (1995). A survey of U.S. manufacturing practices in make-to-order machine shops. *Prod Inventory Mngt J* **1**: 1–7.

10 Koulamas C (1994). The total tardiness problem: review and extensions. *Opns Res* **42**: 1025–1041.

11 Emmons H (1969). One-machine sequencing to minimize certain functions of job tardiness. *Opns Res* **17**: 701–715.

12 Lawler EL (1977). A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Ann Dis Math* **1**: 331–342.

13 Bauer A, Bullnheimer B, Hartl RF and Strauss C (1999). An ant colony optimization approach for the single machine total tardiness problem. In: Angeline PJ, Michalewicz Z, Schoenauer M, Yao X and Zalzala A (eds). *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*. IEEE Press: Piscataway, NJ.

14 den Besten M, Stützle T and Dorigo M (2000). Ant colony optimization for the total weighted tardiness problem. In: Deb *et al* (eds). *Parallel Problem Solving from Nature: 6th International Conference*. Springer: Berlin.

15 Merkle D and Middendorf M (2000). An ant algorithm with new pheromone evaluation rule for the total tardiness problem. In: Cagnioni *et al* (eds). *Real-World Applications of Evolutionary Computing*. Springer: Berlin, pp 287–296.

16 Rubin PA and Ragatz GL (1995). Scheduling in a sequence dependent setup environment with genetic search. *Computers Opns Res* **22**: 85–99.

17 Wilbrecht JK and Prescott WB (1969). The influence of setup time on job performance. *Mngt Sci* **16**: B274–B280.

18 Wortman DB (1992). Managing capacity: getting the most from your firms assets. *Ind Eng* **24**: 47 49.

19 Pinedo M (1995). *Scheduling Theory, Algorithms and Systems*. Prentice-Hall: Englewood Cliffs, NJ.

20 Das SR, Gupta JND and Khumawala BM (1995). A saving index heuristic algorithm for flowshop scheduling with sequence dependent set-up times. *J Opl Res Soc* **46**: 1365 1373.

21 Franca PM, Gendreau M, Laporte G and Muller FM (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *Int J Prod Econ* **43**: 79–89.

22 Conway RN, Maxwell WL and Miller LW (1967). *Theory of Scheduling*. Addison-Wesley: Reading, MA.

23 Allahverdi A, Gupta JND and Aldowaisan T (1999). A review of scheduling research involving setup considerations. *Omega* **27**: 219–239.

24 Yang W-H and Liao C-J (1999). Survey of scheduling research involving setup times. *Int J Systems Sci* **30**: 143–155.

25 Lee YH, Bhaskaran K and Pinedo M (1997). A heuristic to minimize the total weighted tardiness with sequence dependent setups. *IIE Trans* **29**: 45 52.

26 Ragatz GL (1989). Scheduling to minimize tardiness on a single machine with sequence dependent setup times. Working paper, Department of Management, Michigan State University.

27 Tan KC and Narasimhan R (1997). Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega* **25**: 619–634.

28 Deneubourg JL, Pasteels JM and Verhaeghe JC (1983). Probabilistic behaviour in ants: a strategy of errors? *J Theor Biol* **105**: 259–271.

29 Dorigo M and Gambardella LM (1997). Ant colonies for the traveling salesman problem. *BioSystems* **43**: 73–81.

30 Kanellakis P-C and Papadimitriou CH (1980). Local search for the asymmetric traveling salesman problem. *Opns Res* **28**: 1087–1099.

31 Lawler EL, Lenstra JK, Rinnooy-Kan AHG and Shmoys DB (1985). *The Traveling Salesman Problem*. Wiley: New York.

32 Dorigo M, Maniezzo V and Colorni A (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Trans Systems, Man Cybern* **26**: 29–41.

33 Dorigo M, Di Caro G and Gambardella LM (1999). Ant algorithms for discrete optimization. *Artif Life* **5**: 137–172.

34 Bonabeau E, Dorigo M and Theraulaz G (1999). Swarm Intelligence: From Nature to Artificial Systems. Oxford University Press: New York.

35 Michel R and Middendorf M (1999). An ACO algorithm for the shortest common supersequence problem. In: Corne D, Dorigo M and Glover F (eds). *New Ideas in Optimization*. McGraw-Hill: New York.

36 Maniezzo V (1998). Exact and approximate nondeterministic tree-search procedures for the quadratic assignement problem. *INFORMS J Comput* **11**: 358–369.

37 Conover WJ and Iman RL (1981). Rank transformations as a bridge between parametric and nonparametric statistics. *The Am Stat* **35**: 124–129.

Received August 2001;
accepted February 2002 after one revision