Available online at www.sciencedirect.com
ScienceDirect
Int. J. Production Economics 112 (2008) 742–755
international journal of production economics
www.elsevier.com/locate/ijpe

Development of new features of ant colony optimization for flowshop scheduling ☆

B.M.T. Lin[a,*], C.Y. Lu[b], S.J. Shyu[c], C.Y. Tsai[c]

[a]Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Taiwan 300, Taiwan
[b]Department of Information Management, National Chi Nan University, Taiwan 545, Taiwan
[c]Department of Computer Science and Information Engineering, Ming Chuan University, Taiwan 333, Taiwan

Received 5 October 2004; accepted 2 June 2007
Available online 10 July 2007

Abstract

Ant colony optimization (ACO) is a meta-heuristic based on the indirect communication of a colony of artificial ants mediated by pheromone trails with the collaboration and knowledge-sharing mechanism during their food-seeking process. In this study, we introduce two new features that are inspired from real ant behavior to develop a new ACO algorithm to produce better solutions. The proposed ACO algorithm is applied to two NP-hard flowshop scheduling problems. The first problem is to minimize the total completion time and the second is to minimize a combination of makespan and total completion time. Numerical results indicate that the proposed new features of ACO are very effective and the synergy of combining all the new features for the proposed ACO algorithm can solve the two problems to a certain scale by producing schedules of better quality.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Ant colony optimization; Flowshop scheduling; Total completion time; Makespan; Bicriteria

1. Introduction

Scheduling refers to the allocation of limited resources to specific tasks over a planning horizon so as to reach an optimal or satisfactory decision. It is a decision-making process that has a goal as the optimization of one or more objectives (Pinedo, 2002). Scheduling theory has been growing since the first algorithm developed in the 1950s. Many scheduling algorithms have been successfully applied to manufacturing and production systems, computer resource management and many real-life situations (Brucker, 1997; Pinedo, 2002). Although several dispatching rules can provide optimal decisions to some specific scheduling problems, there are many computationally difficult scheduling problems encountered in real applications. For large-scale instances, it is very unlikely to devise good algorithms for finding optimal solutions to these hard problems in an acceptable time. Therefore, it is reasonable for the management to obtain near-optimal solutions in a timely manner by using

☆This research was partially supported by the National Science Council of the ROC under contract number NSC-92-2416-H-260-006.
*Corresponding author. Tel.: +886 3 5131472; fax: +886 3 5729915.
E-mail address: bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

0925-5273/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2007.06.007

approximation heuristics and meta-heuristics such as genetic algorithms (Holland, 1975; Goldberg, 1989), simulated annealing (Kirkpatrick et al., 1983), tabu search (Glover, 1990) and ant colony optimization (Colorni et al., 1991). Among the meta-heuristic algorithms for solving hard combinatorial optimization problems, the ant colony optimization (henceforth, the acronym ACO will be used throughout this paper) was the most recently developed. The theme of the ACO is based on the indirect communication of a colony of real ants mediated by pheromone trails with the collaboration and knowledge-sharing mechanism during their food-seeking process. Since it was introduced, the ACO has been applied to several optimization problems with distinguished performance. In this paper, we exert to develop new features to enhance the problem-solving capability of the ACO. Two scheduling problems will be used to illustrate how our concepts are applied and realized.

The problems under study in this paper are a two-machine flowshop scheduling problem to minimize the total completion time and a two-machine flowshop scheduling problem to minimize a weighted combination of the total completion time and the makespan. In the first problem, the objective is a measurement concerning the minimum average flow time of all jobs in the manufacturing environment as well as the minimum average waiting time for customers. The objective of the second problem concerns both the satisfaction of customers and companies because makespan, i.e. the maximum completion time of all jobs, reflects the facility utilization. In this study, we not only adopt the ACO algorithm to solve the scheduling problems but also propose some new features inspired from real ant behavior to develop a new ACO algorithm for improving the quality of solutions.

This paper is organized into six sections. In Section 2, we introduce the basic concepts and previous applications of the ACO. Formal problem definitions and related works for the studied problems are presented in Section 3. Section 4 is devoted to describing the proposed new features of the ACO for the two scheduling problems. Section 5 includes computational experiments designed to evaluate the effectiveness of the proposed new features for the studied problems. Comparisons with existing ACO algorithms follow. Finally, concluding remarks and discussions are given in Section 6. Before proceeding to the main text, we would like to emphasize that the goal of this paper is not to show the superiority of the ACO algorithm over other meta-heuristics, but to propose new features of the ACO algorithms for dealing with the two sequencing problems.

2. Ant colony optimization

The ACO algorithm was first introduced by Colorni et al. (1991) and the first ant system (AS) was proposed by Dorigo (1992) in his Ph.D. thesis. The ACO is a meta-heuristic algorithm, which is inspired by real ant colony behavior in finding a shortest path from a food source to the nest without using visual cues by exploiting pheromone information (Beckers et al., 1992; Goss et al., 1989; Hölldobler and Wilson, 1994). When ant colonies are seeking for food, they leave a kind of chemical compositions, called *pheromone*, on their trails. The more the ants walk through the path, the more the pheromone left on the ground. Because the next ant will choose one path with a probability proportional to the amount of pheromone, this positive feedback process will finally develop a common path from their nest to the food source. Real ants have the following key characteristics (Colorni et al., 1991): (1) Real ants prefer to choose the trail with a higher intensity of pheromone. (2) The shorter the distance of a path, the more the pheromone left on it. (3) Real ants conduct indirect communication via pheromone. The above behavior of self-organizing real ants for finding the shortest path fostered the development of the ACO. Artificial ants cooperate to come up with the solution by exchanging information via depositing pheromone on paths.

The algorithmic steps of the ACO designed by Dorigo and Gambardella (1997) are outlined as the following. For details of the ACO, the reader is referred to the original paper.

Algorithm: ant colony optimization

1. Initialize
   Set initial pheromone on each edge
2. Loop/* at this level each loop is called an iteration */
   Each ant is positioned on a starting node
   2.1. Loop/* at this level each loop is called a step */
   Each ant applies a state transition rule to incrementally build a solution and a local pheromone updating rule

Until all ants have built complete solutions

2.2. A global pheromone updating rule is applied

Until stopping criterion

3. Output the global best tour.

The ACO has been successfully used to solve many discrete optimization problems such as the traveling salesman problems (Colorni et al., 1996; Dorigo et al., 1996; Dorigo and Gambardella, 1997; Stützle and Dorigo, 1999a), the quadratic assignment problems (Maniezzo et al., 1994; Gambardella et al., 1999; Stützle and Dorigo, 1999b), the vehicle routing problems (Bullnheimer et al., 1999a, b), the partitioning problems (Kuntz and Snyers, 1994), the vertex cover problems (Shyu et al., 2004a), the generalized spanning tree problem (Shyu et al., 2003), the telecommunications networks problems (Schoonderwoerd et al., 1997; Di Caro and Dorigo, 1998; Shyu et al., 2004c) and the image processing problems (Ramos and Almeida, 2000), just to name a few. For more researches and algorithmic details about the ACO, the reader is referred to Dorigo et al. (1996, 1999) and Dr. Dorigo's website http://iridia.ulb.ac.be/~mdorigo/ACO/.

3. Flowshop scheduling and related ACO works

Flowshop scheduling problems have been extensively studied in the scheduling literature since Johnson's seminal work (Johnson, 1954). A two-machine flowshop consists of two machines arranged in a pipeline way. A set of jobs $N = \{1, 2, \ldots, n\}$ is available from time zero onwards for processing on the machines. Each job i of N consists of two operations or subtasks that will be processed on the two machines. The processing times required by the two operations are a_i and b_i. The second operation cannot start until the first operation is finished and the second machine is free. Preemption on either operation is not allowed and the same job order is applied on every machine. Each job can be processed on a machine at a time and each machine can process only one job at a time. In two-machine flowshop scheduling, Johnson proposed an $O(n \log n)$ algorithm to minimize the makespan $C_{\max}$, which is the largest completion time of the jobs $\max_{1 \leqslant i \leqslant n}\{C_i\}$, where C_i is the completion of job i. While makespan reflects facility utilization from the management's point of view, total completion time, or the sum of completion times of all jobs $\Sigma_{1 \leqslant i \leqslant n} C_i$, is an important indicator for the company's WIP or service level. Unfortunately, to minimize the total completion time in a two-machine flowshop is unary NP-hard (Garey et al., 1979). In this paper, we first consider the problem of minimizing total completion time. The second problem is focused on a bicriteira objective with a weighted combination of total completion time and makespan (Nagar et al., 1995). Because the second problem is a generalization of the first one, it is also unary NP-hard. Following the standard three-field notation introduced by Graham et al. (1979), $F2|prmu|\Sigma C_i$ and $F2|prmu|u\Sigma C_i + vC_{\max}$ will be used throughout this paper to denote the studied problems. In the notation, the first field describes the machine environment, the second field indicates the problem constraints and the last field specifies the objective functions to be optimized. Although there are many previous works on the studied problems, we shall not introduce each of them in detail. We just describe the research on the ACO algorithm for the permutation flowshop problem. For more related works on this problem, the reader is referred to Dudek et al. (1992), Reisman et al. (1997) and Stützle (1998a).

As for ACO applications in flowshop scheduling, Stützle (1998a) applied the max–min ant system (MMAS), which was proposed by Stützle and Hoos (1997), to the flowshop scheduling problem for minimizing the makespan. T'kindt et al. (2002) proposed an *SACO* heuristic that incorporates simulated annealing algorithm and adjacent pairwise interchange (API) local search to solve the two-machine flowshop scheduling problem of minimizing both the makespan and total completion time. They also used the hybrid strategy in the state transition rule. To minimize total completion time in a no-wait two-machine flowshop, Shyu et al. (2004b) designed some specific features, including a greedy heuristic for initializing the initial pheromone, a hybrid state transition rule and a hybrid local search. Rajendran and Ziegler (2004) proposed two ACO algorithms for dealing with the studied flowshop scheduling problem. The first algorithm, called *M-MMAS*, refers to the idea of the max–min ant system and incorporates the summation rule suggested by Merkle and Middendorf (2003) and a newly proposed local search approach. The second ACO algorithm, called *PACO*, incorporates the concept of relative distance between a given position and the position of a job in the resultant sequence. The algorithms use the NEH heuristic (Nawaz et al.,

1983) to generate an initial sequence for minimizing the makespan and Rajendran's (1993) heuristic to generate an initial sequence for minimizing the total flow time. In their experiments, the proposed ACO algorithms outperform the heuristics constructed by Liu and Reeves (2001), which provides the best heuristic solutions known for the benchmark instances from OR-Library. Ferretti et al. (2006) and Gajpal and Rajendran (2006) are recent ACO applications in scheduling research.

4. New features for flowshop scheduling

The main idea for developing new features for the ACO will be addressed first in this section. In the remaining part of this section, we focus on the design details of new ACO features for the $F2|prmu|\Sigma C_i$ and $F2|prmu|u\Sigma C_i + vC_{\max}$ problems.

4.1. New features of the ACO

For several decades, researchers have studied many interesting applications using meta-heuristic algorithms, among which the ACO has been shown to be capable of producing quality approximate solutions. In ACO applications to scheduling problems, Colorni et al. (1994) proposed an ant system for job shop scheduling. After this paper was published, many researchers proposed different ant-based approaches for such scheduling problems as bus driver scheduling (Forsyth and Wren, 1997), flowshop scheduling (Stützle, 1998a; T'kindt et al., 2002; Shyu et al., 2004b; Rajendran and Ziegler, 2004) and total tardiness problems (Bauer et al., 1999; Merkle and Middendorf, 2003; Besten et al., 2000). The main idea of the above studies is to design specific evaluation functions, state transition rules and/or pheromone updating rules for different problem settings. Some of the above-mentioned papers proposed to use local search methods to reach an earlier convergence state (Dorigo and Gambardella, 1997). In this paper, we undertake a different way of thinking for designing ACO algorithms. In addition to defining new evaluation functions and pheromone updating rules, we introduce two features or components to enrich the ACO algorithm.

4.1.1. New type of pheromone

In the real world, ants can leave and sense at least five different types of pheromone for indirect communication. Besides for trail marking, pheromones play roles in aggregating group work, alarming for crisis, courtship, attacking the enemy and finding good places to build a nest (Hölldobler and Wilson, 1994). The natural phenomenon has inspired us to take into account a new type of pheromone for the ACO algorithm. Such a feature will not only reflect the real-world situations but also have potentials for providing more informative communication in the ant colony and thus solve combinatorial optimization problems better. The traditional ACO algorithm utilizes only pheromone τ_{ij} to dictate the quality of the edge from node i to node j. In our approach, we make use of a new type of pheromone into the state transition rule. Assume that ant k is currently at node i that dictates the job assigned to position $(l-1)$ in a schedule. Ant k needs to determine a node as the immediate successor of job i, which will occupy position l in the schedule. We use new pheromone ε_{jl} to indicate the preference for assigning job j to position l as the immediate successor of job i. To examine each node j to the visit next, we are concerned about not only the cost from the current node to it but also the relative position it would be assigned to in the schedule. Incorporating the new pheromone, we reshape the preference function as the following:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}(\varepsilon_{jl})^{\gamma}}{\sum_{h \notin tabu_k}(\tau_{ih})^{\alpha}(\eta_{ih})^{\beta}(\varepsilon_{hl})^{\gamma}}, & \text{if } j \notin tabu_k, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $tabu_k$ is the set of nodes that are either already visited or unreachable from node i.

The basic idea behind our design is due to the behavior of real ants as well as practical applications. Standard pheromone τ_{ij} used in the ACO reflects the preference of the edge from node i to node j. In some application problems, the actual position of a job in a sequence may be crucial. Such a pheromone is especially useful, even crucial, in hard scheduling problems. For example, assume that letting job i precede job j can produce a good outcome when they both appear in the head part of the schedule. But, this does not imply that it is also beneficial to let job i precede job j in the rear part of a schedule. Consider the job set N as shown in Table 1. Let schedule $S = 1\text{–}2\text{–}3$. The total completion time of schedule S is $Z(S) = 58$. Consider another schedule $S' = 2\text{–}1\text{–}3$ with $Z(S') = 60$. Letting job 1 precede job 2 as in schedule S can produce a better solution when they constitute the prefix. Consider schedule $\pi = 3\text{–}1\text{–}2$ with $Z(\pi) = 77$ and

Table 1
Data set N of the flowshop problem

Jobs	1	2	3
p_i	5	7	10
q_i	4	3	12

schedule $\pi' = 3$–2–1 with $Z(\pi') = 76$. The situation, however, implies that it is beneficial to let job 2 precede job 1 at the rear part of a schedule.

The above example tells us that positional information is crucial, too. While conventional pheromone reflects the relative significance of the move from a job to its immediate successor job, the new pheromone dictates the relative positional significance of a job in a specific schedule. That is, the new pheromone will reflect the preference for assigning job j to position l in a schedule. Incorporating positional pheromone will not only reflect the fact that ants communicate via multiple pheromones but also provide useful information to compose better solutions.

4.1.2. Dominance criteria

Dominance criteria are very useful and efficient in pruning the enumeration tree by constraining the search space of a branch-and-bound algorithm. Similarly, dominance rules can also be applied to ACO algorithms such that the ants will not waste time in constructing the solution on a non-promising path. From ants' point of view, dominance rules can be regarded as a special type of pheromone that warns or prohibits the ants from some dangerous sites and/or tours. To the best of our knowledge, only Merkle and Middendorf (2003) have proposed a simple dominance rule, which is called *deterministic property*, in their paper. In this paper, we shall investigate the effectiveness of dominance criteria on different scheduling problems.

4.2. Specific design of the ACO for $F2|prmu|\Sigma C_i$

4.2.1. Initialization

In the literature, there are many different strategies proposed for initializing pheromone at the very beginning. Usually a constant amount of initial pheromone is deposited on each edge. However, this strategy is not informative to the ants. Therefore, we can use a problem-specific heuristic to achieve a better initialization of pheromone intensity. In this paper, we propose a hybrid heuristic to initialize the initial pheromone. This hybrid heuristic first sets a constant amount of initial pheromone on each edge and updates initial pheromone according to the solution found via the API and NAPI local search procedures. The intensities of two pheromones will be initialized using the hybrid heuristic strategy.

4.2.2. State transition rule

We propose a new visibility function to evaluate the costs of edges for state transition. In the total completion time problem, there is a very important intrinsic cumulative property specifying that when a job is positioned in the front part of a schedule, its processing time will be counted several times in the objective evaluation. As a sequel, we apply this cumulative property to design a new visibility function for the state transition decision. The new visibility function η_{ij} is defined as

$$\eta_{ij} = \frac{\sum_{r=1}^{i} C_r}{\sum_{r=1}^{i} C_r + (n-\mu+1)b_j}, \tag{2}$$

where variable μ represents the total number of already scheduled jobs in the partial schedule. The numerator reflects the cost already incurred. The second part of the denominator is an estimate of the total completion time of the unscheduled jobs. A job producing a larger total completion time of the unscheduled job will have to be assigned a smaller visibility value. When problem size n becomes large, setting the numerator as 1 will not affect the value of η_{ij}. This is the reason why we set the numerator as the value of total completion time of the jobs already scheduled. The proposed new visibility function can provide a certain degree of guidance for the ants to improve the quality of solutions.

We use the hybrid selection strategy and the new type of pheromone for the preference definition of choosing the next node to move on in our ACO algorithm. For ant k on node i, which is the $(l-1)$st job in the partial schedule, the preference for selecting job j as the next node is determined by the following state transition function:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}(\varepsilon_{jl})^{\gamma}}{\sum_{h \notin tabu_k}(\tau_{ih})^{\alpha}(\eta_{ih})^{\beta}(\varepsilon_{hl})^{\gamma}}, & \text{if } j \notin tabu_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

4.2.3. Pheromone updating rule

Colorni et al. (1991) proposed the elitist strategy that resembles the elitist approach in genetic algorithm. At the end of every iteration, the pheromone

laid on edges is only reinforced by the elitist ants, which construct the best-found tours. The trail of best tours will probabilistically attract all the other ants towards the edges belonging to these tours. Because the problem sizes will be large in our experiments, we need to consider the normalization issue. Therefore, we design a specific pheromone updating rule for large-scale problems as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{4}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{\sum_{s=1}^{m} Z(S_s)}{n\sum_{r=1}^{l} C_r}, & \text{if ant } k \text{ traversed edge } (i,j), \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $\sum_{s=1}^{m} Z(S_s)$ is the summation of total completion times over all ants. The new type of pheromone mentioned in the above section is also updated by the same way as in Eqs. (4) and (5):

$$\varepsilon_{jl} = (1-\rho)\varepsilon_{jl} + \sum_{k=1}^{m} \Delta\varepsilon_{jl}^{k}, \tag{6}$$

$$\Delta\varepsilon_{ij}^{k} = \begin{cases} \dfrac{\sum_{s=1}^{m} Z(S_s)}{n\sum_{r=1}^{l} C_r}, & \text{if ant } k \text{ placed job } j \text{ at position } l, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Dorigo and Gambardella (1997) deployed a local pheromone updating policy for the TSP and Besten et al. (2000) also proposed a local pheromone updating rule for the total tardiness problem. Although the change of pheromone intensity over time is a fact of the natural behavior of real ants, embedding the local update rule, however, did not provide significant improvements in our preliminary experiments. Therefore, to avoid extra computational load, we use only the global pheromone updating rule in our experimental setting.

4.2.4. Local search method

Dorigo and Gambardella (1997) used a 3-OPT local search approach to improve the quality of solutions to the TSP. In the literature, many researches applied different local search methods because the local search method usually plays a key role in improving the quality of solutions. We use API and NAPI local search methods in our ACO algorithm. In each iteration, all ants complete their tours and then the best one is identified. The best solution is used as the starting point of the local search heuristic. Therefore, local search is invoked once at the end of each iteration.

4.2.5. Stopping criterion

In our experiment setting, the program will terminate when a given maximum number of iterations within which no better solution is encountered.

4.2.6. Dominance criteria

Della Croce et al. (1996, 2002) have proposed four dominance rules for the total completion time problem. Dominance rules can be classified as either static or dynamic. A static rule means that the rule is determined once the input instance is given and remains unchanged along the solution-finding process. If the conditions specifying a rule to be satisfied vary depending on the scenarios of problem status, then such a rule is called dynamic. Activation of dynamic rules incurs extra computing efforts during resolution. We therefore incorporate only static dominance rules for our ACO algorithm because these rules can be easily examined during the running sessions.

Dominance Rule 1. . If there exists an unscheduled job i such that $p_i \leqslant q_i$, $p_i \leqslant p_j$ and $q_i \leqslant q_j$ for each other unscheduled job j, then in at least one optimal sequence job i is placed first among all the unscheduled jobs (Della Croce et al., 2002).

In the rule developed by Cadambi and Sathe (1993), variable μ represents the total number of jobs already scheduled.

Dominance Rule 2. . If $b_i \leqslant b_j$ and $b_i - b_j + \max\{0, a_i - a_j\} + (n-\mu+1)(\min\{a_i, b_j\} - \min\{b_i, b_j\}) \leqslant 0$, then job i precedes job j when they are adjacent in a sequence (Cadambi and Sathe, 1993).

4.3. Specific design of the ACO for $F2|prmu|u\Sigma C_i + vC_{max}$

We shall also apply the specific new features of the ACO to solve the $F2|prmu|u\Sigma C_i + vC_{\max}$ flowshop problem. From the problem definition, we know that the major difference between $F2|prmu|u\Sigma C_i + vC_{\max}$ and $F2|prmu|\Sigma C_i$ problems is that the first problem involves not only the total completion time but also the makespan. Two parameters, u and v, are used to control the relative importance for the bicriteria decision. The specific

strategies with the new features of the ACO for $F2|prmu|u\Sigma C_i + vC_{\max}$ are similar to the strategies for $F2|prmu|\Sigma C_i$. Therefore, we simply highlight the major differences in our ACO application to the $F2|prmu|u\Sigma C_i + vC_{\max}$ problem.

4.3.1. State transition rule

The cumulative property still has evident influences on the solution quality. Therefore, it is incorporated into the definition of the preference value as shown in the following:

$$\eta_{ij} = \frac{u\sum_{r=1}^{i} C_r + v(C_i + b_j)}{u(\sum_{r=1}^{i} C_r + (n-\mu+1)b_j) + v(C_i + b_j)}. \quad (8)$$

Hybrid selection strategy is applied in the state transition rule.

4.3.2. Pheromone updating rule

We design the pheromone updating rule for the $F2|prmu|u\Sigma C_i + vC_{\max}$ problem as

$$\Delta\tau_{ij}^{k} = \begin{cases} u\left(\dfrac{\sum_{i=1}^{m} Z(S_s)}{n\sum_{i=1}^{i} C_r}\right) \\ +v(C_i + b_j), & \text{if ant } k \text{ traversed edge } (i,j), \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

The new type of pheromone mentioned in the above section is updated as follows:

$$\Delta\varepsilon_{jl}^{k} = \begin{cases} \dfrac{\sum_{i=1}^{m} Z(S_s)}{\sum_{r=1}^{i} C_r} \\ +v(C_i + b_j), & \text{if ant } k \text{ placed job } j \text{ to position } l, \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

The elitist strategy is used in the global pheromone updating rule.

5. Computational experiments

In this section, we describe the computational experiments designed to evaluate the effectiveness of the proposed new features of the ACO. The platform of our experiments is a personal computer with a Pentium-4 1.6 GHz CPU and 1 GB RAM. The operating system is Linux Red Hat 7.3. The programs were coded in Java. Following the commonly adopted scheme in flowshop research, we generated job processing times on machines one and two from the interval [1, 100]. The experiments include two parts designed for $F2|prmu|\Sigma C_i$ and $F2|prmu|u\Sigma C_i + vC_{\max}$, respectively. Both of the two experiments used the same setting for the ACO algorithms. We set the number of ants = 10, the number of iterations for stopping criterion = 50, evaporation rate = 0.1, parameters α, β and γ for state transition rule were 2, 4 and 4, respectively. Threshold value λ of 0.7 was set for the hybrid strategy in the state transition stage. The problem size n was set to be 50, 100, 200, 300, 400 or 500. For each problem size n, 10 input instances were randomly generated and the numerical results were averaged through each 10 instances. The above experimental setting, especially the threshold value and parameters α, β and γ, were determined by extensive preliminary experiments that suggested the best setting of these parameters.

We first implemented different strategies of the ACO algorithms for the $F2|prmu|\Sigma C_i$ problem and recorded the average and the maximum objective values. The first concern of our experiments is to study the effectiveness of the proposed visibility function. For convenience, we denote the basic ACO algorithm, which does not include our proposed visibility function and pheromone updating rules, as *Algorithm ACO_Base*. The ACO algorithm that uses the new visibility function and pheromone updating rule is denoted by *Algorithm ACO_New*. The numerical results of *ACO_Base* and *ACO_New* are summarized in Table 2. The first column n denotes the number of jobs. In the column under each algorithm, Mean indicates the average objective value of each 10 test instances, Maximum the maximum objective value in each 10 test instances and Improvement the percentage of relative improvement in objective value between *ACO_Base* and *ACO_New*. In all tables shown in the paper, the entry with a best value in each scenario is highlighted in boldface. The results clearly show that *Algorithm ACO_New* provides better solutions with average improvements of about 4–5%. The comparison suggests that applying the cumulative property to design a new visibility function for the state transition decision in the total completion time problem can provide better solutions.

We move on to examine the effectiveness of the new pheromone and dominance rules. Because the statistics of the above experiments confirm the significance of the new visibility function and pheromone updating rules, we shall use *ACO_New* as the baseline for measuring the potential gains that can be brought forth by the new pheromone

Table 2
Solution values of the new visibility function and pheromone updating rule

N	*ACO_Base*		*ACO_New*		
	Mean	Maximum	Mean	Maximum	Improvement (%)
50	60,513.8	70,685.0	57,514.3	63,942.0	5.2
100	232,922.4	265,777.0	218,105.0	236,951.0	6.8
200	999,351.3	1,035,317.0	946,723.3	1,022,225.0	5.6
300	2,248,663.4	2,392,810.0	2,157,731.6	2,294,407.0	4.2
400	3,999,218.5	4,237,702.0	3,891,694.2	4,060,783.0	2.8
500	6,504,035.2	6,826,198.0	6,208,090.1	6,541,192.0	4.8

Table 3
Solution values of the new features for $F2\|\|\Sigma C_i$

	N	50	100	200	300	400	500
ACO_New	Mean	59,713.8	254,501.2	1,029,271.0	2,317,734.2	4,151,265.3	6,535,600.9
	Maximum	67,857.0	284,304.0	1,086,548.0	2,440,002.0	4,298,046.0	6,615,499.0
ACO_Domi	Mean	57,859.0	239,659.0	986,159.1	2,211,197.6	3,940,477.6	6,180,904.1
	Maximum	63,907.0	257,777.0	1,089,692.0	2,377,897.0	4,098,513.0	6,426,580.0
	Improvement (%)	3.1	5.8	4.2	4.6	5.1	5.4
ACO_Pher	Mean	56,282.1	226,538.9	978,785.4	2,169,474.3	3,940,182.4	6,236,174.9
	Maximum	62,244.0	249,016.0	1,059,486.0	2,384,138.0	4,215,962.0	6,507,733.0
	Improvement (%)	5.7	11.0	4.9	6.4	5.1	4.6
ACO_All	Mean	**55,803.8**	**221,099.7**	**904,918.1**	**2,013,321.4**	**3,676,940.5**	**5,720,991.4**
	Maximum	61,823.0	238,277.0	986,905.0	2,225,316.0	4,090,737.0	6,146,974.0
	Improvement (%)	6.5	13.1	12.1	13.1	11.4	12.5

and the application of dominance rules. In the following context, the columns entitled Improvement in the following tables indicate the relative improvement in percentages over *ACO_New*. Title *Algorithm ACO_Domi* stands for the ACO algorithm that applies new visibility function, new pheromone updating rule and two dominance rules. *Algorithm ACO_Pher* represents the algorithm using the new visibility function, new pheromone updating rule and the new type of pheromone. Finally, *Algorithm ACO_All* denotes the algorithm that incorporates all the proposed new features.

The numerical results of four combinations of the features are shown in Table 3. Algorithms *ACO_Domi* and *ACO_Pher* both outperform *ACO_New*. For the instances with 100 jobs, the new pheromone provides an impressive improvement of 11%. We also find that *ACO_Pher*, which has an average percentage of improvement of 6.3%, performs better than *ACO_Domi*, whose average percentage of improvement is about 4.7%. Another interesting observation from Fig. 1 is that when the problem size becomes large, the performance of dominance rules becomes better than that of *ACO_Pher*. The reason behind this observation could be that in large-scale problems more dominance relations among the jobs are satisfied and thus a larger portion of infeasible paths were identified and eliminated.

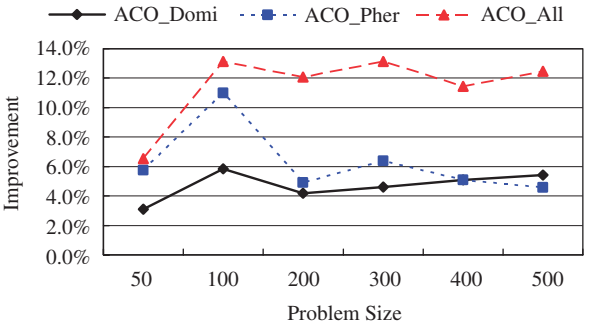

Fig. 1. Comparisons on improvement ratios of new features for $F2\|\|\Sigma C_i$.

We further examine the situation when the new pheromone and dominance rules are simultaneously deployed. *Algorithm ACO_All* presents significant synergy of the proposed features. The improvement percentages are from 6.5% to 13.1%; especially, the improvement percentages are over 11% for large-scale instances.

The second part of the computational experiments is dedicated to the study on the bicriteria $F2|prmu|u\Sigma C_i + vC_{\max}$ problem. The experimental settings for the total completion time problem are similarly applied except that we need to consider another parameter u for tuning the weights between total completion time and makespan. In the experiments, u is 0.2, 0.5 or 0.8, and accordingly v is 0.8, 0.5 or 0.2. Setting $u = 1$ means solving the $F2|prmu|\Sigma C_i$ problem, and setting $u = 0$ indicates the sole focus on $F2\|C_{\max}$. Therefore, the settings with $u = 0$ or 1 were not tested. The computational results are shown in Table 4. Figs. 2–4 illustrate the relative improvement ratios of three algorithms for different values of u. We found that *ACO_Pher* is still superior to *ACO_Domi* when solving the bicriteria problem. *Algorithm ACO_All* achieves synergy effects with an average percentage of improvement from 6.5% to 11.4%. The performance is not as impressive as that for the $F2|prmu|\Sigma C_i$ problem. The phenomenon might be attributed to the fact that when solving the bicriteria problem the dominance rules do not make significant contributions to the reduction of solution space and they even deteriorate the performance of *ACO_All*.

By and large, numerical results showed that the proposed new features of the ACO algorithms

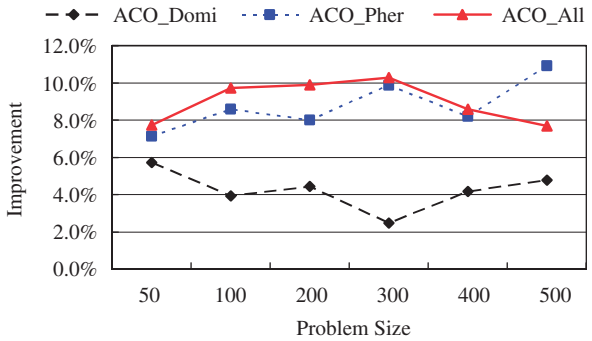

Fig. 2. Comparisons on improvement ratios of new features for $u = 0.2$.

Table 4
Evaluation results of the new features for $F2\|u\Sigma C_i + vC_{\max}$

N	u	*ACO_New*		*ACO_Domi*			*ACO_Pher*			*ACO_All*		
		Mean	Maximum	Mean	Maximum	Impr. (%)	Mean	Maximum	Impr. (%)	Mean	Maximum	Impr. (%)
50	0.2	14,254.6	16,694.0	13,440.6	15,360.4	5.7	13,239.8	14,460.2	7.1	**13,152.4**	14,511.0	7.7
	0.5	31,387.7	36,678.0	30,089.2	35,235.0	4.1	**29,050.2**	32,417.5	7.4	29,200.6	33,831.5	7.0
	0.8	47,978.8	56,004.6	45,948.5	52,250.4	4.2	45,122.2	50,982.6	6.0	**44,855.3**	53,202.6	6.5
100	0.2	52,952.2	59,415.4	50,871.0	54,944.6	3.9	48,410.7	52,511.4	8.6	**47,800.9**	53,740.6	9.7
	0.5	126,207.8	140,259.5	118,724.2	130,186.0	5.9	115,231.7	134,889.5	8.7	**111,844.4**	124,593.0	11.4
	0.8	202,073.7	222,775.0	190,296.9	216,061.8	5.8	**177,508.9**	198,901.6	12.2	179,091.5	205,165.8	11.4
200	0.2	207,346.0	228,321.6	198,135.4	216,057.6	4.4	190,774.7	212,617.8	8.0	**186,852.6**	210,379.4	9.9
	0.5	509,612.7	571,542.5	475,183.7	531,064.0	6.8	464,272.7	521,362.5	8.9	**455,213.3**	509,533.5	10.7
	0.8	810,129.5	877,943.2	770,392.5	874,396.6	4.9	737,010.8	786,062.8	9.0	**718,861.4**	795,115.6	11.3
300	0.2	459,967.5	477,137.2	448,583.8	463,441.4	2.5	414,533.6	458,266.4	9.9	**412,681.0**	455,352.4	10.3
	0.5	1,140,693.3	1,179,663.5	1,095,209.3	1,151,222.5	4.0	**1,018,818.8**	1,164,491.5	10.7	1,043,902.3	1,160,943.0	8.5
	0.8	1,837,173.8	1,969,210.8	1,753,725.4	1,871,998.8	4.5	1,662,491.7	1,805,508.8	9.5	**1,634,905.3**	1,831,020.0	11.0
400	0.2	829,450.5	875,766.8	794,849.5	848,701.8	4.2	761,411.5	826,885.4	8.2	**758,206.0**	820,171.4	8.6
	0.5	2,066,884.1	2,159,000.0	1,968,102.9	2,136,205.0	4.8	**1,877,515.5**	2,093,691.0	9.2	1,879,695.8	2,043,478.0	9.1
	0.8	3,306,375.0	3,451,038.8	3,154,700.2	3,322,084.8	4.6	3,129,769.4	3,369,178.4	5.3	**3,066,076.2**	3,250,935.2	7.3
500	0.2	1,309,920.4	1,389,081.6	1,247,351.0	1,328,280.4	4.8	**1,167,022.9**	1,286,109.4	10.9	1,209,202.7	1,262,423.0	7.7
	0.5	3,195,009.3	3,314,018.5	3,104,667.3	3,222,830.0	2.8	2,975,883.1	3,427,605.5	6.9	**2,886,682.2**	3,165,256.5	9.7
	0.8	5,164,165.9	5,333,183.2	4,972,551.0	5,274,530.0	3.7	**4,794,820.8**	5,111,511.8	7.2	4,823,266.2	5,121,649.0	6.6

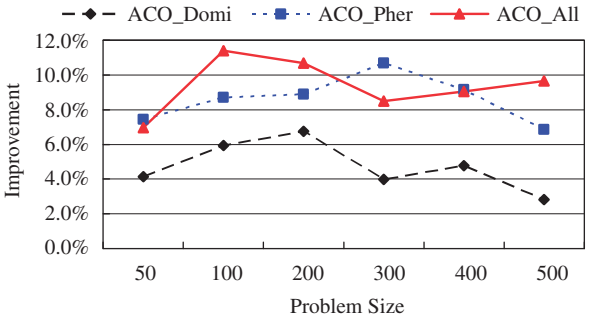

Fig. 3. Comparisons on improvement ratios of new features for $u = 0.5$.

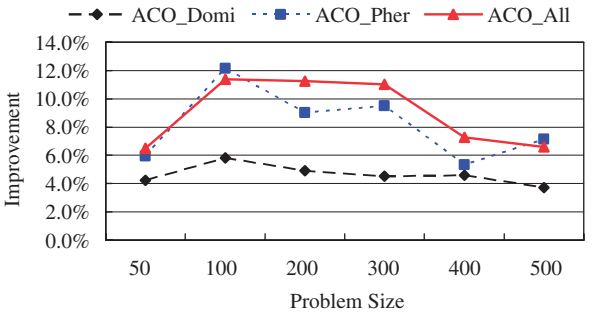

Fig. 4. Comparisons on improvement ratios of new features for $u = 0.8$.

outperform the conventional ones: the average percentage of improvement of *ACO_Domi*, *ACO_Pher* and *ACO_All* are 4.5%, 8.5% and 9.1%, respectively. We may conclude that in the studied sequencing problems, the new type of pheromone conveys positional information for the ant colony to conduct precise and informative communication and construct quality tours.

To better assess the performance of the new features, we conducted further experiments[1] to compare the proposed algorithm *ACO_ALL* with *SACO* (T'kindt et al., 2002), *M-MMAS* and *PACO* (Rajendran and Ziegler, 2004). All algorithms deployed the same population of 10 ants. Numerical results of solution quality and run time for the $F2|prmu|C_{\max}$ problem are summarized in Tables 5 and 6, respectively. It is clear that *PACO* and *M-MMAS* produce better solutions than *ACO_ALL* and *SACO*. Nevertheless, *ACO_ALL* and *SACO* take less time. The long execution time of *PACO* and *M-MMAS* could be attributed to the way the local search heuristic was invoked. In the two algorithms, each ant in each iteration triggers local search when it completes a schedule. Frequent deployment of the local search heuristic has a better opportunity to reach a good solution at the cost of a longer execution time. Comparing *ACO_ALL* and *SACO*, the former can find better schedules. *ACO_ALL* takes a longer time for small-size instances. The run time required by *SACO* becomes longer when 300 or more jobs are considered. The randomness of simulated annealing might have delayed the convergence process of *SACO* when there are more jobs. The results for the $F2|prmu|u\Sigma C_i + vC_{\max}$ problem are given in Tables 7 and 8. Similarly, *PACO* demonstrates absolute superiority concerning solution quality. On the other hand, for all test scenarios *ACO_ALL* takes the shortest execution time. The experiments provide observations on the possible tradeoffs between solution quality and run time when different algorithms are considered.

The investigated ACO algorithms are hybridizations of ACO and local search heuristics. The approaches proposed in this paper does not emphasize too much on local search, but focus on the intrinsic nature of ACO and real ants. Real ants use multiple types of pheromones for communication, whereas sequencing or scheduling problems require extra positional information to produce better solutions. The natural phenomenon of multiple pheromones paves the way to the design of a new mechanism for acquiring positional information in sequencing problems. Our design might suggest a new alternative or way of thinking for the development of ACO algorithms.

6. Concluding remarks and future research

In this paper, we developed new ingredients to reshape the ACO algorithms. The first new feature is a new type of pheromone that will dictate the preference for assigning a job to a specific position in a schedule so as to provide more informative communication in the ant colony and thus better solve hard combinatorial optimization problems. This pheromone design could be useful specifically for sequencing problems. Moreover, such an idea might motivate more alternatives of ACO design. The second new feature is about dominance rules that can curtail unnecessary paths so that the ants will not waste time in traversing infeasible or

[1]The experiments were conducted during the revision of this paper. Therefore, a modern computer (AMD Athlon MP 2200+CPU, 1 GB MB RAM, 20 GB HD, Linux Red Hat 8.0) was used as the platform.

Table 5
Solution values of $F2\|\sum C_i$ with different ACO algorithms

	N	50	100	200	300	400	500
ACO_ALL	Mean	56,705	221,955	958,059	2,151,480	3,757,637	5,875,092
	Maximum	64,317	269,829	1,085,141	2,474,699	4,253,432	6,745,419
M-MMAS	Mean	54,806	204,587	810,570	1,805,098	3,145,376	4,891,381
	Maximum	62,375	231,740	864,458	1,850,812	3,260,178	5,058,216
PACO	Mean	**54,777**	**204,342**	**809,641**	**1,803,170**	**3,142,351**	**4,887,836**
	Maximum	62,387	231,668	864,161	1,848,439	3,257,317	5,058,594
SACO	Mean	62,067	245,936	997,317	2,256,341	3,968,490	6,185,820
	Maximum	67,477	274,681	1062,690	2,306,964	4,109,679	6,288,014

Table 6
Average elapsed run time (s) of different ACO algorithms for $F2\|\sum C_i$

N	50	100	200	300	400	500
ACO_All	4.01	19.26	75.98	**379.23**	**542.84**	**1236.21**
M-MMAS	26.75	197.45	1079.98	2943.78	7895.45	14,728.82
PACO	29.69	170.49	920.39	3471.82	7597.72	17,014.64
SACO	**2.21**	**16.97**	**135.02**	450.90	1081.48	2175.18

Table 7
Solution values of $F2\|u\sum C_i + vC_{\max}$ with different ACO algorithms

N	u	*ACO_All*		*M-MMAS*		*PACO*		*SACO*	
		Mean	Maximum	Mean	Maximum	Mean	Maximum	Mean	Maximum
50	0.2	13,633	15,394	13,145	14,882	**13,141**	14,860	14,369	16,588
	0.5	29,946	33,885	28,771	32,657	**28,760**	32,658	31,772	36,648
	0.8	45,974	52,554	**44,380**	50,484	44,384	50,452	48,808	55,856
100	0.2	47,149	53,028	45,127	50,990	**45,118**	51,009	52,161	59,889
	0.5	110,760	126,613	104,902	118,775	**104,835**	118,748	122,639	137,819
	0.8	172,318	196,463	164,673	186,480	**164,622**	186,549	193,462	222,696
200	0.2	181,131	200,101	170,433	181,725	**170,261**	181,555	202,340	218,462
	0.5	440,257	488,901	410,417	437,585	**410,167**	437,585	493,591	514,883
	0.8	690,388	751,543	650,434	692,388	**649,702**	692,835	788,634	833,339
300	0.2	404,377	415,108	373,396	382,357	**372,936**	381,818	453,836	463,614
	0.5	978,873	1,014,845	910,228	932,899	**909,293**	931,824	1,107,614	1,125,861
	0.8	1,551,631	1,588,144	1,446,758	1,482,423	**1,445,645**	1,481,715	1,768,030	1,802,767
400	0.2	695,840	724,665	645,322	668,810	**644,915**	668,402	788,158	825,449
	0.5	1,719,186	1,756,559	1,582,597	1,640,632	**1,581,679**	1,638,617	1,937,232	2,057,875
	0.8	2,713,958	2,805,259	2,520,192	2,611,355	**2,519,041**	2,608,255	3,084,945	3,272,494
500	0.2	1,089,426	1,154,058	998,578	1,032,278	**998,191**	1,031,777	1,230,456	1,270,146
	0.5	2,690,311	2,812,971	2,458,413	2,542,692	**2,456,581**	2,542,205	3,032,094	3,141,046
	0.8	4,310,216	4,584,747	3,917,877	4,052,012	**3,915,832**	4,052,039	4,862,816	5,021,643

Table 8
Average elapsed run time (s) of different ACO algorithms for $F2\|u\sum C_i + vC_{\max}$

N	u	ACO_*All*	*M-MMAS*	*PACO*	*SACO*
50	0.2	**2.35**	22.27	28.38	13.09
	0.5	**2.51**	24.46	29.64	13.10
	0.8	**2.79**	27.01	29.55	13.09
100	0.2	**17.56**	149.49	172.14	114.46
	0.5	**16.57**	164.77	145.49	114.42
	0.8	**17.68**	214.04	163.49	114.42
200	0.2	**82.77**	1177.37	1051.71	1005.68
	0.5	**84.15**	1129.27	1079.69	1005.56
	0.8	**80.32**	1119.80	992.42	1005.64
300	0.2	**277.27**	2862.75	3766.53	3610.07
	0.5	**304.87**	3575.73	3351.74	3612.16
	0.8	**301.14**	3379.26	3,911.68	3613.72
400	0.2	**666.40**	9145.58	7699.58	8014.59
	0.5	**538.52**	8047.84	7974.58	8018.11
	0.8	**619.58**	7364.61	8093.64	8016.87
500	0.2	**1516.34**	17,303.44	15,013.90	17,953.41
	0.5	**1181.76**	16,225.55	16,125.87	17,953.80
	0.8	**1144.13**	16,445.94	16,340.88	17,935.69

non-promising paths during their food-seeking process. The proposed new features of the ACO algorithm were applied to two flowshop scheduling problems. Due to the cumulative property of total completion time, we also developed a new visibility function. Computational results showed that the proposed new strategies for state transition and pheromone update are very effective in comparison with the generic ACO algorithm. Furthermore, the results also evinced the significance of new pheromone and dominance rules. Further experiments were also done to compare the proposed approach with the three existing algorithms.

Although many applications of the ACO have been proposed to deal with complex sequencing problems thus far, there are still many interesting research directions. For example, considering other sequencing problems to test and verify the effectiveness and robustness of the proposed new features of ACO is a worthy topic. And, ant algorithms are well suited for parallelization but not much relevant research has been done. To the best of our knowledge, only a few papers on parallel implementations of ACO algorithms (Bullnheimer et al., 1998; Stützle, 1998b; Talbi et al., 1999; Middendorf et al., 2002; Randall and Lewis, 2002) have been reported. It is interesting to examine whether our proposed new features of ACO are effective for parallelization or not. It could be interesting to create another new type of pheromone for parallelization. Finally, we may regard the behavior of artificial ants cooperating with one another to exchange their information and to share their knowledge as a group decision and learning process. Therefore, another direction of potential interest could be the application of the ACO in e-learning and knowledge management.

Acknowledgments

The authors are grateful to the anonymous referees who provided suggestions that have improved the presentation of this paper.

References

Bauer, A., Bullnnherimer, B., Hartl, R.F., Strauss, C., 1999. An ant colony optimization approach for the single machine total tardiness problem. In: Proceedings of the 1999 Congress on Evolutionary Computation. Washington, DC, USA, pp. 1445–1450.

Beckers, R., Deneubourg, J.L., Goss, S., 1992. Trails and U-turns in the selection of the shortest path by the ant *Lasius niger*. Journal of Theoretical Biology 159, 397–415.

den Besten, M., Stützle, T., Dorigo, M., 2000. Ant colony optimization for the total weighted tardiness problem. In: Proceedings of Parallel Problem Solving from Nature Conference. Also available as Tech.Rep.IRIDIA/99-16, Université Libre de Bruxelles, Belgium.

Brucker, P., 1997. Scheduling Algorithms. Springer, Berlin.

Bullnheimer, B., Kotsis, G., Strauβ, C., 1998. Parallelization strategies for the ant system. High Performance Algorithms and Software in Nonlinear Optimization; Series: Applied Optimization, vol. 24, Kluwer, Dordrecht, pp. 87–100.

Bullnheimer, B., Hartl, R.F., Strauss, C., 1999a. Applying the Ant System to the Vehicle Routing Problem. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston.

Bullnheimer, B., Hartl, R.F., Strauss, C., 1999b. An improved ant system algorithm for the vehicle routing problem. In: The Sixth Viennese Workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna, Austria, May 21–23.

Cadambi, B.W., Sathe, Y.S., 1993. Two-machine flowshop scheduling to minimize mean flow time. Operations Research 30 (1), 35–41.

Colorni, A., Dorigo, M., Maniezzo, V., 1991. The ant system: An autocatatlytic process. Technical Report No. 91-016, Politecnico di Milano, Italy.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34 (1), 39–53.

Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., Trubian, M., 1996. Heuristics from nature for hard combinatorial problems. International Transactions in Operational Research 3 (1), 1–21.

Della Croce, F., Narayan, V., Tadei, R., 1996. The two-machine total completion time flow shop problem. European Journal of Operational Research 90 (2), 227–237.

Della Croce, F., Ghirardi, M., Tadei, R., 2002. An improved branch-and-bound algorithm for the two-machine total completion time flow shop problem. European Journal of Operational Research 139 (2), 293–301.

Di Caro, G., Dorigo, M., 1998. AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365.

Dorigo, M., 1992. Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computing 1, 53–66.

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.

Dorigo, M., Di Caro, G., Gambardella, L.M., 1999. Ant algorithms for discrete optimization. Artificial Life 5 (3), 137–172.

Dudek, R.A., Panwalkar, S.S., Smith, M.L., 1992. The lessons of flowshop scheduling research. Operations Research 40, 7–13.

Ferretti, I., Zanoni, S., Zavanella, L., 2006. Production–inventory scheduling using ant system metaheuristic. International Journal of Production Economics 104 (2), 317–326.

Forsyth, P., Wren, A., 1997. An ant system for bus driver scheduling. In: The Seventhth International Workshop on Computer-Aided Scheduling of Public Transport, Boston, August 1997.

Gajpal, Y., Rajendran, C., 2006. An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. International Journal of Production Economics 101 (2), 259–272.

Gambardella, L.M., Taillard, E., Dorigo, M., 1999. Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 50, 167–176.

Garey, M.R., Johnson, D.S., Sethi, R., 1979. The complexity of flowshop and job shop scheduling. Mathematics of Operations Research 1, 117–129.

Glover, F., 1990. Tabu search: A tutorial. Interfaces 20, 74–94.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Learning. Addison Wesley, New York, NY.

Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M., 1989. Self-organized shortcuts in the argentine ant. Naturwissenschaften 76, 579–581.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326.

Holland, J.H., 1975. Adaptation in Natural and Artificial System. The University of Michigan, Ann Arbor.

Hölldobler, B., Wilson, E.O., 1994. Journey to the Ants: A Story of Scientific Exploration. Harvard University Press, USA.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–67.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Kuntz, P., Snyers, D., 1994. Emergent colonization and graph partitioning. In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3. MIT Press, Cambridge, MA.

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 (2), 439–452.

Maniezzo, V., Colorni, A., Dorigo, M., 1994. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium.

Merkle, D., Middendorf, M., 2003. An ant algorithm with a new pheromone evaluation rule for total tardiness problems. Applied Intelligence 18 (1), 105–111.

Middendorf, M., Reischle, F., Schmeck, H., 2002. Multi colony ant algorithms. Journal of Heuristics 8 (3), 305–320.

Nagar, A., Haddock, J., Heragu, S.S., 1995. Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research 81 (1), 88–104.

Nawaz, M., Enscore Jr., E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA 11 (1), 91–95.

Pinedo, M., 2002. Scheduling: Theory, Algorithms, and Systems, second ed. Prentice-Hall, New York, NY.

Ramos, V., Almeida, F., 2000. Artificial ant colonies in digital image habitats—a mass behavior effect study on pattern recognition. In: Proceedings of ANTS'2000—Second International Workshop on Ant Algorithms, Brussels, Belgium, pp. 113–116.

Rajendran, C., 1993. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. International Journal of Production Economics 29 (1), 65–73.

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 (2), 426–438.

Randall, M., Lewis, A., 2002. A parallel implementation of ant colony optimization. Journal of Parallel and Distributed Computing 62 (9), 1421–1432.

Reisman, R.A., Kumar, A., Motwani, J., 1997. Flowshop scheduling/sequencing research: A statistical review of the literature. 1952–1994. IEEE Transactions on Engineering Management 44 (3), 316–329.

Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L., 1997. Ant-based load balancing in telecommunications networks. Adaptive Behavior 5 (2), 169–207.

Shyu, S.J., Yin, P.Y., Lin, B.M.T., Haouari, A., 2003. Ant-Tree: An ant colony optimization approach to the generalized minimum spanning tree problem. Journal of Experimental and Theoretical Artificial Intelligence 15, 103–112.

Shyu, S.J., Lin, B.M.T., Yin, P.Y., 2004a. An application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total complete time. Computers & Industrial Engineering 47, 181–193.

Shyu, S.J., Lin, B.M.T., Hsiao, T.S., 2004b. Ant colony optimization algorithm for the cell assignment problem in PCS networks. Computers & Operations Research 33 (6), 1713–1740.

Shyu, S.J., Yin, P.Y., Lin, B.M.T., 2004c. An ant colony optimization algorithm for the minimum weight vertex cover problem. Annals of Operations Research 131, 283–304.

Stützle, T., 1998a. An ant approach to the flow shop problem. In: Proceedings of EUFIT'98, Aachen, pp. 1560–1564.

Stützle, T., 1998b. Parallelization strategies for ant colony optimization. In: Proceedings of Parallel Problem Solving from Nature, vol. 1498. Springer, Berlin, pp. 722–741.

Stützle, T., Dorigo, M., 1999a. ACO Algorithms for the Traveling Salesman Problem. Evolutionary Algorithms in Engineering and Computer Science. Wiley, NY.

Stützle, T., Dorigo, M., 1999b. ACO Algorithms for the Quadratic Assignment Problem. New Ideas in Optimization. McGraw-Hill, New York, NY.

Stützle, T., Hoos, H., 1997. The max–min ant system and local search for the traveling salesman problem. In: Proceedings of ICEC'97, pp. 309–314.

Talbi, E., Roux, O., Fonlupt, C., Robillard, D., 1999. Parallel ant colonies for combinatorial optimization problems. In: Proceedings of Parallel and Distributed Processing, vol. 1586. Springer, Berlin, pp. 239–247.

T'kindt, V., Monmarché, N., Tercinet, F., Laügt, D., 2002. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. European Journal of Operational Research 142 (2), 250–257.