J Intell Manuf
DOI 10.1007/s10845-014-1023-3

# Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning

**S. Zhang · T. N. Wong**

Received: 19 June 2014 / Accepted: 9 December 2014


**Abstract** This study develops an enhanced ant colony optimization (E-ACO) meta-heuristic to accomplish the integrated process planning and scheduling (IPPS) problem in the job-shop environment. The IPPS problem is represented by AND/OR graphs to implement the search-based algorithm, which aims at obtaining effective and near-optimal solutions in terms of makespan, job flow time and computation time taken. In accordance with the characteristics of the IPPS problem, the mechanism of ACO algorithm has been enhanced with several modifications, including quantification of convergence level, introduction of node-based pheromone, earliest finishing time-based strategy of determining the heuristic desirability, and oriented elitist pheromone deposit strategy. Using test cases with comprehensive consideration of manufacturing flexibilities, experiments are conducted to evaluate the approach, and to study the effects of algorithm parameters, with a general guideline for ACO parameter tuning for IPPS problems provided. The results show that with the specific modifications made on ACO algorithm, it is able to generate encouraging performance which outperforms many other meta-heuristics.

**Keywords** Integrated process planning and scheduling · Ant colony optimization · Algorithm parameter tuning

## Introduction

Over the past decades, researchers have attempted various approaches to integrate the manufacturing process planning and scheduling functions. In today's manufacturing environment, process planning and scheduling are still performed separately. That is, the process plan has to be generated first, scheduling will then allocate resources in accordance with the process plan constraints. In real-time manufacturing, the shopfloor may encounter many kinds of uncertainties—machine breakdowns, order cancel, design changes, rush jobs, etc. The pre-defined process plans and schedules may then become deficient or infeasible due to the dynamic changes. Integrated process planning and scheduling (IPPS) is to combine both the process planning and scheduling problems in the consideration, that is, the actual process plan and the schedule are determined dynamically in accordance with the order details and the status of the manufacturing system. The merit of IPPS is to eliminate scheduling conflicts, reduce flow-time and work-in-process, improve production resources utilization and adapt to irregular shop floor disturbances (Lee and Kim 2001), thus the feasibility and optimality of both process and schedule will be significantly improved. However, the integration of these two manufacturing planning functions is a challenging task. Due to the combination of two complex optimization problems and the flexibilities existing in the manufacturing system, IPPS is a difficult NP-hard problem (Kim et al. 2003; Kis 2003). In most studies on IPPS, process selection and sequencing still need to be predefined; the integrated approach is mainly to handle the alternative routings in consideration of the scheduling constraints (Leung et al. 2010).

This paper presents an enhanced ant colony optimization (E-ACO) algorithm for the IPPS problem. In comparison with the standard ant colony optimization (ACO) algorithm, the proposed E-ACO meta-heuristic has been specifically modified to cater for the characteristics of IPPS problems. It is implemented on a fully distributed multi-agent system (MAS) structure, in which the ants are simulated by software agents running independently and simultaneously, with

S. Zhang · T. N. Wong (✉)
Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Pokfulam, Hong Kong
e-mail: tnwong@hku.hk

Published online: 20 December 2014

a supervisory agent controlling the flow of the algorithm. With respect to the IPPS problem of producing $N$ parts with $M$ machines in job-shop or similar kind of flexible production environment, the proposed model takes into consideration a full set of flexibilities including alternative routings, alternative sequences and alternative machines. The widely used test cases adopted from Kim et al. (2003) are used to evaluate the performance of the algorithm by comparing the results with other approaches in several aspects, as well as to study the ACO algorithm parameters, for their significance and effect on results, providing a general guideline for parameter tuning of ACO application on IPPS problems.

## Related works

Earlier IPPS research was mainly on the applications of classical exact and analytical approaches such as branch and bound algorithm (Iwata et al. 1978) and mixed integer programming (Nasr and Elsayed 1990). These approaches were limited to solving relatively simple IPPS problems only. Due to the highly complex nature of IPPS problems, the required computation time and effort of the analytic approaches increase in a dramatic manner for larger-sized problems. In contrast, heuristics-based approaches have been found more effective and efficient for realistic scale IPPS problems, with the objective to find near-optimal solutions within reasonable time. Examples are dispatching rules (Khoshnevis and Chen 1991), tabu search (TS) (Ko et al. 2001; Seo and Egbelu 1996; Weintraub et al. 1999), genetic algorithm (GA) (Lawrynowicz 2008; Morad and Zalzala 1999; Qiao and Lv 2012; Shao et al. 2009; Ventura and Yoon 2013), simulated annealing (SA) (Palmer 1996), symbolic evolutionary algorithm (SEA) (Kim et al. 2007, 2003), particle swarm optimization (PSO) (Guo et al. 2009; Zhu et al. 2009), hybrid algorithm (HA) (Li et al. 2012, 2010) and ant colony optimization (ACO) (Arnaout et al. 2014, 2010; Korytkowski et al. 2013; Kumar et al. 2003; Leung et al. 2010; Lin et al. 2013; Liu et al. 2013; Wang et al. 2014; Wong et al. 2012).

With the advent of agent technology, many researchers have studied and developed agent-based systems for manufacturing application. Balasubramanian et al. (1996) presented a multi-agent architecture for the integration of design, manufacturing, and shop-floor control activities. Gu et al. (1997) established a bidding-based multi-agent system for IPPS; Usher (2003) proposed a negotiation-based approaching for the routing problem in job-shop with collaborative agents. The authors have also engaged in developing agent-based systems for IPPS (Wong et al. 2006a, b, c). An MAS-based decentralized multi-agent negotiation (MAN) system has been established for solving the IPPS problems, with advantages of flexible system architectures and responsive fault tolerance. The actual process plan and schedule for producing a particular product are determined through negotiation between part agents and machine agents representing parts and machines respectively. In later research, the MAN approach was extended to hybrid-based agent negotiation (HAN) by the addition of a supervisory agent to improve the performance and effectiveness of the negotiation-based approach. Tehrani Nik Nejad et al. (2010) presented a multi-agent architecture for generating suitable process plans based on the status of the flexible manufacturing system.

Ant colony optimization (ACO) meta-heuristic, originally designed by Dorigo (1992), has been widely applied to many combinatorial optimization problems like travelling salesman problem (TSP), quadratic assignment problem (QAP), and job-shop scheduling problem (JSP). The ACO or ant system algorithm is inspired by the capability of a colony of real ants to find the shortest path between their nest and a food source. The path-finding ability refers to the behaviour that foraging ants leave pheromones on the path, which stimulates succeeding ants to follow the pheromone trail. When the ants are given with several possible paths, each ant initially chooses one path randomly, consequentially some ants picking the shorter routes will return faster. So immediately after their completion of one tour, there will be more pheromones on the shortest path, influencing other later ants to follow this path. After some time, this results in the whole colony travelling on the shortest path. A detailed description of the basics of the ACO algorithm, it applications, and its successful variants can be found in Blum (2005). ACO has been widely applied to solve various types of scheduling problems. Merkle et al. (2002), Chiang et al. (2008) and Zhang (2012a) respectively presented ACO approaches for the resource-constrained project scheduling problem. Arnaout et al. (2010) presented a two-stage ACO approach for unrelated parallel machine scheduling problem with sequence dependent setup-time consideration; an extension of their study can be found in Arnaout et al. (2014). Lin et al. (2013) also used an ACO heuristic for unrelated parallel machine scheduling problem. In Tavares Neto et al. (2013), ACO is applied to the parallel machine scheduling problem with outsourcing consideration. Wu et al. (2012) presented the application of ACO on a two-agent single-machine scheduling problem with learning and deteriorating effects. Korytkowski et al. (2013) presented an ACO assigned with multi-attribute dispatching rules for job shop scheduling problem. On another aspect, Liu et al. (2013) applied the ACO for the process planning problem by graphically mapping it into a travelling salesman problem and constructing a corresponding mathematical model. In the IPPS problem domain, however, only a few researches are found. An earlier attempt to use the ACO algorithm was reported in Kumar and Rajotia (2003). However, their work considered only alternative machines in the IPPS problems, while the routing flexibility in parts was neglected. Leung et al. (2010) presented a

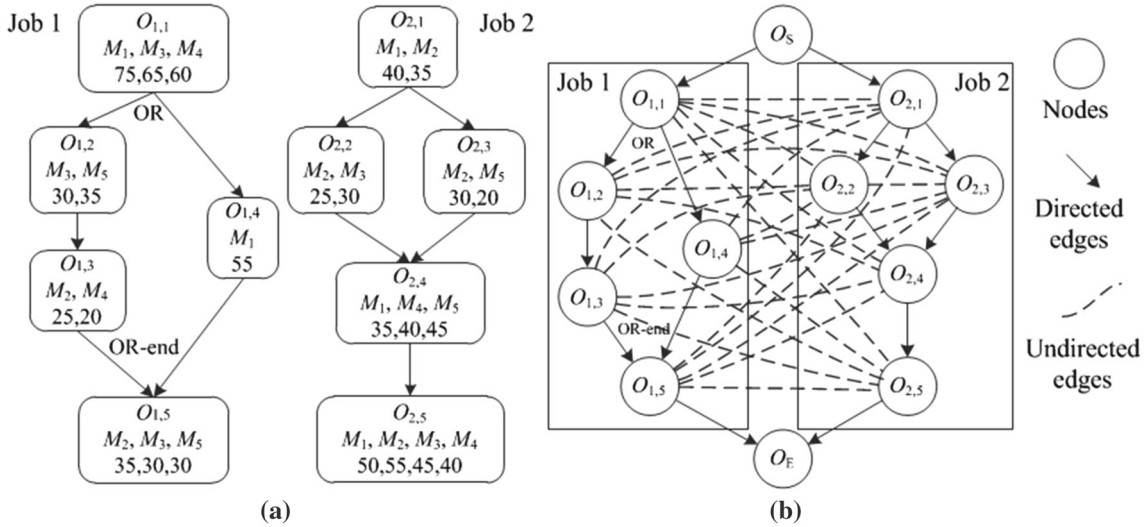


**Fig. 1** A sample two-jobs five-machines IPPS problem. **a** AND/OR graph of process plans of two jobs. **b** Disjunctive graph of IPPS problem

preliminary application of ACO algorithm implemented on MAS to solve IPPS problems with a full set of manufacturing flexibilities. This was a feasibility study and the approach was based on raw ACO heuristic without specific setting to cater for the IPPS problem. The performance was therefore limited as the approach was focused on the sequencing of processes but it did not attempt to scrutinise the selection of alternative processes. Subsequently, the authors proposed a modified ACO algorithm for handling process selection and sequencing in two stages (Wong et al. 2012). The performance of the solutions was improved significantly. However, there was a lack of elitist pheromone deposit strategy in the two-stage ACO, and the dispatching rule was still simply based on shortest processing time. Wang et al. (2014) presented an ACO with strategy to avoid local convergence and stagnation in IPPS problems, with the report that their approach was able to generate good solutions in simulation experiments. However, the process planning and scheduling constraints were not clearly defined in their algorithm and the results were not validated.

In this present paper, the enhanced version of ACO heuristics (E-ACO) is proposed to deal with IPPS problems. It is focused on the algorithm issues of ACO heuristics on IPPS, such as the convergence phenomenon, necessity of introducing node-based pheromone, changing dispatching rule, using elitist strategy, and parameter tuning, etc.

## ACO approach for IPPS

### Problem formulation

Given an $N$-jobs $M$-machines instance, let $\boldsymbol{N}$ denotes the set of jobs such that $\boldsymbol{N} = \{1, 2, \ldots, n, \ldots, N\}$, $\boldsymbol{M}$ denotes the set of machines where $\boldsymbol{M} = \{1, 2, \ldots, m, \ldots, M\}$, the IPPS problem is an optimization problem to select and schedule operations of all $N$ jobs such that precedence constraints between the operations are satisfied. The objective is to optimize the performance criterion, most typically, to minimize the overall makespan.

To begin with, alternative process plans of each job are visualized into AND/OR graphs (Ho and Moodie 1996). As an example, Fig. 1a presents the AND/OR graphs of the two-jobs five-machines sample problem, where operations are denoted by nodes and precedence constraints between operations are expressed by directed edges. The alternative routes are indicated by "OR" tokens, defined as "OR-subgraphs"; those bifurcations without "OR" tokens are defined as "AND-subgraphs". To apply the ACO heuristics for the IPPS problem, a disjunctive graph $\boldsymbol{D} = (\boldsymbol{O}, \boldsymbol{A}, \boldsymbol{B})$ is established with the addition of a dummy start node $O_S$ and a dummy end node $O_E$ to combine the AND/OR graphs of all jobs together (Leung et al. 2010) (Fig. 1b, the alternative processes are omitted), where $\boldsymbol{O}$ is the set of nodes (circles, representing individual operations), $\boldsymbol{A}$ is the set of directed edges (indicating precedence constraints) and $\boldsymbol{B}$ is the set of undirected edges (dashed curves). Let $O_{n,j}$ denotes the $j$th operation of job $n$, there exist undirected edges between: (i) any two operations belonging to different jobs, e.g. $O_{1,1}$ and $O_{2,1}$; and (ii) any two operations belonging to different branches of an AND subgraph of a job, e.g. $O_{2,2}$ and $O_{2,3}$. Each alternative process of an operation can be uniquely determined by $O_{n,j}(M_m)$ where $m$ is the index of alternative machine.

Typically, the following assumptions are made for most models constructed for IPPS:

- Jobs are independent from each other, no priority is assigned among jobs;

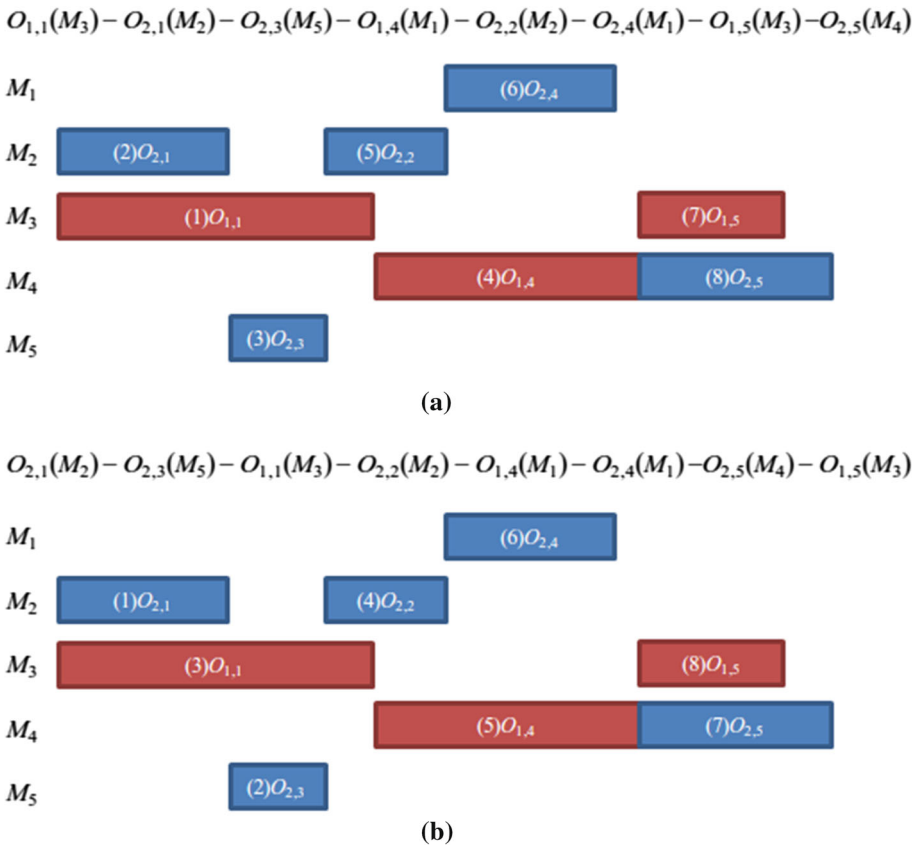


**Fig. 2** Yielding schedules from process series. **a** Yielding detailed schedule (Gantt chart) from a process series. **b** A different series yielding the same schedule

- All jobs are immediately available at the beginning;
- Jobs may have alternative process plans containing different combinations of operations;
- Each operation may have one or more alternative machines;
- Each machine can process only one job at a time;
- Each job can only be processed by at most one machine at any time;
- Processing times are known and deterministic;
- Machines are equipped with the set of required tools.

ACO heuristic

With the construction of the disjunctive graph, the ACO heuristic is then applied to find a route on the graph, from $O_S$ to $O_E$, which corresponds to a feasible solution to the IPPS problem. At the beginning of each iteration, a colony of $K$ ants are assigned at $O_S$, and allowed to travel freely node by node until $O_E$ is reached. During the whole process, both directed edges and undirected edges are feasible paths, as long as the precedence constraints specified by directed edges are followed. An operation may have multiple predecessors and successors. If the relationship among these branch routes is "OR" (indicated by an "OR" token at the beginning, and "OR-end" token at the end), then only one of these branches will be performed. For example, after finishing $O_{1,1}$, an ant can choose to go to either $O_{1,2} \rightarrow O_{1,3}$ or $O_{1,4}$; $O_{1,5}$ can start after either one of $O_{1,3}$ and $O_{1,4}$ is finished. Conversely, all branches are compulsory if the relationship is "AND" (no token). For example, both $O_{2,2}$ and $O_{2,3}$ have to be executed after $O_{2,1}$; and $O_{2,4}$ can start only after both $O_{2,2}$ and $O_{2,3}$ are completed. In the end, a series of processes are identified, they are then one by one assigned to the corresponding machines to yield a certain sequence, which is the scheduling solution suggested by the ants. Figure 2 gives an example of how a series of processes is transformed into a detailed machining schedule for the sample problem.

During the traversing, ants are guided by two factors: pheromone trail $\tau$ and a heuristic desirability $\eta$. At each step, an ant chooses the next node from all possible destinations by a probabilistic procedure. Let $u$, $v$ denote the current and destination nodes respectively, the probability that ant moves from $u$ to $v$ is:

$$p(u,v) = \begin{cases} \frac{[\tau(u,v)]^{\alpha}[\eta(u,v)]^{\beta}}{\sum_{w \in S}[\tau(u,w)]^{\alpha}[\eta(u,w)]^{\beta}}, & \text{if} \quad v \in \boldsymbol{S}. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $\alpha$ and $\beta$ are respective weights of pheromone trail and heuristic desirability; $\boldsymbol{S}$ denotes the set of possible destination nodes, which are connected to current node $u$ by either directed edge or undirected edge without violation of precedence constraints.

The pheromone trail plays the key character in ACO heuristics, which reflects the "community intelligence" of the ant colony. The value of pheromone trail on a route represents how it is favoured by the ants passing through. In real life, as ants deposit pheromones on their paths but then pheromone evaporates as time passes, more pheromones will naturally accumulate on shorter routes. Incoming ants will intend to follow these routes with higher pheromone values, since these routes are "suggested" by previous ants as "good" routes.

The heuristic desirability is the artificially added factor representing the "interest" of individual ants, aiming at generating solutions with good fitness. This value directly reflects the destination selection strategy, or "dispatching rule", in the search-based algorithm. The most widely used dispatching rule might be the "greedy" strategy which attempts to choose the locally optimal solution in each iteration. For instance, in a TSP whose objective is to minimize the total travel distance, the greedy strategy is to select the most nearby position at each step, thus the heuristic desirability can be defined as $\eta = C/d$ where $C$ is a positive constant and $d$ is the distance between the two cities. For scheduling problems, makespan is commonly chosen to be the primary criterion. A frequently used dispatching rule for ACO application on scheduling is based on the shortest processing time (SPT), that is, $\eta = C/PT(v, m)$ where $PT(v, m)$ is the processing time to complete the process by machine $m$ on the corresponding node $v$. This dispatching rule significantly affects the effectiveness of the ACO algorithm. In the following section, we will prove that there are other more competitive dispatching rules rather than the most commonly used SPT.

In summary, the general steps of applying ACO on IPPS are listed in Fig. 3.

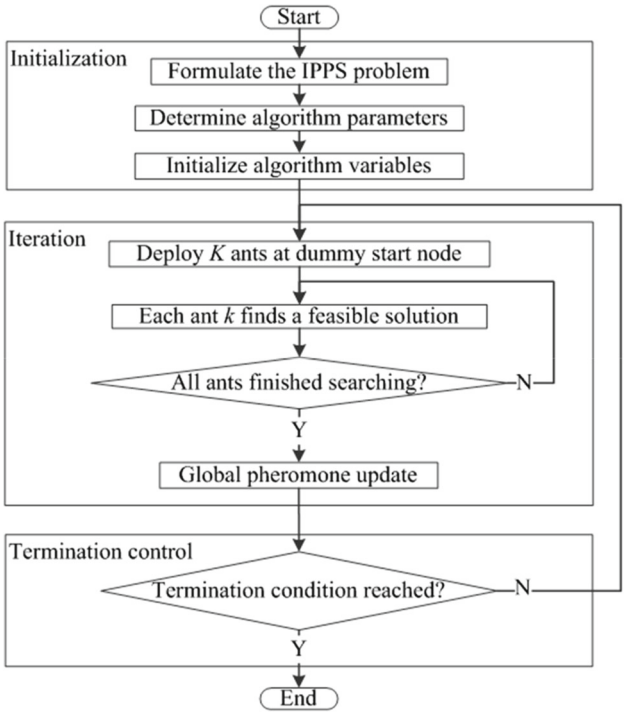


**Fig. 3** Flow chart of applying the ACO heuristics on IPPS

## E-ACO heuristic

In our pilot study to apply standard ACO on IPPS (Leung et al. 2010), the performance was found to be not quite competitive compared with other approaches. This was mainly due to the ineffective SPT dispatching rule and lack of elitist strategy in pheromone deposit. The two-stage ACO (Wong et al. (2012) attempts to improve the performance by introducing node-based pheromone. The dispatching rule remains unchanged, however, still limiting the effectiveness of the heuristic. The following sections discuss how these issues are dealt with in the enhanced ACO heuristic in this paper.

### Quantification of convergence level

Convergence is a commonly encountered phenomenon in most algorithms including ACO, thus, it is often used as a measurement of algorithm effectiveness and the trend of being stable. Theoretically, all effective algorithms should finally converge on the optimal solution. However, over-convergence, i.e. the algorithm continuously sticks to certain sub-optimal solutions and stops to find new solution, is a waste of computation power and should be avoided.

This paper presents a quantified definition to measure the level of convergence of ACO algorithm in scheduling problem:

$$\sigma_i = \begin{cases} 0, & \text{if} \quad i \leq 2. \\ \frac{|N(X_{\text{ib},i-2}) \cap N(X_{\text{ib},i-1})|}{|N(X_{\text{ib},i-1})|}, & \text{else.} \end{cases} \tag{2}$$

where $i$ is the iteration index, $X_{\text{ib},i}$ is the best solution found in iteration $i$, $N(X_{\text{ib},i})$ is the set of processes chosen in the process series of $X_{\text{ib},i}$. It's apparent that $\sigma_i$ is a direct measurement of convergence: the more processes are inherited from the last $X_{\text{ib}}$, the larger $\sigma_i$ will be, indicating the higher level of convergence. It is assigned to 0 at the beginning of algorithm, and updated iteratively after then.

### Introduction of pheromones on nodes

This modification has already been implemented in our preliminary two-stage ACO study (Wong et al. 2012). In ACO heuristics, pheromone is the feedback mechanism of solution fitness, which leads to the progressive improvement by iterations and the final convergence of the algorithm. This is the usual configuration for most ACO applications on other graphically formulated combinatorial optimization

problems. For classical JSP, this is also inherited since all operations are assigned to machines.

However, for IPPS, due to the existence of flexibilities, i.e. alternative process plans, alternative sequences and alternative machines, not all operations have to be processed. A feasible solution to an IPPS is a combination of some chosen operations (instead of all), assigned to chosen machines, in a chosen sequence. In terms of graphical formulation, this means that not all the nodes are to be visited in a feasible route, which is quite different from TSP and JSP. In such case, sequencing is no longer the only issue to be concerned; the selection of an appropriate process set is also of importance. Therefore, it is not enough to associate pheromones with edges; nodes should also be considered as elements in a route. For example, pheromone $\tau(u, v)$ only represents the attractiveness of the path $(u, v)$, rather than that of the destination $v$. It will only attract ants that are currently at node $u$, rather than ants at all nearby positions.

We therefore extend beyond the classical ACO of depositing pheromones on edges only, and to introduce pheromone deposit associated with nodes as well. In our previous two-stage ACO (Wong et al. 2012), the IPPS solution procedure involves the following subsequent steps: (i) selecting an appropriate set of processes; and (ii) determining the sequence. To begin with, pheromones are associated with nodes to determine the preferable processes; after the ants have converged on a certain node set for a number of iterations, pheromones associated with edges will take place to determine an optimal sequence to schedule these processes. In this current paper, the algorithm is further enhanced that the two tasks are conducted simultaneously. Both types of pheromones are deposited at the same time, assigned with dynamically tuned weights:

$$\tau(u, v) = (1 - \sigma_i)\mu(v) + \sigma_i \lambda(u, v) \tag{3}$$

where $\mu(v)$ is the pheromone associated with node $v$, $\lambda(u, v)$ is the pheromone associated with edge $(u, v)$. The respective weights are determined by the convergence rate, $\sigma_i$. Thus the dominator will gradually transfer from node-based pheromone to edge-based pheromone, as the ants tend to converge on a specific node set. This is the serialization of the discrete two-stage approach.

## Elitist strategy and MIN–MAX strategy in pheromone update

Elitist strategy and MIN–MAX strategy are advanced features of ACO heuristics. The elitist strategy (Dorigo et al. 1991; Dorigo and Gambardella 1996) is to provide strong additional reinforcement to the components belonging to the best route. In our algorithm, the pheromone update strategy can be summarized as follows:

$$\begin{cases} \mu(v) \leftarrow (1 - \rho)\mu(v) + \sum \Delta\mu_k(v) \\ \lambda(u, v) \leftarrow (1 - \rho)\lambda(u, v) + \sum \Delta\lambda_k(u, v) \end{cases} \tag{4}$$

where $\rho$ is the deterministic evaporation rate, $\Delta\mu_k(v)$ and $\Delta\lambda_k(u, v)$ is the deposited amount for individual node or edge by ant $k$ $(1 \le k \le K)$, which is determined by:

$$\Delta\mu_k(v) = \begin{cases} \frac{Q}{F(X_k)} D^{AB(v)}, & \text{if} \quad X_k = X_{\text{ib}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\Delta\lambda_k(u, v) = \begin{cases} \frac{Q}{F(X_k)} D^{AB(u,v)}, & \text{if} \quad X_k = X_{\text{ib}} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $Q$ is a positive constant, $D$ is a positive constant larger than 1, $X_k$ is the solution by ant $k$, $F(X_k)$ is its fitness value, $X_{\text{ib}} = \min\{X_k\}$ is the best solution found in the iteration ("iteration-best" solution). $A$ and $B(v)/B(u, v)$ are factors of overall solution quality, and global pheromone level of individual node or edge. This is a relatively extreme elitist strategy, which prevents all ants from depositing pheromone to their solution, except for the "iteration-best" solution.

The MIN-MAX ant system was introduced by Stützle (1997) to avoid excessive accumulation of pheromones on certain routes as well as over-evaporation by limiting all pheromone trails within a reasonable range $[\tau_{\min}, \tau_{\max}]$, while all pheromone trials are initialized to the maximum value at the beginning. This strategy is also used in our study.

The factors $A$ and $B$ are designed for further elitist strategy. The factor $A$ represents the *overall solution quality* of the solution found:

$$A = \begin{cases} 0, & \text{if} \quad i = 1 \\ \frac{\overline{F} - F(X_{\text{ib}})}{\overline{F} - \hat{F}}, & \text{if } i > 1, F(X_{\text{ib}}) < \overline{F} \\ -\infty, & \text{if} \quad i > 1, F(X_{\text{ib}}) \ge \overline{F} \end{cases} \tag{7}$$

where $\overline{F}$is the average value of fitness for all solutions found since the start of the algorithm, $\hat{F}$ is the best fitness value since the start of the algorithm. Note that $A$ is assigned to $-\infty$ when the "iteration-best" solution is not better than average, leading to a zero deposit amount. For an $A$ value which is less than, but close to 1, it refers to a good solution close to the best-so-far solution; while for a new better solution, the $A$ value will be larger than 1.

The factor $B$ stands for the *global pheromone level* of individual components:

$$\begin{cases} B(v) = \frac{\tau_{\max} - \mu(v)}{\tau_{\max} - \tau_{\min}} \\ B(u, v) = \frac{\tau_{\max} - \lambda(u,v)}{\tau_{\max} - \tau_{\min}} \end{cases} \tag{8}$$

The value of $B$ is bounded within [0, 1], while larger values indicate lower global pheromone level. It is obvious that nodes or edges with lower pheromone level will be deposited

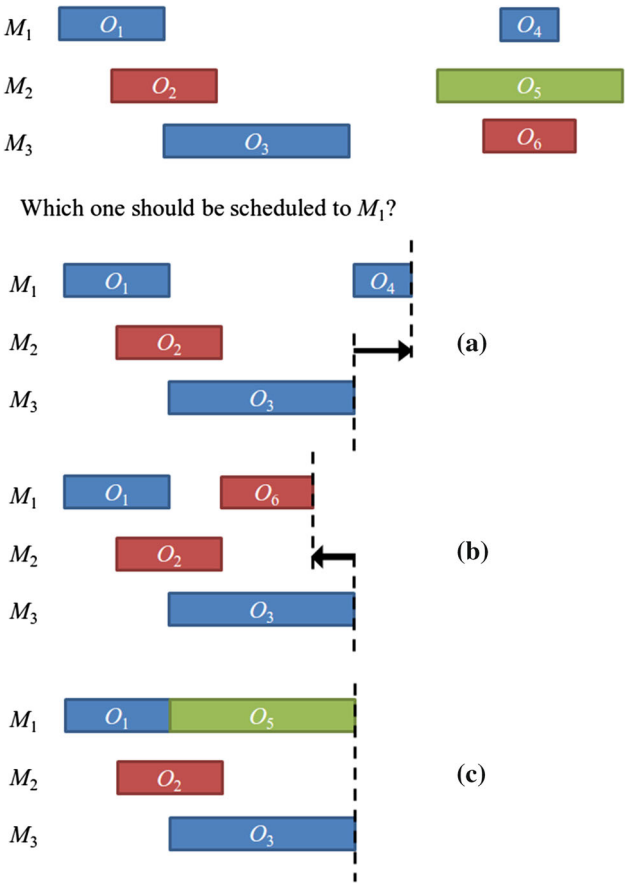


**Fig. 4** Different dispatching rules of scheduling

with more pheromone. This balancing mechanism is reasonable because components with lower pheromone level are less frequently used. When a favourable solution is obtained and deposited with pheromone, those less frequently used components apparently contribute more to such a solution, in comparison to those "popular" components which also frequently exist in other poorer solutions but do not make them favourable. The idea is encouraged by the exploration of "good" genes and elimination of "bad" ones in genetic algorithm (GA).

Dispatching rule: determine the heuristic desirability

As discussed in section "ACO heuristic", the dispatching rule is an important issue in scheduling problems. It has long been studied and discussed in various types of scheduling problem as it significant affects the optimality of schedule. The most commonly used dispatching rules are: shortest process time (SPT) first; longest processing time (LPT) first (looks weird, but might be favourable in some special situations); earliest starting time (EST) first; and earliest finishing time (EFT) first. The example in Fig. 4 illustrates how these strategies are applied. Under the specific circumstance, SPT tends to choose $O_4$, LPT and EST tend to choose $O_5$, EFT tends to choose $O_6$.

**Table 1** Comparison of different dispatching rules for problem 24

| Run no. | Strategy | | | | Best |
|---|---|---|---|---|---|
| | SPT | LPT | EST | EFT | |
| 1 | 523 | 517 | 511 | 487 | EFT |
| 2 | 514 | 518 | 508 | 490 | EFT |
| 3 | 521 | 510 | 520 | 486 | EFT |
| 4 | 509 | 512 | 530 | 497 | EFT |
| 5 | 515 | 519 | 517 | 486 | EFT |
| 6 | 510 | 514 | 530 | 492 | EFT |
| 7 | 510 | 520 | 521 | 487 | EFT |
| 8 | 514 | 517 | 522 | 485 | EFT |
| 9 | 509 | 525 | 521 | 493 | EFT |
| 10 | 507 | 518 | 523 | 477 | EFT |
| Avg. | 513.2 | 517.0 | 520.3 | 488.0 | EFT |

To improve the performance, we have attempted different dispatching rules in the proposed ACO algorithm. The results are compared using the 24 problem test case by Kim et al. (2003) (the parameters and test cases are discussed in the section "Parameter tuning of E-ACO"). Table 1 shows the results applying the different dispatching rules for 10 runs of one test case (problem 24, Kim et al. (2003)):

Although the EFT strategy encourages the ants to choose the optimal process at each step (this is the true sense "greedy strategy" for scheduling), it cannot guarantee that the final solution is globally optimal. Nevertheless, it is still found to outperform the other three.

The formula to compute the heuristic desirability under the EFT strategy is given by:

$$\eta(u,v) = \begin{cases} \frac{C}{\Delta T(u,v)}, & \text{if} \quad \Delta T(u,v) \geq 1. \\ C(2 - \Delta T(u,v)), & \text{if} \quad \Delta T(u,v) < 1. \end{cases} \tag{9}$$

where $C$ is a positive constant; $u$ is the previous node; $v$ is the current node; $FT(u,v)$ is the finishing time of the corresponding process of node $v$ with $u$ being the previous node; $\Delta T(u,v)$ is the difference between the finishing time $FT(u,v)$ and the current makepan $CT_{\max}$, and

$$\Delta T(u,v) = FT(u,v) - CT_{\max} \tag{10}$$

which can be either positive (Fig. 4a), negative (Fig. 4b) or zero (Fig. 4c).

Algorithm termination

The most commonly used method to determine when to terminate the algorithm is to limit the total number of iterations,

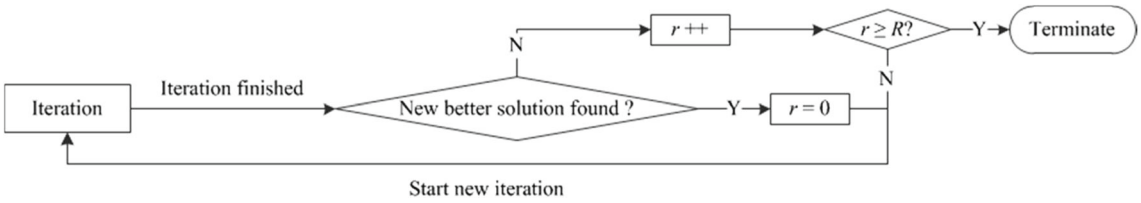


**Fig. 5** Termination control

which is quite easy to implement. However, the number of iterations needed to obtain a satisfactory solution varies in each run. It is due to many factors such as problem complexity, randomness, etc. Since the test case we are using includes problems with different complexities, a more reasonable manner is implemented to control the termination of algorithm. Figure 5 depicts the termination control strategy, the approach is to limit the number of "consecutive non-improving iteration", recorded by a counter $r$, by a maximum tolerance $R$. This strategy effectively reduces the waste of computation power.

## Parameter tuning of E-ACO

To reach the most satisfactory performance of the algorithm, a series of experiments have been conducted to determine an appropriate configuration of parameters. The following parameters are to be determined: algorithm constants $C$, $Q$ and $D$; respective weights of pheromone and heuristic desirability $\alpha$ and $\beta$; ant colony size $K$; pheromone evaporation rate $\rho$; minimum and maximum values of pheromone $\tau_{\min}$, $\tau_{\max}$ and maximum tolerance for "consecutive non-improving iterations" $R$.

A review on parameter adaption of ACO could be found in a recent literature (Stützle et al. 2012). The methodology has been classified into two categories: offline tuning in which parameters are determined before the algorithm is practically executed to solve the problem; and online tuning which involves parameter modification while solving a problem instance. Nevertheless, most of the approaches are based on trial-and-error, which is time-consuming, human-intensive and error-prone. There is no directly available criterion on how to set ACO parameters on scheduling problems.

### Test case

The test case used in this study is adopted from Kim et al. (2003), which has been widely used in many other IPPS researches. For comparison purpose, we have chosen this popular problem set of IPPS with a full consideration of manufacturing flexibilities. Each problem in the set is a combination of different jobs from 18 jobs in total with different complexities, and there are 15 available machines to handle these jobs. The details are listed in Table 2.

As an indicative purpose, the makespan lower bound of each of the 24 problem sets is also presented in Table 2. With respect to the IPPS problem modelling assumptions listed in section "Problem formulation", the IPPS problem models are governed by the constraints that each machine can process only one job at a time; and each job can only be processed by at most one machine at any time. With this consideration, the makespan lower bound for each problem set is determined by taking the maximum of the shortest process plans of each job. Take the two-jobs five-machines sample problem (see Fig. 1) as an example:

- For Job 1, $O_{1,1}$ and $O_{1,5}$ are compulsory, take the machine with the shortest processing time for them (60 and 30 respectively); for the OR-branch, we may choose to process $O_{1,2} - O_{1,3}$ or $O_{1,4}$, take the branch with the shortest possible total processing time (branch $O_{1,2} - O_{1,3}$, 30 for $O_{1,2}$ and 20 for $O_{1,3}$). Hence the lower bound for this job is $60 + 30 + 20 + 30 = 140$.
- For Job 2, all operations are compulsory, take the machine with the shortest processing time for them, lower bound $= 35 + 25 + 20 + 35 + 40 = 155$.

These values of lower bound are the minimum times (i.e., best process plans) required to complete these jobs if processed individually. As stated, we use $\max\{140, 155\} = 155$ as the lower bound for the sample problem.

To set the parameters for E-ACO, numerous experiments were conducted to study the impact and effects of different parameters. Accordingly, the following empirical parameters have been adopted: $C = 150$, $Q = 600$, $D = 15$, $\rho = 0.35$, $\alpha = 1.0$, $\beta = 2.0$, $\tau_0 = 10.0$, $\tau_{\min} = 1.0$, $\tau_{\max} = 20.0$, $K = 50$, $R = 50$. Each problem set is to be run 10 times in the experiment. Actually, we had attempted to test the effects of different number of runs and found that 10 runs was appropriate as additional number of runs, even up to 100, could only generate slightly better results. Table 3 shows the statistical analysis result for problem 24 for different run times with the empirical values of parameters and EFT dispatching rule.

It can be seen that 10 runs are already abundant for the experiments since the standard deviation is nearly the same as

**Table 2** Summary of test-bed problems in Kim et al. (2003)

| No. | No. of jobs | Job combination | Characteristic | Lower bound |
|---|---|---|---|---|
| 1 | 6 | 1, 2, 3, 10, 11, 12 | Low *PF* | 427 |
| 2 | 6 | 4, 5, 6, 13, 14, 15 | Mid *PF* | 343 |
| 3 | 6 | 7, 8, 9, 16, 17, 18 | High *PF* | 344 |
| 4 | 6 | 1, 4, 7, 10, 13, 16 | Low *SF* | 306 |
| 5 | 6 | 2, 5, 8, 11, 14, 17 | Mid *SF* | 304 |
| 6 | 6 | 3, 6, 9, 12, 15, 18 | High *SF* | 427 |
| 7 | 6 | 1, 4, 8, 12, 15, 17 | Low *OF* | 372 |
| 8 | 6 | 2, 6, 7, 10, 14, 18 | Mid *OF* | 342 |
| 9 | 6 | 3, 5, 9, 11, 13, 16 | High *OF* | 427 |
| 10 | 9 | 1, 2, 3, 5, 6, 10, 11, 12, 15 | Low or Mid *PF* | 427 |
| 11 | 9 | 4, 7, 8, 9, 13, 14, 16, 17, 18 | Mid or High *PF* | 344 |
| 12 | 9 | 1, 4, 5, 7, 8, 10, 13, 14, 16 | Low or Mid *SF* | 306 |
| 13 | 9 | 2, 3, 6, 9, 11, 12, 15, 17, 18 | Mid or High *SF* | 427 |
| 14 | 9 | 1, 2, 4, 7, 8, 12, 15, 17, 18 | Low or Mid *OF* | 372 |
| 15 | 9 | 3, 5, 6, 9, 10, 11, 13, 14, 16 | Mid or High *OF* | 427 |
| 16 | 12 | 1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15 | Low or mid *PF* | 427 |
| 17 | 12 | 4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18 | Mid or high *PF* | 344 |
| 18 | 12 | 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17 | Low or mid *SF* | 306 |
| 19 | 12 | 2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18 | Mid or high *SF* | 427 |
| 20 | 12 | 1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18 | Low or mid *OF* | 372 |
| 21 | 12 | 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18 | Mid or high *OF* | 427 |
| 22 | 15 | 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18 | – | 427 |
| 23 | 15 | 1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18 | – | 372 |
| 24 | 18 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 | – | 427 |

*PF* process flexibility (alternative process plans), *SF* sequence flexibility (alternative sequences), *OF* operation flexibility (alternative machines for operations)

**Table 3** Statistical analysis for different numbers of run time (problem 24)

| Run time | Mean | STDV | STDV/mean % |
|---|---|---|---|
| 5 | 485.80 | 4.595 | 0.95 |
| 10 | 486.30 | 5.541 | 1.14 |
| 25 | 486.88 | 5.535 | 1.14 |
| 50 | 486.40 | 5.481 | 1.13 |
| 75 | 486.33 | 5.973 | 1.23 |
| 100 | 486.44 | 5.754 | 1.18 |

that of larger numbers of runs, which is very slight (approximately 1 % of mean). Besides, 10 runs were also adopted in other literature using the same test case (Kim et al. 2003; Qiao and Lv 2012; Wong et al. 2012).

Constants C, Q, initial pheromone value $\tau_0$ and pheromone evaporation rate $\rho$

As depicted in Eqs. (5), (6) and (9), the constant $C$ determines the value of heuristic desirability, while the value of $Q$ decides how much pheromone is deposited each time. The core idea of ACO heuristic is that the pheromone and the heuristic desirability both decide where the ants will go. This requires that the numerical values of $\tau(u, v)$ and $\eta(u, v)$ are similar; otherwise the relative weight setting will be of no sense. As for what value should they be, it can be arbitrarily chosen for computational convenience. For instance, if we attempt to have the average of $\eta$ values to be around 10, the preferable value of $C$ could be computed by multiplying the estimation value of $\Delta T(u, v)$ by 10. The latter has been recorded with a mean value of around 15.0 in numerous experiments, hence the $C$ value was set to150 in our experiments.

Effects of the pheromone evaporation rate $\rho$ are summarized in Fig. 6 (adopted from Zhang 2012b).

In Leung et al. (2010), a very high evaporation rate ($\rho = 0.75$) was used, indicating that the pheromone trail left by an ant can almost be ignored after only a few iterations. The purpose of this setting was to accelerate the concentration of pheromone on routes. However, this leads to frequent reset of the algorithm (resetting all pheromone trails to their

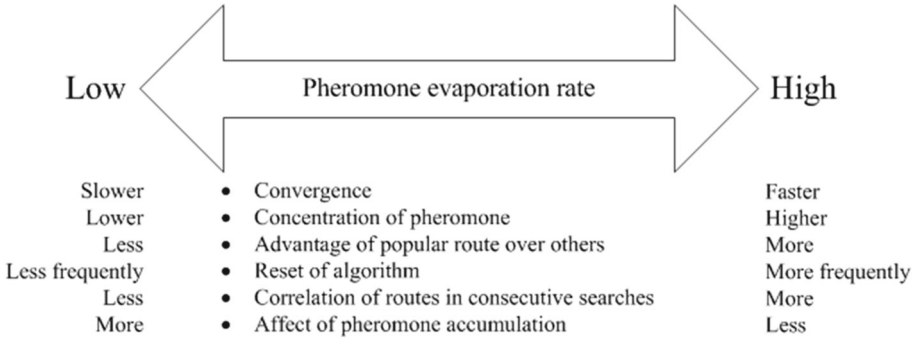


**Fig. 6** Summary of effects of the pheromone evaporation rate

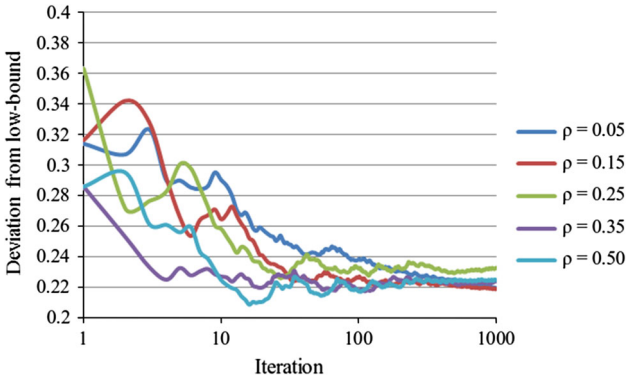


**Fig. 7** Comparison of the different pheromone evaporation rates

initial value $\tau_0$). This goes against the original intent of ACO which is a "constructive" heuristic whose self-adaptive property is all based on pheromone accumulation.

Based on this viewpoint, we did not use a high pheromone evaporation rate (i.e. exceeding 0.5) in our experiments. We conducted tests with different pheromone evaporation rates using the test case and found $\rho = 0.15$ to be the most favourable one. To illustrate, comparison on the effects of the different pheromone evaporation rates for an instance of problem no. 24 of the test case is presented in Fig. 7 and Table 4. As shown in Table 4, with a pheromone evaporation rate of 0.5, it took only five iterations to reduce the pheromone to 5 % of the original value, where 5 % = 1.0/20.0, $\tau_{\min}/\tau_{\max}$ (see section "Minimum and maximum value of pheromone trails $\tau_{\min}$, $\tau_{\max}$").

The initial value of pheromone trails $\tau_0$ should be set to a value that is expected to be "stable" in the long run. Since we have set the majority of $\eta$ to be around 10, we also assigned this value to be the initial value of pheromone trails.

The equation $Q/FT^* = \rho\tau_0$ was used to determine $Q$, where $FT^*$ is the fitness of the solution that the algorithm is expected to converge to in the long run. It is estimated to be 400, for the average of all test bed problems, and hence we get $Q = \rho\tau_0 FT^* = 0.15 \times 10 \times 400 = 600$.

**Table 4** Number of iterations required to reduce pheromone to 5 % of origin

| Pheromone evaporation rate | No. of iterations needed |
|---|---|
| 0.01 | 299 |
| 0.02 | 129 |
| 0.05 | 59 |
| 0.10 | 29 |
| 0.15 | 19 |
| 0.20 | 14 |
| 0.25 | 11 |
| 0.30 | 9 |
| 0.35 | 7 |
| 0.40 | 6 |
| 0.45 | 6 |
| 0.50 | 5 |

Elitist strategy variable D

This parameter is to purposely direct the ants to the favourable solutions and retain their critical components for further improvements. Different values of $D$ indicate different levels of bias on the "elitist" solution. $D = 1$ means no bias at all, while $D = 50$ would yield a very large bias. Figure 8 (adopted from Zhang 2012b) shows the comparison on using different values of $D$ for an instance of problem 24. Though not exhaustive, we have tested different values of $D$(1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 25, 30, 40, 50) in our experiments. The linear regression analysis was used to determine an appropriate $D$ value. Table 5 gives the results of different $D$ values for an instance of the test problem.

It can be seen that in the case $D = 1$ (Fig. 8a), due to the lack of bias, the result could not show a tendency of improvement. When $D$ increases to a value which is noticeable enough [in the cases $D = 5$ (Fig. 8b) and $D = 15$ (Fig. 8c)], an apparent descending tendency could be observed, indicating that the ants are gradually directed to those favourable routes and making continuous improvement of the quality

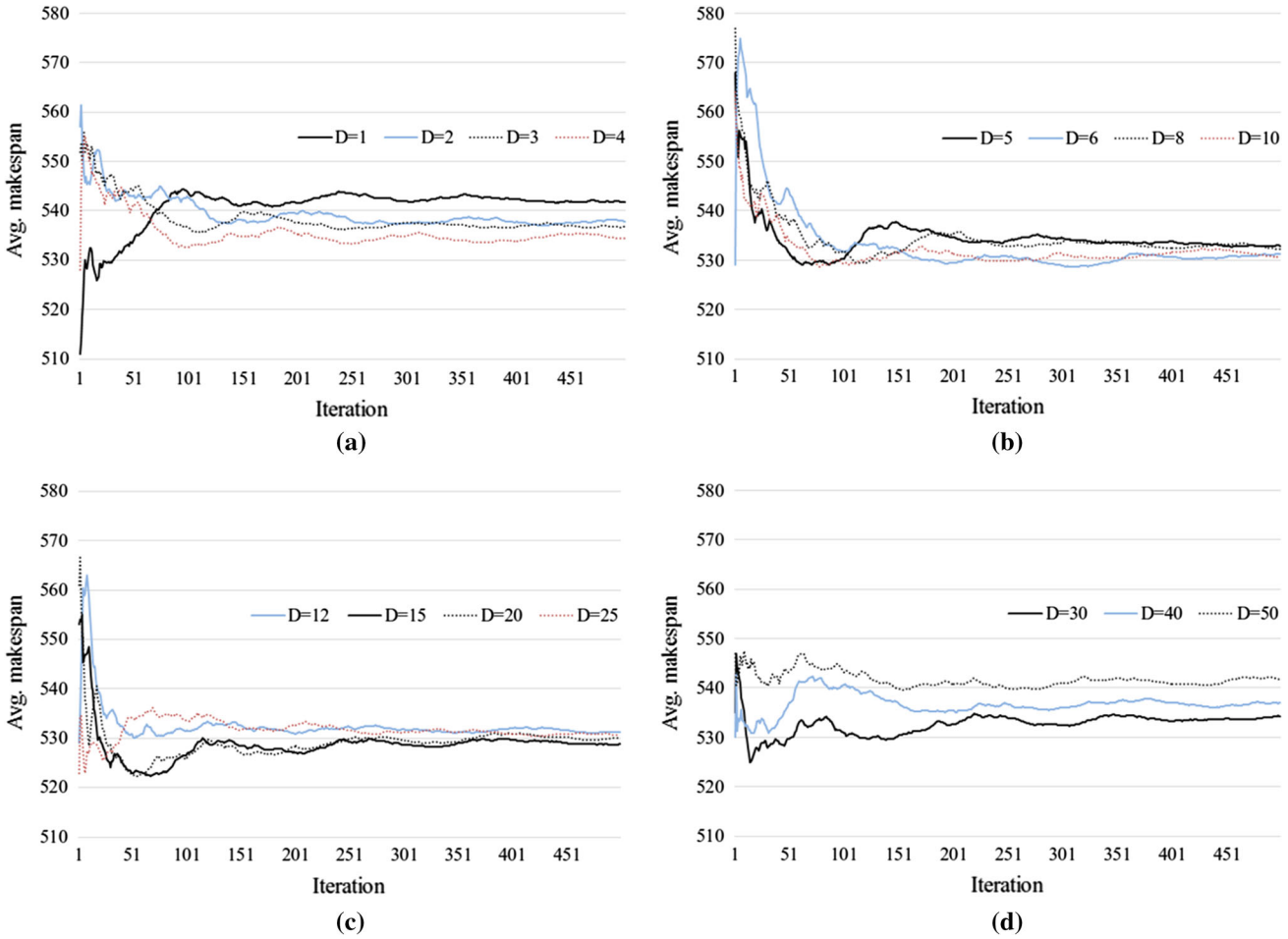


**Fig. 8** Comparison of average makespan over iterations by different $D$ values

of incoming solutions. However, with an over large $D$ value ($D \geq 30$, Fig. 8d), the ants are quickly directed to converge around some solutions which may only be the local optimal, leading to the fast descending tendency in the early stage and ascending again in further iteration, as no improvements could be made for a number of iterations.

In Table 5, noticeable results of linear regression analysis are marked with *, indicating that they are significantly better than the others in the same group of experiment. Most of these values appear around the $D$ value of 15, hence we chose to use $D = 15$ in our further experiments.

### Minimum and maximum value of pheromone trails $\tau_{\min}, \tau_{\max}$

It is easy to prove that, in normal ACO without elitist strategy, the maximum value of pheromone trail is bounded at around $Q/\rho FT^*$ (where $FT^*$ is the estimated optimal fitness). Since when the pheromone reaches this value, the evaporated amounted will be almost equal to the amount that is possibly deposited. This is often used to determine $\tau_{\max}$. In our approach, however, the elitist strategy is involved, which requires a larger upper bound of pheromone. Consider a node or edge with medium pheromone level ($B = 0.5$), belonging to a solution with ordinary quality (A = 0.5), we will then have $D^{AB} = 15^{0.5\times0.5} \approx 2$. Hence $\tau_{\max}$ is computed by $\tau_0 \times D^{AB} = 10 \times 2 = 20$. The lower bound of pheromone $\tau_{min}$ is determined to be 1.0 by trial and error, which makes the effect of pheromone trails not too small to be ignored.

### Relative weights of pheromone and heuristic desirability $\alpha, \beta$

According to Stützle et al. (2012), the $\alpha$ value is usually fixed to 1.0 and often omitted for many ant colony system applications. This is reasonable because it is the relative weights of pheromone and heuristic desirability that affects the node selection process. Thus we only need to tune the $\beta$ value. Taken the following results for problem 24 as an instance (Table 6):

It can be seen that the value of $\beta$ should not be too large or too small, otherwise one of the factors (pheromone or

**Table 5** Linear regression analysis: $y = ax + b$ (y: avg. fitness, $x$: iterations) for different $D$ values

| $D$ | Iter. 1−25 | | Iter. 1−50 | | Iter. 1−100 | | Iter. 1−250 | | Iter. 1−500 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $a$ | $b$ | $a$ | $b$ | $a$ | $b$ | $a$ | $b$ | $a$ | $b$ |
| 1 | −0.1954 | 552.14 | +0.2158 | 551.95 | +0.2195 | 523.74* | +0.0576 | 532.05* | +0.0158 | 536.84 |
| 2 | −0.4067 | 521.76 | −0.2126 | 523.90 | −0.0793 | 548.75 | −0.0385 | 546.11 | −0.0153 | 543.35 |
| 3 | −0.3510 | 554.61 | −0.2193 | 553.02 | −0.1622* | 551.87 | −0.0454 | 545.75 | −0.0149 | 542.26 |
| 4 | −0.2411 | 549.85 | −0.1447 | 548.93 | −0.1733* | 549.24 | −0.0436 | 542.47 | −0.0125 | 538.90 |
| 5 | −1.0336* | 561.47 | −0.5533* | 555.70 | −0.2481* | 548.19 | −0.0217 | 538.00 | −0.0086 | 536.60 |
| 6 | −0.2855 | 566.36 | −0.6227* | 569.18 | −0.3666* | 563.14 | −0.1058 | 549.83 | −0.0313 | 541.23 |
| 8 | −1.1170* | 565.84 | −0.4856* | 558.73 | −0.2620* | 553.36 | −0.0509 | 542.25 | −0.0155 | 538.23 |
| 10 | −0.6516 | 553.30 | −0.3221 | 549.75 | −0.2145* | 546.96 | −0.0439 | 538.46 | −0.0104 | 534.61 |
| 12 | −0.6383 | 554.46 | −0.4982* | 552.37 | −0.1869* | 544.88 | −0.0329 | 537.54 | −0.0096 | 534.87 |
| 15 | −1.2455* | 556.35 | −0.6221* | 548.50 | −0.2006* | 538.76* | −0.0129 | 529.92* | +0.0009 | 528.49* |
| 20 | −0.9545* | 552.09 | −0.5665* | 547.24 | −0.1765* | 537.79* | −0.0150 | 530.24* | +0.0033 | 528.78* |
| 25 | −0.0792 | 528.79 | +0.1085 | 526.19 | +0.0985 | 526.84* | +0.0086 | 531.25* | −0.0029 | 532.45* |
| 30 | −0.7591 | 542.12 | −0.1927 | 535.50 | +0.0059 | 531.20* | +0.0110 | 530.39* | +0.0072 | 530.86* |
| 40 | −0.0953 | 534.10 | +0.0211 | 532.52 | +0.1177 | 530.89* | +0.0019 | 536.67 | +0.0001 | 536.71 |
| 50 | −0.0792 | 545.10 | −0.0567 | 544.50 | +0.0133 | 543.19 | −0.0176 | 544.41 | −0.0045 | 542.82 |

*: Noticeable values. In early iterations, the slope $a$ is more important since it implies the improve rate of the algorithm, while the value of $b$ might be largely affected by randomness at this stage; in later iterations, the intercept $b$ becomes more important since it indicates the long term effectiveness of the algorithm; it can also be easily anticipated that the absolute value of $a$ will become smaller and smaller as a long run effect, yielding the flat looking of the curve

**Table 6** Comparison of different $\beta$ values

| $\beta$ value | Average fitness |
|---|---|
| 0.1 | 547.7 |
| 0.2 | 536.1 |
| 0.5 | 519.3 |
| 1.0 | 496.9 |
| 2.0 | 491.2 |
| 5.0 | 499.5 |
| 10.0 | 534.2 |

heuristic desirability) will have little effect in the node selection process, leading to poor algorithm performance. Therefore we chose $\beta = 2.0$, which was also proved to be most favourable for majority of the remaining problems.

Ant colony size K and termination controlling variable R

These parameters affect the algorithm efficiency. Apparently, with other conditions being the same, running the algorithm with more ants simultaneously will yield satisfactory solution with less iterations, by the cost of long time taken per iteration (since each ant requires a thread in CPU). On the other hand, a more complex problem will need more ants to explore the more complex routes. There should be a balance between efficiency and solution quality.

**Table 7** Comparison of different sizes of ant colony

| $K$ | Avg. fitness | Impr.‰ per additional ant | Total iter. | Total CPU time (s) | Avg. CPU time/iter. (s) |
|---|---|---|---|---|---|
| 1 | 531.7 | – | 451 | 17 | 0.038 |
| 5 | 515.7 | 7.52 | 408 | 38 | 0.092 |
| 10 | 510.5 | 2.02 | 409 | 62 | 0.152 |
| 25 | 501.5 | 1.18 | 427 | 147 | 0.343 |
| 50 | 491.2 | 0.82 | 496 | 341 | 0.688 |
| 100 | 490.7 | 0.02 | 392 | 560 | 1.427 |
| 250 | 488.4 | 0.03 | 393 | 1,381 | 3.515 |
| 500 | 485.7 | 0.02 | 390 | 2,833 | 7.263 |

The most complex problem in the set is problem 24, therefore if the ant colony is large enough for problem 24, we can be confident that it is large enough for the rest of the test-bed problems. Table 7 lists the results for different values of $K$ using $R = 50$.

It can be seen that the CPU time taken per iteration almost grow linearly as the ant colony size grows, especially for large values of the latter. When the ant colony size exceeds 50, the improvement made by adding more ants is little enough to be ignored. Therefore 50 ants will be enough for all test-bed problems.

The similar approach was used to determine $R$, by running problem 24 with different values of $R$ (Table 8):

**Table 8** Comparison of different values of "consecutive non-improving iteration" tolerance

| Run no. | $R = 10$ | $R = 25$ | $R = 50$ | $R = 100$ | $R = 150$ | $R = 250$ | $R = 500$ |
|---|---|---|---|---|---|---|---|
| 1 | 507 | 507 | 494 | 494 | 484 | 477 | 477 |
| 2 | 495 | 495 | 495 | 480 | 480 | 480 | 481 |
| 3 | 489 | 489 | 488 | 484 | 473 | 473 | 479 |
| 4 | 504 | 497 | 497 | 487 | 487 | 487 | 476 |
| 5 | 498 | 498 | 487 | 487 | 487 | 487 | 483 |
| 6 | 511 | 495 | 495 | 481 | 481 | 481 | 484 |
| 7 | 491 | 491 | 491 | 491 | 489 | 476 | 477 |
| 8 | 514 | 496 | 496 | 475 | 475 | 475 | 480 |
| 9 | 519 | 490 | 490 | 490 | 483 | 483 | 477 |
| 10 | 508 | 506 | 493 | 493 | 487 | 487 | 487 |
| Avg. | 503.6 | 496.4 | 492.6 | 486.2 | 482.6 | 480.6 | 480.1 |
| Impr. $^{0}/_{00}$ per additional $R$ | | 0.95 | 0.31 | 0.26 | 0.15 | 0.08 | 0.004 |

**Table 9** Comparison of makespan

| No. | SEA | | IGA | | 2-stage ACO | | E-ACO | | Best | Lower bound | Improved rate[a] (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. | Best | Avg. | Best | Avg. | Best | Avg. | Best | | | |
| 1 | 437.6 | 428 | 427.0 | 427[b] | 445.6 | 441 | 427.1 | 427[b] | IGA | 427 | 4.15 |
| 2 | 349.7 | 343[b] | 344.5 | 343[b] | 367.7 | 362 | 343.1 | 343[b] | E-ACO | 343 | 6.69 |
| 3 | 355.2 | 347 | 351.0 | 344[b] | 374.3 | 368 | 345.0 | 344[b] | E-ACO | 344 | 7.83 |
| 4 | 306.2 | 306[b] | 307.4 | 306[b] | 320.2 | 318 | 307.6 | 306[b] | SEA | 306 | 3.94 |
| 5 | 323.7 | 319 | 309.8 | 304[b] | 324.0 | 322 | 319.6 | 318 | IGA | 304 | 1.36 |
| 6 | 443.8 | 438 | 427.0 | 427[b] | 459.8 | 444 | 427.1 | 427[b] | IGA | 427 | 7.11 |
| 7 | 372.4 | 372[b] | 372.7 | 372[b] | 381.0 | 372[b] | 372.0 | 372[b] | E-ACO | 372 | 2.36 |
| 8 | 348.3 | 343 | 357.0 | 342[b] | 361.0 | 343 | 343.3 | 343 | E-ACO | 342 | 4.90 |
| 9 | 434.9 | 428 | 427.0 | 427[b] | 434.2 | 427[b] | 427.1 | 427[b] | IGA | 427 | 1.64 |
| 10 | 456.5 | 443 | 431.6 | 427[b] | 445.6 | 432 | 427.6 | 427[b] | E-ACO | 427 | 4.04 |
| 11 | 378.9 | 369 | 379.7 | 368 | 387.6 | 368 | 350.2 | 348 | E-ACO | 344 | 9.65 |
| 12 | 332.8 | 328 | 323.7 | 312 | 375.3 | 349 | 323.4 | 322 | E-ACO | 306 | 13.83 |
| 13 | 469.0 | 452 | 442.8 | 429 | 461.7 | 456 | 427.6 | 427[b] | E-ACO | 427 | 7.39 |
| 14 | 402.4 | 381 | 415.3 | 386 | 416.1 | 386 | 374.3 | 373 | E-ACO | 372 | 10.05 |
| 15 | 445.2 | 434 | 427.4 | 427[b] | 439.9 | 437 | 427.3 | 427[b] | E-ACO | 427 | 2.86 |
| 16 | 478.8 | 454 | 449.4 | 433 | 488.3 | 450 | 430.9 | 429 | E-ACO | 427 | 11.76 |
| 17 | 448.9 | 431 | 426.0 | 415 | 473.1 | 460 | 381.2 | 377 | E-ACO | 344 | 19.43 |
| 18 | 389.6 | 379 | 373.6 | 364 | 407.6 | 398 | 361.5 | 357 | E-ACO | 306 | 11.31 |
| 19 | 508.1 | 490 | 471.3 | 450 | 476.0 | 434 | 434.9 | 431 | E-ACO | 427 | 8.63 |
| 20 | 453.8 | 447 | 446.6 | 429 | 483.6 | 447 | 392.4 | 386 | E-ACO | 372 | 18.86 |
| 21 | 483.2 | 477 | 447.8 | 433 | 461.7 | 440 | 429.4 | 428 | E-ACO | 427 | 7.00 |
| 22 | 548.3 | 534 | 508.1 | 491 | 520.6 | 549 | 447.2 | 444 | E-ACO | 427 | 14.10 |
| 23 | 507.5 | 498 | 477.8 | 465 | 516.8 | 449 | 420.3 | 413 | E-ACO | 372 | 18.67 |
| 24 | 602.2 | 587 | 548.5 | 532 | 611.0 | 570 | 479.3 | 460 | E-ACO | 427 | 21.55 |

[a] Improve rate from 2-stage ACO

[b] The best results for these problems have already reached the theoretical lower bound of the problem and cannot be further improved

**Table 10** Comparison of mean flow time and CPU time(s)

| No. | SEA | | IGA | | 2-stage ACO | | Enhanced ACO | |
|---|---|---|---|---|---|---|---|---|
| | MFT | CPU time | MFT | CPU time | MFT | CPU Time | MFT | CPU Time |
| 1 | 318.9 | 61 | 313.3 | 11 | 356.7 | 10 | 317.3 | 17 |
| 2 | 287.7 | 69 | 292.2 | 11 | 338.1 | 8 | 293.4 | 15 |
| 3 | 304.8 | 82 | 307.8 | 11 | 339.1 | 9 | 306.4 | 14 |
| 4 | 251.3 | 66 | 258.7 | 8 | 289.1 | 7 | 266.6 | 14 |
| 5 | 289.3 | 64 | 263.5 | 8 | 309.7 | 7 | 282.7 | 11 |
| 6 | 384.7 | 73 | 382.3 | 13 | 416.5 | 11 | 376.7 | 20 |
| 7 | 314.1 | 69 | 317.4 | 9 | 345.9 | 7 | 328.4 | 11 |
| 8 | 295.2 | 67 | 305.3 | 17 | 328.3 | 9 | 301.4 | 13 |
| 9 | 298.9 | 73 | 292.7 | 9 | 377.3 | 10 | 296.6 | 21 |
| 10 | 349.2 | 136 | 356.4 | 17 | 403.8 | 17 | 340.7 | 34 |
| 11 | 312.9 | 166 | 318.6 | 16 | 332.2 | 13 | 307.0 | 31 |
| 12 | 279.6 | 143 | 278.6 | 13 | 344.0 | 11 | 286.0 | 24 |
| 13 | 387.0 | 161 | 383.2 | 19 | 419.4 | 18 | 367.7 | 39 |
| 14 | 346.9 | 151 | 351.6 | 16 | 367.5 | 11 | 343.5 | 26 |
| 15 | 316.1 | 156 | 319.7 | 14 | 361.6 | 18 | 308.1 | 33 |
| 16 | 359.7 | 334 | 364.5 | 23 | 420.4 | 22 | 348.9 | 50 |
| 17 | 364.7 | 435 | 368.1 | 23 | 428.3 | 22 | 340.4 | 64 |
| 18 | 322.5 | 357 | 316.3 | 20 | 363.6 | 20 | 309.6 | 53 |
| 19 | 406.4 | 418 | 396.5 | 28 | 441.0 | 27 | 369.1 | 78 |
| 20 | 372.0 | 384 | 371.7 | 26 | 420.4 | 18 | 347.7 | 55 |
| 21 | 365.4 | 392 | 365.3 | 24 | 415.5 | 27 | 341.1 | 67 |
| 22 | 417.8 | 1033 | 415.3 | 27 | 444.9 | 37 | 374.5 | 121 |
| 23 | 404.7 | 1017 | 399.7 | 26 | 450.8 | 39 | 366.6 | 93 |
| 24 | 452.9 | 1623 | 447.6 | 39 | 503.9 | 59 | 413.5 | 186 |

We can see that as iteration goes on, it becomes more difficult to improve the solution; the extent of improvement becomes less and less significant. It is not worth to consume much time and computation efforts to obtain a slight improvement. The results of $R = 50$ is already satisfactory enough, after which the improvement is quite minor. Therefore, this value is chosen for all further experiments.

## Summary

No doubt, the performance of a meta-heuristic algorithm depends on the setting of its parameters. The parameter tuning technique used in this study is still preliminary, it is basically by trial and error and time consuming. Besides, the same parameter set is used for all problems, which is somehow unreasonable. However, the results can be referred to in future studies. They can be used as initial values for more advanced online self-adaptive parameter tuning strategy.

An empirical guideline for ACO parameter tuning is listed as follows:

- Use the following sequence to determine all parameters.

(1) For $\tau_0$, decide a value of computational convenience for the average of $\eta$ values and long run pheromone values, initiate all pheromone trails to this value.
(2) For $C$, set an initial value which makes the average of $\eta$ values at around $\tau_0$. This is done by estimate or compute the average value of desirability factor, that is, $\Delta T(v)$ for EFT scheduling (see Eq. 9), processing time for SPT or LPT scheduling, or distance between cities for TSP, etc. Then adapt the parameter by re-computing when there are incoming jobs making the $\eta$ value bias from expected.
(3) For $\rho$, which should not be too large, we recommend a value with the bound [0.05, 0.50]. 0.15 will be suitable for most scheduling problems. For relatively simple problem, a larger value might be of preference, while for very complex problems, a smaller value could be used.
(4) For $Q$, estimate the long fun fitness of the problem $FT^*$, compute $Q$ by $Q = \rho\tau_0 FT^*$.

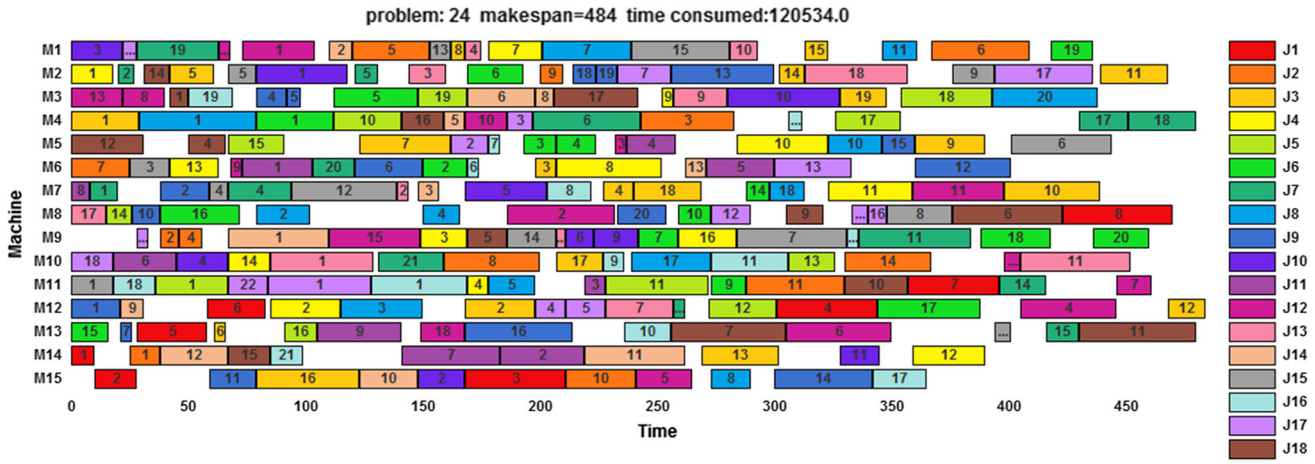


**Fig. 9** Gantt chart of a sample feasible schedule for problem 24

(5) Determining the elitist strategy variable ($D$ in our study) is somehow arbitrary, which may involve "trial-and-error";
(6) If elitist strategy is applied, use $\tau_{max} = \tau_0 \times$ elitist strategy variable; else use $\tau_{max} = \tau_0$. For $\tau_{min}$, "trial-and-error" is needed, usually 5−10 % of $\tau_{max}$ is preferable.
(7) For relative weights of pheromone and heuristic desirability, $\alpha$ is usually fixed to 1.0; for $\beta$, still "trial-and-error", 2.0 is the recommended value.
(8) Variables that affect algorithm efficiency, say, ant colony size and maximum number of iterations (or maximum tolerance for "consecutive non-improving iterations", or other termination controlling techniques, etc.), are to be determined at last, depending on the problem complexity and the users' own willing of how good the solutions should be.

- For the above parameters whose determining involves "trial-and-error", make them self-adaptive might be a smart choice, using the recommended values as initial values.

The parameter set used for the evaluation in this paper is as follows: $K = 50$, $\rho = 0.15$, $\alpha = 1.0$, $\beta = 2.0$, $C = 150$, $Q = 600$, $\tau_{min} = 0.1$, $\tau_0 = 10.0$, $\tau_{max} = 20.0$, $D = 15$, $R = 50$.

## Evaluation

To evaluate the E-ACO heuristic, all 24 test bed problems have been run with the algorithm using an Intel Core i7 3.4 GHz, 4 GB RAM desktop computer, each for 10 times with the average value taken. The results are compared with the symbiotic evolutionary algorithm (SEA) (Kim et al. 2003), improved genetic algorithm (IGA) (Qiao and Lv 2012) and two-stage ACO algorithm (Wong et al. 2012), listed as follows (Tables 9, 10):

Figure 9 gives the Gantt chart of a sample feasible schedule for problem 24 with makespan 484.

We can see that in terms of makespan, E-ACO outperforms the other heuristics in most of the test bed problems (19 of 24). The solution quality has been improved significantly from the two-stage ACO approach, which proves the potential effectiveness for using ACO heuristics to solve IPPS problems.

On the other hand, the computation time for E-ACO is generally longer than that of 2-stage ACO under similar conditions of parameters. This might be caused by the involvement of massive computation in applying the EFT dispatching rule and elitist strategy. Nevertheless, the efficiency is still acceptable, which requires only 3 min for the most complex problem in the test cases.

## Conclusions and future work

An enhanced ACO approach for IPPS is proposed in this paper. By studying the algorithm and conducting the tests, the weaknesses and limitation of original ACO for IPPS are evaluated and identified, and corresponding modifications are designed to enhance the performance. The parameter tuning of ACO has been studied and an empirical guideline is proposed. The advantages of the enhanced algorithm have been illustrated by experimental evaluation.

For further study, dynamic scheduling involving unexpected disturbances could be a potential orientation. Multiple performance criteria optimization, such as mean flow time, machine utilization, as well as ACO with self-adaptive parameter setting, are also of interest in our future work.

**Acknowledgments** The work described in this paper is fully supported by a grant from the Research Grants Council of Hong Kong (Project Code HKU 718809E).

## References


Arnaout, J. P., Musa, R., & Rabadi, G. (2014). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines—part ii: Enhancements and experimentations. *Journal of Intelligent Manufacturing*, *25*(1), 43–53.

Arnaout, J. P., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*(6), 693–701.

Balasubramanian, S., Maturana, F. P., & Norrie, D. H. (1996). Multi-agent planning and coordination for distributed concurrent engineering. *International Journal of Cooperative Information Systems*, *05*(02n03), 153–179.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, *2*(4), 353–373.

Chiang, C. W., Huang, Y. Q., & Wang, W. Y. (2008). Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent and Fuzzy Systems*, *19*(4–5), 345–358.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Italy: Politecnico di Milano.

Dorigo, M., & Gambardella, L. M. (1996). *A study of some properties of ant-Q parallel problem solving from nature-PPSN IV*. Berlin: Springer.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). The ant system: An autocatalytic optimizing process. Technical report.

Gu, P., Balasubramanian, S., & Norrie, D. H. (1997). Bidding-based process planning and scheduling in a multi-agent system. *Computers and Industrial Engineering*, *32*(2), 477–496.

Guo, Y. W., Li, W. D., Mileham, A. R., & Owen, G. W. (2009). Applications of particle swarm optimisation in integrated process planning and scheduling. *Robotics and Computer-Integrated Manufacturing*, *25*(2), 280–288.

Ho, Y. C., & Moodie, C. L. (1996). Solving cell formation problems in a manufacturing environment with flexible processing and routing capabilities. *International Journal of Production Research*, *34*(10), 2901–2923.

Iwata, K., Murotsu, Y., & Oba, F. (1978). Optimization of selection of machine-tool, loading sequence of parts and machining conditions in job-shop type machining systems. *Annals of the CIRP*, *27*, 447–451.

Khoshnevis, B., & Chen, Q. M. (1991). Integration of process planning and scheduling functions. *Journal of Intelligent Manufacturing*, *2*(3), 165–175.

Kim, Y. K., Kim, J. Y., & Shin, K. S. (2007). An asymmetric multileveled symbiotic evolutionary algorithm for integrated FMS scheduling. *Journal of Intelligent Manufacturing*, *18*(6), 631–645.

Kim, Y. K., Park, K., & Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers and Operations Research*, *30*(8), 1151–1171.

Kis, T. (2003). Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, *151*(2), 307–332.

Ko, C. S., Kim, T., & Hwang, H. (2001). External partner selection using tabu search heuristics in distributed manufacturing. *International Journal of Production Research*, *39*(17), 3959–3974.

Korytkowski, P., Rymaszewski, S., & Wisniewski, T. (2013). Ant colony optimization for job shop scheduling using multi-attribute dispatching rules. *International Journal of Advanced Manufacturing Technology*, *67*(1–4), 231–241.

Kumar, M., & Rajotia, S. (2003). Integration of scheduling with computer aided process planning. *Journal of Materials Processing Technology*, *138*(1–3), 297–300.

Kumar, R., Tiwari, M. K., & Shankar, R. (2003). Scheduling of flexible manufacturing systems: An ant colony optimization approach. *Proceedings of the Institution of Mechanical Engineers Part B-Journal of Engineering Manufacture*, *217*(10), 1443–1453.

Lawrynowicz, A. (2008). Integration of production planning and scheduling using an expert system and a genetic algorithm. *Journal of the Operational Research Society*, *59*(4), 455–463.

Lee, H., & Kim, S.-S. (2001). Integration of process planning and scheduling using simulation based genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, *18*(8), 586–590.

Leung, C. W., Wong, T. N., Mak, K. L., & Fung, R. Y. K. (2010). Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers and Industrial Engineering*, *59*(1), 166–180.

Li, X., Gao, L., & Li, W. (2012). Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. *Expert Systems with Applications*, *39*(1), 288–297.

Li, X., Shao, X., Gao, L., & Qian, W. (2010). An effective hybrid algorithm for integrated process planning and scheduling. *International Journal of Production Economics*, *126*(2), 289–298.

Lin, C. W., Lin, Y. K., & Hsieh, H. T. (2013). Ant colony optimization for unrelated parallel machine scheduling. *International Journal of Advanced Manufacturing Technology*, *67*(1–4), 35–45.

Liu, X.-J., Yi, H., & Ni, Z.-H. (2013). Application of ant colony optimization algorithm in process planning optimization. *Journal of Intelligent Manufacturing*, *24*(1), 1–13.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, *6*(4), 333–346.

Morad, N., & Zalzala, A. (1999). Genetic algorithms in integrated process planning and scheduling. *Journal of Intelligent Manufacturing*, *10*(2), 169–179.

Nasr, N., & Elsayed, E. A. (1990). Job shop scheduling with alternative machines. *International Journal of Production Research*, *28*(9), 1595–1609.

Palmer, G. J. (1996). A simulated annealing approach to integrated production scheduling. *Journal of Intelligent Manufacturing*, *7*(3), 163–176.

Qiao, L. H., & Lv, S. P. (2012). An improved genetic algorithm for integrated process planning and scheduling. *International Journal of Advanced Manufacturing Technology*, *58*(5–8), 727–740.

Seo, Y., & Egbelu, P. J. (1996). Process plan selection based on product mix and production volume. *International Journal of Production Research*, *34*(9), 2639–2655.

Shao, X. Y., Li, X. Y., Gao, L., & Zhang, C. Y. (2009). Integration of process planning and scheduling—A modified genetic algorithm-based approach. *Computers and Operations Research*, *36*(6), 2082–2096.

Stützle, T. (1997). MAX–MIN ant system for quadratic assignment problems: Technical report AIDA-97-4. FB Infomatik, TU Darmstadt, Germany: FG Intellektik.

Stützle, T., López-Ibánez, M., Pellegrini, P., Maur, M., de Oca, M. M., Birattari, M., et al. (2012). Parameter adaptation in ant colony optimization. In *Autonomous search* (pp. 191–215). Springer.

Tavares Neto, R., Godinho Filho, M., & da Silva, F. (2013). An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-013-0811-5.

Tehrani Nik Nejad, H., Sugimura, N., Iwamura, K., & Tanimizu, Y. (2010). Multi agent architecture for dynamic incremental process planning in the flexible manufacturing system. *Journal of Intelligent Manufacturing*, *21*(4), 487–499.

Usher, J. M. (2003). Negotiation-based routing in job shops via collaborative agents. *Journal of Intelligent Manufacturing*, *14*(5), 485–499.

Ventura, J., & Yoon, S.-H. (2013). A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. *Journal of Intelligent Manufacturing*, *24*(6), 1185–1196.

Wang, J., Fan, X., Zhang, C., & Wan, S. (2014). A graph-based ant colony optimization approach for integrated process planning and scheduling. *Chinese Journal of Chemical Engineering*, *22*(7), 748–753.

Weintraub, A., Cormier, D., Hodgson, T., King, R., Wilson, J., & Zozom, A. (1999). Scheduling with alternatives: A link between process planning and scheduling. *Iie Transactions*, *31*(11), 1093–1102.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006a). An agent-based negotiation approach to integrate process planning and scheduling. *International Journal of Production Research*, *44*(7), 1331–1351.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006b). Dynamic shopfloor scheduling in multi-agent manufacturing systems. *Expert Systems with Applications*, *31*(3), 486–494.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006c). Integrated process planning and scheduling/rescheduling—an agent-based approach. *International Journal of Production Research*, *44*(18–19), 3627–3655.

Wong, T. N., Zhang, S. C., Wang, G., & Zhang, L. P. (2012). Integrated process planning and scheduling—Multi-agent system with two-stage ant colony optimisation algorithm. *International Journal of Production Research*, *50*(21), 6188–6201.

Wu, W.-H., Cheng, S.-R., Wu, C.-C., & Yin, Y. (2012). Ant colony algorithms for a two-agent scheduling with sum-of processing times-based learning and deteriorating considerations. *Journal of Intelligent Manufacturing*, *23*(5), 1985–1993.

Zhang, H. (2012a). Ant colony optimization for multimode resource-constrained project scheduling. *Journal of Management in Engineering*, *28*(2), 150–159.

Zhang, S. (2012b). *An enhanced ant colony optimization approach for integrating process planning and scheduling based on multi-agent system*. (Master of Phylosophy M.Phil. thesis), The Uinversity of Hong Kong, Hong Kong SAR, China.

Zhu, H. Y., Ye, W. H., & Bei, G. X. (2009, 26–29 Nov. 2009). *A particle swarm optimization for integrated process planning and scheduling*. Paper presented at the IEEE 10th international conference on computer-aided industrial design and conceptual design, 2009. CAID and CD 2009.