Flexible Services and Manufacturing Journal (2020) 32:523–560
https://doi.org/10.1007/s10696-019-09360-9

Evolving ant colony system for large-sized integrated process planning and scheduling problem considering sequence-dependent setup times

Chunghun Ha[1]

Published online: 31 May 2019

Abstract
This paper proposes a new ant colony optimization (ACO) algorithm suitable for integrated process planning and scheduling (IPPS) that optimizes both process planning and scheduling simultaneously. The IPPS covered in this study, when compared to the conventional IPPS, is more flexible and complicated because sequence-dependent setups and tool-related capacity constraints are additionally considered. Traditional ACOs have limitations in improving the solution quality and computation time for IPPS. The high flexibility and complexity of IPPS requires a large size of repository for pheromone trails and it causes the long computation time for updating them, excessive evaporation of pheromones, and unbalancing between pheromones and desirability. In the proposed ACO, each ant agent improves their own incumbent solution or finds a new solution using the pheromone trails that is composed of the experience information of the colony. Therefore, the proposed ACO conducts individual and cooperative evolving at the same time. Furthermore, we propose a simplified updating rule for pheromone trails and standardization of the transition probability to increase efficiency of the algorithm. Experimental results show that the proposed ACO is superior to recently proposed meta-heuristics for benchmark problems of different sizes in terms of both solution quality and computation time.

Keywords Integrated process planning and scheduling problem · Ant colony optimization · Sequence-dependent setup

✉ Chunghun Ha
chunghun.ha@hongik.ac.kr

[1] Department of Industrial Engineering, Hongik University, 94 Wausan-Ro, Mapo-Gu, Seoul 04066, Korea

1 Introduction

Process planning and scheduling are two key activities in manufacturing planning. Process planning or computer aided process planning (CAPP) is the activity of determining the appropriate process, tool, sequence, and processing conditions to transform raw materials into a designed shape. CAPP plays a central role in building computer integrated manufacturing (CIM) systems by connecting computer aided design (CAD) and computer aided manufacturing (CAM) (Reddy 1999; Tan and Khoshnevis 2000). Scheduling is an activity that determines the starting times and the completion times of the operations to be processed. Optimization of these two activities has traditionally been carried out independently and sequentially, that is, the scheduling is conducted with the predetermined process plan. Shao et al. (2009), however, argued that the sequential approach involves four vulnerabilities. First, the optimal plan can be often infeasible according to the status of the shop floor at the time of execution. Second, even though the plan is optimized by reflecting the real-time status, it also may become infeasible due to the time delay between planning and execution. Third, the optimal schedule based on the optimal process plan can lead to shortages of certain resources and bottlenecks in certain machines. Fourth, optimization for a single-objective may be inappropriate for multi-objective practical manufacturing.

To overcome those shortcomings, integrated process planning and scheduling (IPPS) that optimizes the two distinct activities simultaneously has been considered. It is known that IPPS can improve facility efficiency, mean flow time, work-in-process, machine utilization, and robustness of the shop floor (Rachamadugu and Stecke 1994; Shao et al. 2009). Over the past two decades, research on IPPS has grown. In previous studies, various flexibilities and constraints were covered to reflect an actual manufacturing environment. Flexibilities include process flexibility for alternative process routings, sequence flexibility for alternative operation orders, machine and tool flexibility for alternative facilities, and TAD flexibility for alternative tool access directions (TADs) (Kim et al. 2007; Petrović et al. 2016; Dou et al. 2018). As constraints, precedence relations between operations are essential. Capacity constraints such as tool capacity and tool magazine capacity and sequence-dependent setups have been rarely dealt with in a few studies (Kim et al. 2007; Srinivas et al. 2012).

Although various flexibilities and constraints have been handled in IPPS, no study has covered all of the flexibilities and constraints, i.e., each individual study has only considered some of them by adopting impractical assumptions. A typical example is setups. Setups include handling tools, setting the jigs and fixtures, loading and unloading workpieces, and inspecting the material. There are two types of setups: sequence-independent setups and sequence-dependent setups. If the setups of the current operation are dependent on the preceding operations, they are sequence-dependent; otherwise, they are sequence-independent. Most existing studies assume sequence-independent setups and merge the constant setup times into the processing time. However, in flexible manufacturing system (FMS), where IPPS is primarily applied, the sequence-dependent setups make

up a majority of setups, because multi-purpose machines are the main facilities (Eren 2010). In this case, separate consideration of the sequence-dependent setup times can draw a better schedule (Srinivas et al. 2012).

Both process planning and scheduling belong to the NP-hard class, and therefore so does IPPS. This implies IPPS has a high complexity and huge solution space. Hence, most researchers have preferred meta-heuristics as an optimization approach for IPPS, which include genetic algorithm (GA), ant colony optimization (ACO), and particle swam optimization (PSO). Among them, ACO is an optimization method that utilizes the stigmergic behavior of multi-agents represented by an ant colony. For each iteration, all ant agents seek a route, i.e., a solution for the problem, and the information of the best route in the colony is accumulated in the pheromone trails. It is combined with the preferred node selection rule, so called heuristic desirability, and is used by an ant agent to search for a new route. This procedure is a kind of collective intelligence. ACO has been successfully applied to the optimization of NP-Hard problems, such as the traveling salesman problem (TSP), scheduling, routing, and digital image processing (Chandra and Baskaran 2012). ACO, however, is often vulnerable to very complicated problems like IPPS. As IPPS considers various flexibilities, the number of nodes and edges are large, and subsequently the repository size of the pheromone trails is large. To maintain and update the pheromone trails, a considerable amount of computation is required for deposition and evaporation procedures. Moreover, as the information belonging to the best route is only a small fraction of the information in all the pheromone trails, the amount of evaporated pheromone is larger than the amount of deposited pheromone. As the iteration is progressed, the influence of the pheromone trails weakens, and the influence of the heuristic desirability increases relatively. Therefore, the stigmergic property of ACO does not operate properly.

The goal of this study is to develop an efficient ACO for IPPS considering almost all the flexibilities and constraints previously dealt with. Covered flexibilities include process, sequence, machine, tool, and TAD flexibility. Considered constraints include precedence relationship, tool capacity, tool magazine capacity, and sequence-dependent setup times. Transportation time, which is essential for the distributed manufacturing, is also considered. This comprehensive IPPS problem has not been considered yet within the IPPS field. This IPPS is much more complex than the existing one, so the solution space is larger, implementation is hard, and computation time is longer. We overcome the obstacles by proposing an evolving-based ACO, which we call the evolving ant colony system (EACS). The main procedure of EACS is similar to that of conventional ACO. However, as with conventional ACOs, not every ant agent is replaced on every iteration. In EACS, some ant agents improve their solution route using the proposed greedy heuristics and the rest search for a new route according to the route construction rule using pheromone trail and desirability. For the next generation, some of superior ant agents are replaced by the current ant agents with low performance. The routes of the iteration best solution in the colony and the general best solution are accumulated in the pheromone repository at every iteration in different amounts. Through these individual and cooperative evolving procedures, high quality solutions can be efficiently found for the comprehensive IPPS.

This paper is organized as follows. Section 2 defines the comprehensive IPPS problem we are dealing with, shows how to calculate sequence-dependent setup times, and describes a conventional ant colony system (ACS) for IPPS. Section 3 intensively investigates previous studies focusing on sequence-dependent setups and ACO in IPPS. Section 4 details the procedures of EACS proposed in this study. Section 5 summarizes the benchmark problems and the experimental environment to verify the performance of EACS. In Sect. 6, we analyze the influence of sequence-dependent setup times and verify the superiority of EACS by comparing its performance with that of recently developed meta-heuristics. Finally, Sect. 7 states the conclusions of this study and suggests directions for future research.

2 Background knowledge for IPPS and ACO

Notations

j	index of job, $j \in \{1, 2, 3, \ldots, n^{JOB}\}$, n^{JOB} is the total number of jobs
o	index of operation of the job j, $o \in \{1, 2, 3, \ldots, n_j^{OP}\}$, n_j^{OP} is the total number of operations of the job j
m	index of machine, $m \in \{1, 2, 3, \ldots, n^M\}$, n^M is the total number of machines
t	index of tool, $t \in \{1, 2, 3, \ldots, n^T\}$, n^T is the total number of tool types
k	index of TAD, $k \in \{-x, x, -y, y, -z, z\}$, $-x, x, -y, y, -z, z$ are directions of tool access
$O_{j,o}$	the operation o of the job j not yet assigned m, t, and k
$O_{j,o}^{m,t,k}$	the $O_{j,o}$ assigned m, t, and k
CA	a set of ant agents
Φ	a set of pheromone trails
c	the current node (operation) where the ant agent is staying or the most recently assigned operation, $c = u$ or v
A	a set of nodes where the ant agent can move from c
d	the destination node, $d \in A$
u	the most recently assigned operation with the same job index with d
v	the most recently assigned operation in the same machine with d
$sct(d)$	the setup change time of the d
$tct(d)$	the tool change time of the d
$pt(d)$	the processing time of the d
$ult(c)$	the unload time of the c
$trt(c)$	the transportation time of the c
$ect(d)$	the expected completion time of the d
$\tau(c, d)$	the pheromone trail on the edge (c, d), $\tau(c, d) \in \Phi$
$\eta(d)$	the desirability of the d
$c_t, c_m, \gamma_t, \gamma_m$	adjusting parameters for calculating penalty.

2.1 IPPS representation

There are three types of representations used to describe IPPS: mathematical programming (Li et al. 2010; Nourali et al. 2012; Shen and Yao 2015), disjunctive graph (Leung et al. 2010), and network representation (Kim et al. 2007; Petrović et al. 2016). Among them, network representation is the most preferred in IPPS studies. It is easy to understand intuitively and has the advantage of representing various flexibilities in a graph. It, however, is lacking in describing the mechanism of ACO because it has only conjunctive edges. Thus, we use a combination of the network representation and the disjunctive graph.

An IPPS can be expressed as a graph as shown in Fig. 1a, i.e., $IPPS = (O, C \cup D)$, where O refers to the set of nodes, and C and D denote the sets of conjunctive and disjunctive edges, respectively. A node represents an operation to be executed at a machine, and an edge represents the relationship between operations. For example, Fig. 1a shows a network representation of an IPPS consisting of two jobs (or parts).

Nodes O consists of a starting node SN, an ending node EN, and several intermediate nodes. A solution of an IPPS is a sequence of the intermediate nodes from SN

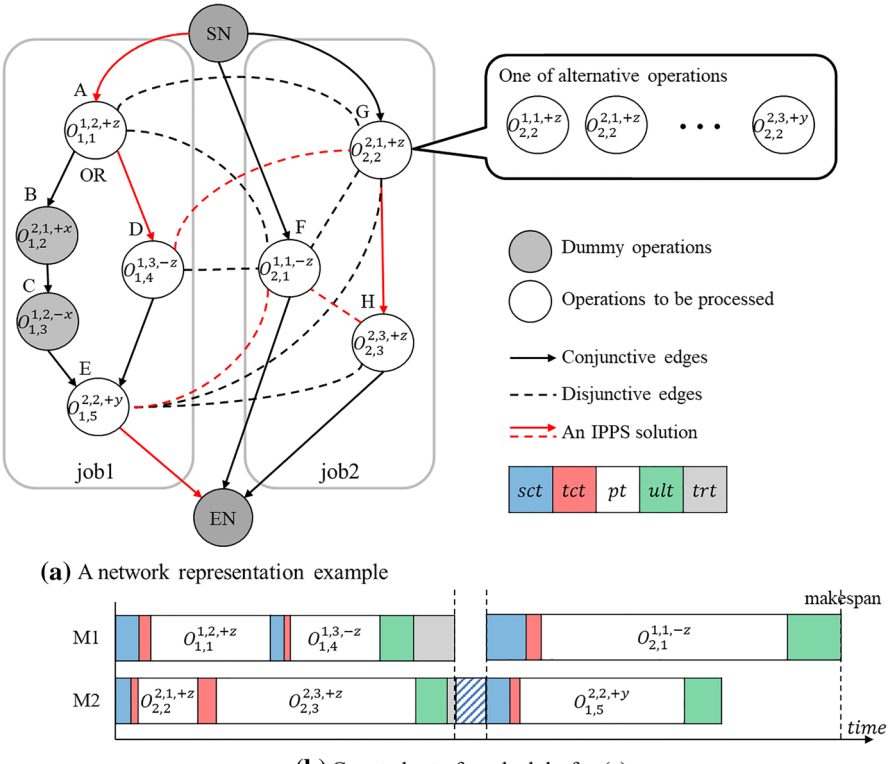

Fig. 1 Simple IPPS example composed of two jobs

to EN. Every intermediate node $O_{j,o}^{m,t,k}$ has attributes of job and operation indices (j, o), machine, tool, and TAD indices (m, t, k) and operation times (sct, tct, pt, ult, trt). An operation $O_{j,o}$ can be performed in the same way by alternative machines (machine flexibility) with alternative types of tools (tool flexibility) under alternative TADs (TAD flexibility). The operation times (sct, tct, pt, ult) depend on the combination of (m, t, tad).

There are two types of edges: conjunctive (solid line with an arrowhead) and disjunctive (dashed line without an arrowhead) edges. Two distinct nodes are connected by a conjunctive edge if there is a precedence relationship between them; otherwise, they are connected by a disjunctive edge. As disjunctive edges are non-directional, various sequences of operations can be constructed as a solution route (sequence flexibility). Every diverging conjunctive sub-route is in an OR or AND relation. The relationship ends at the rejoining node. The OR relation is marked by "OR" at the bottom of the first node, otherwise, it implies an AND relation. The OR relation implies that there are alternative processing routes for processing the same feature (process flexibility). If several sub-routes are in an OR relation, only one of them must be selected and processed. Otherwise (AND relation), all of them must be processed. SN, EN, and all nodes belonging to the unselected sub-routes by the OR relations, are dummy nodes (gray-colored) that are not actually performed. All other nodes (white-colored) should be performed.

IPPS is generally defined under the following assumptions: (1) jobs to be produced are determined prior to optimization, that is, no job is inserted or removed until completion of the predetermined jobs; (2) all machines are ready to work at the starting time of the schedule, i.e., available starting times of all machines are zero; (3) all raw materials are always available; (4) no interruption is allowable during an operation; (5) assigned machine, tool, and TAD do not change during an operation; (6) there is no breakdown on machines or tools; (7) the number of slots in the tool magazine of a machine is fixed; and (8) the number of each type of tool is fixed.

Network representation is easy to understand IPPS, but rigorous description of problems and constraints is impossible. To overcome this, we have presented a mathematical programming model for IPPS in Appendix 1. The proposed IPPS cannot maintain linearity due to sequence-dependent setup times and transportation times of the next subsection, so it is mixed integer nonlinear programming (MINLP).

2.2 Setup and transportation times

Setups are non-value added activities that prepare related workers, machines, tools, and drawings to facilitate the operation (Allahverdi 2015). Setups can be classified as follows: sequence-dependent and sequence-independent. If a setup is sequence-dependent, the setup times or costs depends on the machines, tools, and TADs of the preceding operation and the current operation. If a setup is sequence-independent, the setup times or costs are considered as constants. According to the literature surveys by Allahverdi et al. (2008) and Allahverdi (2015), in about 90% of scheduling studies, setups were ignored or were assumed to be sequence-independent. Moreover, they argued that this

situation paradoxically means that separate consideration of sequence-dependent setups can lead to further improvement of solution quality in scheduling.

In most IPPS studies, the setups have also been assumed as sequence-independent and have been ignored in optimization. However, in IPPS, consideration of sequence-dependent setups is important, as such setups are common in FMS (Eren 2010) which is a major application of IPPS. Three types of sequence-dependent setup times have been considered in IPPS (Azab et al. 2009): (1) the setup change time for positioning a workpiece on a machine; (2) the tool change time for changing tool; and (3) the unloading time for removing a workpiece from a machine.

Transportation time is the time taken to move a machined workpiece to another machine for subsequent operation. The amount of the time is determined by the attributes of the workpiece such as weight and volume, the ability of the transport such as speed and capacity, and the distance between the machines. In IPPS, transportation time has also been frequently ignored. However, if the times are large enough or a distributed manufacturing environment is considered, those are not negligible.

In this study, setup times and transportation times are determined through the following procedure. Let d be the current operation to be processed, u be the last operation assigned that belongs to the same job as d, and v be the last operation performed on the machine where d should be performed. Suppose that the amounts of setup change time, tool change time, and unloading time are determined by only the type of machine, i.e., $SCT(m)$, $TCT(m)$, and $ULT(m)$, respectively. In fact, those sequence-dependent setup times are functions of (m, t, tad) of d, u, and v. However, we adopt the simplified functions in this study because the complicated functions only change the amounts of times. From an optimization approach point of view, it is not significant. The actual sequence-dependent setup times can easily be obtained at the shop floor during execution. In addition, let the amount of transportation time depend on only the distance between two machines and its size as $TRT(m_1, m_2)$.

Under these assumptions, setup times and transportation times are calculated by Eqs. (1)–(4), where '∧' and '∨' are logical operators representing 'and' and 'or', respectively.

$$sct(d) = \begin{cases} SCT(m_d), & if\ (m_u \neq m_d) \vee (j_v \neq j_d) \vee ((m_u = m_d) \wedge (j_v = j_d) \wedge (tad_v \neq tad_d)) \\ 0, & o.w. \end{cases} \tag{1}$$

$$tct(d) = \begin{cases} TCT(t_d), & if\ t_v \neq t_d \\ 0, & o.w. \end{cases} \tag{2}$$

$$ult(u) = \begin{cases} ULT(m_u), & if\ m_u \neq m_d \\ 0, & o.w. \end{cases}, \quad ult(v) = \begin{cases} ULT(m_v), & if\ j_v \neq j_d \\ 0, & o.w. \end{cases} \tag{3}$$

$$trt(u) = \begin{cases} TRT(m_u, m_d), & if\ m_u \neq m_d \\ 0, & o.w. \end{cases} \tag{4}$$

Setup change time $sct(d)$ occurs when the machines of two consecutive operations in a job differ or the jobs or TAD of two consecutive operations in a machine differ.

Tool change time $tct(d)$ occurs when the types of tools of two consecutive operations in a machine differ. Unloading time $ult(u)$ occurs when the machines of two consecutive operations in a job differ and $ult(v)$ occurs when the jobs of two consecutive operations in a machine differ. Transportation time $trt(u)$ occurs when the machines of two consecutive operations in the same job change.

2.3 Conventional ant colony system for IPPS

ACO introduced by Dorigo and Gambardella (1997) is a population-based algorithm that accumulates information about excellent routes searched by a group of ant agents (an ant colony) and reuses the experience for searching new routes. ACO has been evolved into a variety of variants. Those includes the ant system (AS), the ant colony system (ACS), the $\mathcal{MAX}$–$\mathcal{MIN}$ ant system (MMAS), and the rank-based version of the ant system (Stützle and Dorigo 1999). This section describes the mechanism of ACO for IPPS based on ACS, the most popular ACO.

Figure 2 shows the typical procedure of conventional ACS used to minimize the makespan for IPPS, where $s = \left\{ O_{j,o}^{m,t,k} \right\}$ is a sequence of operations (solution); $f(s)$ is the objective function; and τ_0 is an initial pheromone level. At the start of the algorithm, all the pheromone trails are initialized with τ_0. For every iteration, each ant agent in the colony finds a solution route from SN to EN using the route construction rule described below. During the route search, whenever an ant agent determines the next visiting node, the pheromone trails Φ for the edge is locally updated to increase the diversification. If a solution route is constructed, the general best solution is updated depending on the objective value. Once all ant agents complete their route searches, the pheromone trails Φ for the general best solution are updated to increase intensification. These processes are repeated until the termination condition is reached. Then, the general best solution becomes the final solution and the algorithm is stopped.

Algorithm 1 ACS: Main procedure of ACS

Require: $(O, C \cup D)$: IPPS problem;
Ensure: s^{gb}: the general best solution;
1: initialize pheromone trails Φ as τ_0;
2: $s^{gb} \leftarrow \emptyset$;
3: **while** termination condition not met **do**
4: **for** $i \in CA$ **do**
5: construct a solution s_i including local-update of Φ;
6: **if** $f(s_i) < f(s^{gb})$ **then**
7: $s^{gb} \leftarrow s_i$;
8: **end if**
9: **end for**
10: global-update of Φ for s^{gb};
11: **end while**

Fig. 2 Main procedure of typical ACS for IPPS

In the ACS procedure, design factors are a route construction rule and pheromone updating rules. They differ between AS, ACS, and MMSS. The route construction rule is a selection rule for an ant agent to select the next visiting node among the candidates. Suppose that c is a current node (operation), d is a candidate node, and A is a set of candidate nodes that satisfy precedence constraints from c. Let $\tau(c,d) \in \Phi$ denote a magnitude of pheromone deposited on the edge (c,d), and $\eta(c,d)$ denote a magnitude of heuristic desirability for the edge. The destination node d^* is selected by Eq. (5). If a generated random number $rand(\cdot)$ is less than the parameter $p_{ns}\left(0 \le p_{ns} \le 1\right)$, then $\arg\max_{d\in A}\left\{\tau(c,d)^{\alpha}\eta(c,d)^{\beta}\right\}$ becomes d^*, where α and β are design parameters for adjusting the contribution of τ and η, respectively. Otherwise d^* is selected by roulette wheel selection ($\text{roulette}_{\text{wheel}\{\cdot\}}$) with the state transition probability $\frac{\tau(c,d)^{\alpha}\eta(c,d)^{\beta}}{\sum_{d\in A}\tau(c,d)^{\alpha}\eta(c,d)^{\beta}}$. A large value of p_{ns} enhances intensification and a small value enhances diversification.

$$d^* = \begin{cases} \underset{d\in A}{\arg\max}\left\{\tau(c,d)^{\alpha}\eta(c,d)^{\beta}\right\}, & if\ rand(\cdot) < p_{ns} \\ \underset{d\in A}{\text{roulette}_{\text{wheel}}}\left\{\frac{\tau(c,d)^{\alpha}\eta(c,d)^{\beta}}{\sum_{d\in A}\tau(c,d)^{\alpha}\eta(c,d)^{\beta}}\right\}, & o.w. \end{cases} \tag{5}$$

The heuristic desirability $\eta(c,d)$ is an indicator of the degree of preference of the ant agent to the destination node. If the objective is to minimize makespan, the inverse of the processing time of Eq. (6) is generally applied, where Q_{η} is an operating parameter of the ACS.

$$\eta(d) = \frac{Q_{\eta}}{pt(d)} \tag{6}$$

The pheromone updating rule determines how to manage pheromones in pheromone trails. There are two updating rules in ACS: the global update and local update rules. The global update is performed on the general best solution of the ith iteration s_i^{gb} as shown in Eq. (7), where $\tau_i(c,d)$ denotes the amount of pheromone on the edge (c,d) at the ith iteration. First, the increment in the pheromone, $\Delta\tau_i^{gb}$, is calculated by $Q_{\tau}/C_{MAX}\left(s_i^{gb}\right)$, where Q_{τ} is a design parameter for adjusting the amount of pheromone deposition and $C_{MAX}\left(s_i^{gb}\right)$ denotes the makespan of the solution s_i^{gb}. Second, the amount of pheromone of all edges in $C \cup D$ is reduced by $1-\rho_{gu}$ times, where $\rho_{gu}(0 < \rho_{gu} < 1)$ is the evaporation rate. Finally, for only edges belonging to s_i^{gb}, $\Delta\tau_i^{gb}$ is added.

$$\tau_i(c,d) = \begin{cases} \left(1-\rho_{gu}\right)\tau_{i-1}(c,d) + \Delta\tau_i^{gb}, & if\ (c,d) \in s_i^{gb} \\ \left(1-\rho_{gu}\right)\tau_{i-1}(c,d), & if\ (c,d) \notin s_i^{gb}\ and\ (c,d) \in C \cup D \end{cases} \tag{7}$$

The local update adjusts pheromone as shown in Eq. (8). It is controlled by the evaporation rate $\rho_{lu}\left(0 < \rho_{lu} < 1\right)$ for the local update. The local update increases diversification by reducing the pheromone of the visited edges, while the global update increases intensification.

$$\tau_i(c,d) = \left(1 - \rho_{lu}\right)\tau_i(c,d) \tag{8}$$

Let us explain the above ACS procedure in detail using the network representation in Fig. 1a and the Gantt chart in Fig. 1b. Suppose that an ant agent is trying to find a solution route from SN to EN. Due to precedence constraints, the ant agent at SN can only move to nodes {A, F, G}. Assume that the ant agent selects node A according to the route construction rule in Eq. (5). Node A, $O_{1,1}^{1,2,+z}$, must be machined by machine M1 with tool 1 in the +z direction. Since M1 is currently empty and the operation does not have any preceding operations, a setup change time, a tool change time, and a processing time is scheduled to M1 as shown in Fig. 1b. Now, as node A is already scheduled, it is dropped from the candidate set. In Fig. 1a, node A is connected to nodes B and D by conjunctive edges and to nodes F and G by disjunctive edges. Note that Fig. 1a only shows the edges associated with the route search of the ant agent among all disjunctive edges for the sake of simplicity. Hence, nodes B, D, F, and G become candidates for the next visiting nodes and those are merged into the current candidate set {F, G}. Finally, the candidate set becomes {F, G, B, D}.

Assume that the ant agent selects node D from {F, G, B, D} as the next node according to the route construction rule. Node D is in an OR relationship with route B–C, so nodes B and C are deactivated as dummy nodes. Node D, $O_{1,4}^{1,3,-z}$, must be processed on the same machine as the previous operation $O_{1,1}^{1,2,+z}$, but the TAD and tool are changed. Therefore, the workpiece does not need to be unloaded after $O_{1,1}^{1,2,+z}$ is completed, but the operation can only start after exhausting the setup change time and tool change time. After scheduling of node D, it is dropped from the candidates and new candidate nodes are searched by precedence relations. As a result, the candidate set becomes {F, G, E} by insertion of node E. Assume that the ant agent has finally arrived at node E. Node E, $O_{1,5}^{2,2,+y}$, must be machined by M2. The operation can start when the workpiece of job1 arrives at M2 after completion of $O_{1,4}^{1,3,-z}$ in M1. In this case, the setup change time schedule of $O_{1,5}^{2,2,+y}$ overlaps with the transportation time schedule of the workpiece of job2 (shaded area). However, it does not affect to the starting time of $O_{1,5}^{2,2,+y}$ because M2 is empty. Finally, the ant agent arrives at EN. Then, the final solution route (red lines) is determined as A–D–G–H–F–E and the makespan of the solution is determined as the latest completion time of all operations.

3 Literature review

3.1 Setup consideration on IPPS

Although numerous studies have been conducted for IPPS over the past two decades, few have explored the consideration of setups. Moon et al. (2002) proposed a GA for IPPS with sequence-dependent setup times and transportation times on a multi-plant supply chain environment. However, they did not handle the setup change time, tool change time, and unloading time separately. Li and McMahon (2007) proposed a simulated annealing approach to optimize makespan, utilization, tardiness, and

cost for multi-objective IPPS, in which the sequence-dependent setup change time and tool change time are considered. Guo et al. (2009) proposed a PSO to re-plan multi-objective IPPS for occurrence of machine breakdown and new order arrival, in which a united sequence-dependent setup time is considered. Wan et al. (2011) proposed an ACO for IPPS. They argued that the sequence-dependent setup times should be handled separately to obtain an efficient schedule because the setup times cannot be ignored if two immediately preceding jobs are sequence-dependent. They considered the tool change time, loading (setup change) time, and unloading time separately. Furthermore, they verified that separate consideration of the sequence-dependent setup times can improve the makespan using an example proposed by Li et al. (2002) and Li and McMahon (2007). However, their research lacks mathematical descriptions of applying the sequence-dependent setup times and extensive experiments for various problems. Nourali et al. (2012) proposed a mixed integer linear programming model for IPPS considering a unified sequence-dependent setup time. Srinivas et al. (2012) proposed an approach combining ACO and PSO to minimize total manufacturing cost for IPPS, in which a sequence-dependent tool change cost and a setup change cost is considered. Recently, three hybrid PSOs (Petrović et al. 2016; Miljković and Petrović 2017; Dou et al. 2018) were proposed to minimize total manufacturing cost and total weighted production time for IPPS with sequence-dependent setups, in which a tool change time/cost, setup change time/cost, and transportation time/cost are dealt with. However, unloading time/cost is not considered. PSO combined with chaos theory (Petrović et al. 2016), PSO combined with GA (Miljković and Petrović 2017), and discrete PSO combined with GA (Dou et al. 2018) are proposed to overcome premature convergence of PSO.

All the above studies deal with sequence-dependent setup times and/or transportation time in IPPS. However, most of them do not handle various sequence-dependent setup times separately. In addition, most of the studies ignore some of the various sequence-dependent setup times, do not provide clear mathematical descriptions, or lack experiments. In this study, we have considered all existing sequence-dependent setup times and transportation time separately.

3.2 Ant colony optimization for IPPS

ACO focused studies in IPPS are as follows. Leung et al. (2010) proposed ACS with an elite strategy, but did not consider TAD flexibility, setups, and tool capacity. Srinivas et al. (2012) argued that the two planning activities conflict with objectives; process planning is meant to satisfy technological requirements and scheduling is meant to optimize timing aspects. As an approach to solve the conflict, they proposed a sequential hybrid algorithm in which ACO finds an optimal process planning and PSO finds an optimal schedule under the optimal process plan. However, they did not provide sufficient experimental results except only a few selected small-sized examples. Wang et al. (2014) proposed an improved AS for IPPS. To overcome the weakness of ACO of the local convergence, they proposed a method to reduce extraordinary accumulation of pheromones and a local pheromone updating rule that allows repeated accumulation of pheromones to avoid stagnation. Zhang

and Wong (2014) proposed an ACO approach to retain two distinct pheromone trails on nodes and edges, respectively. To improve the performance of their approach, they also adopted the elitist strategy, the $\mathcal{MAX}$–$\mathcal{MIN}$ strategy, and the desirability function as the inverse of the increment of make span. Liu et al. (2016) introduced a mathematical programming model for IPPS and proposed a typical ACO to solve the problem. Zhang and Wong (2016) proposed a constructive meta-heuristic based on ACO for IPPS. To improve the solution quality, during operation selection, they adopted the earliest finishing time rule and time-window based mapping to reduce idling time. However, all three studies (Liu and MacCarthy 1997; Zhang and Wong 2014, 2016) did not consider TAD flexibility, sequence-dependent setups, and tool capacity constraints.

Most existing IPPS studies applying ACO as a main solver have omitted to consider sequence-dependent setups, TAD flexibility, and tool capacity constraints. The tool capacity constraints were discussed only by Kim et al. (2007). They, however, applied an asymmetric multileveled symbiotic evolutionary algorithm rather than ACO as an optimization algorithm, and ignored sequence-dependent setups and TAD flexibility. Srinivas et al. (2012) considered TAD flexibility and sequence-dependent setups, but they also ignored tool capacity constraints and provided experimental results for only a small-sized problem example. We believe that typical ACOs are limited in their ability to cover the various flexibilities and high complexity that IPPS has, and as an alternative to this, we suggest the evolving ant colony system in Sect. 4.2.

4 Proposed evolving ant colony system (EACS)

4.1 Issues of the conventional ant colony system

The stigmergic property of ACO is useful for improving solution quality, but it also causes a local convergence and/or a stagnation (Wang et al. 2014). Zhang and Wong (2013, 2014) pointed out three drawbacks in applying typical ACO to IPPS: (1) high likelihood of local convergence owing to a greedy strategy based on the shortest processing time when selecting the next node, (2) frequent reset of the algorithm and premature convergence due to excessive evaporation of pheromone trails, and (3) inefficient improvement of solution quality by adopting the makespan that is indistinct in value as an objective function. We fully agree with their claims; in particular, the evaporation issue is the most serious for IPPS.

Due to various flexibilities of IPPS, an operation can have many alternative operations depending on the machine, tool, or TAD to be processed. Since each alternative operation generates a distinct schedule, it is regarded as an independent operation in IPPS. Thus, as IPPS allows more flexibility, the number of nodes and edges increases sharply, and consequently, the size of the pheromone repository increases. For example, suppose that there are 20 jobs (parts) to be processed in total, each job consists of 20 operations, and the number of alternatives for machine, tool, and TAD for each operation is 3, 3, and 3, respectively. Then, the total number of independent operations is 10,800 ($=20\times20\times3\times3\times3$); therefore, the total number of edges or the size of pheromone repository becomes $10{,}800\times10{,}799$. When the

pheromone trails are updated globally, the evaporation process should be performed at every edge. The update requires a large calculation, which slows down the algorithm. However, the deposit process is performed only on the edges belonging to the general best solution at every iteration. It is only quite a small fraction of the total number of edges in pheromone trails. If the evaporation rate is not large enough, it implies that the total amount of deposited pheromone is smaller than the total amount of evaporated pheromone in the trails. As the iteration runs, the total amount of pheromone in trails decreases, and the pheromone trails become disabled. If the evaporation rate is set too large to avoid the disability, the relatively large pheromone on some edges causes a premature convergence. The magnitude of the effect depends on the number of edges in IPPS and the objective values of intermediate solutions. It implies that ACO is too sensitive to problems. Zhang and Wong (2013, 2014) attempted to solve these drawbacks of ACO for IPPS by various add-ons and a complicated procedure. We will tackle these using a simpler approach.

4.2 Overall procedure of EACS

In IPPS, all alternatives of an operation should be handled as independent nodes. Nevertheless, only a node among the alternative nodes is selected in a final solution route. In other words, almost all nodes in the IPPS network are redundant. Compared to the TSP where all nodes (cities) are included in the route, this situation leads to ineffective maintenance of pheromone trails. To overcome it, Zhang and Wong (2014) managed the pheromone trails for only operation $O_{j,o}$ and the alternative machines of the operation were determined by the dispatching rule during ACO execution. Since this approach does not generate redundant nodes, the size of pheromone trails can be kept constant number regardless of the number of alternative machines. This approach, however, does not retain all information for the best solution completely because the pheromone trails only accumulate experience for $O_{j,o}$ other than $O_{j,o}^{m,t,k}$. To compensate for the lack of information, they operated two distinct pheromone trails: edge-based trails for determining a sequence of operations and node-based trails for determining the resources (m, t, k) of an operation. However, the performance of their approach is doubtful because the effect of the resources on the schedule of an operation highly depends on the sequence of operations.

We, therefore, suggest an approach that accumulates the experience of sequences in the pheromone trails while the experience for resources stores in each ant agent. This approach not only allows efficient management of pheromone trails by disregarding redundant nodes, but also improves the solution quality by utilizing the experience for resources. In addition, it has the advantage of saving computation time by recycling the resource information included in the excellent solution routes. We will call the proposed approach the evolving ant colony system (EACS). A typical ACS, for each iteration, regenerates a certain number of ant agents, searches for new solution routes, stores information of the general best solution in the pheromone trails, and then destroys the ant agents. On the other hand, the proposed EACS conserves some ant agents that provide excellent solution for every iteration as

Algorithm 2 EACS: Main procedure of EACS

Require: $(O, C \cup D)$: IPPS problem;
Ensure: s^{gb}: the general best solution;
1: initialize pheromone trails Φ as τ_0;
2: construct $\{s_i, \forall i \in CA\}$ with randomly assigned m, t, and k;
3: $s^{gb} \leftarrow \arg\max_{s_i, i \in CA}\{f(s_i)\}$;
4: **for** $n = 1 : n_{iter}$ **do**
5: **for** $i \in CA$ **do**
6: construct $\hat{s}_i$ by route construction rule or improve s_i by tuning;
7: **end for**
8: replace $p_{ns} \times n_{CA}$ inferior solutions in $\{s_i\}$ as the superior solutions in $\{\hat{s}_i\}$;
9: $s^{ib} \leftarrow \arg\max_{s_i, i \in CA}\{f(s_i)\}$;
10: local-update of Φ for s^{ib};
11: **if** $f(s^{lb}) > f(s^{gb})$ **then**
12: $s^{gb} \leftarrow s^{lb}$;
13: **end if**
14: global-update of Φ for s^{gb};
15: **end for**

Fig. 3 Pseudo-code for the main procedure of EACS

Algorithm 3 CONSTRUCT: Solution construction procedure of EACS

Require: $s = \{O_{j,o}^{m,t,k}\}$: a sequence of operations;
Ensure: $\hat{s}$: a solution induced by s;
1: **if** $rand(\cdot) < p_{tune}$ **then**
2: $\hat{s} \leftarrow s$;
3: run **OC** for $\hat{s}$
4: run **MTTC** for $\hat{s}$
5: run **TUNING** for $\hat{s}$
6: **else**
7: construct a new solution $\hat{s}$ using the route construction rule;
8: **end if**

Fig. 4 Pseudo-code for the solution construction procedure of EACS

shown in Fig. 3, where $s = \left\{ O_{j,o}^{m,t,k} \right\}$ is a sequence of operations (solution); $f(s)$ is the objective function; τ_0 is an initial value of a pheromone tail; n_{iter} is the number of iteration, and n_{CA} is the size of CA. Then, it improves the current solutions or searches for new solutions as shown in Fig. 4. Finally, it replaces a number of ant agents according to their performance and stores information of the iteration best and the general best solution in the pheromone trails.

4.3 Route construction rule of EACS

The route construction rule of EACS is similar to that of ACS but differs from the heuristic desirability function and standardization of pheromone and desirability. Zhang and Wong (2014) demonstrated that applying the earliest completion time rule rather than the typical shortest processing time rule to the

desirability function was more superior for improving the solution quality. After performing the preliminary experiments, we also reached the same conclusion.

EACS applies the desirability function $\eta(d)$ of Eq. (9) that is the difference of the expected completion time $ect(d)$ from the maximum among the candidates, i.e., $\max_{d\in A}\{ect(d)\}$. In the equation, one is applied to avoid zero division error. The $ect(d)$ can be calculated using Eqs. (1)–(4) and (10), where notation u and v are the same as described above. Transportation time should be applied with caution. If a workpiece on the move is out of machine, so it only affects the starting time of subsequent operations belonging to the same job, the first term of Eq. (10) represents the earliest starting time of d. Unlike ACS, EACS does not use correction parameters such as Q_η in Eq. (6) in the desirability calculation because the magnitude of η is adjusted by the standardization described below.

$$\eta(c,d) = \max_{d\in A}\{ect(d)\} - ect(d) + 1 \tag{9}$$

$$ect(d) = \max\{ult(u) + trt(u), ult(v)\} + sct(d) + tct(d) + pt(d) \tag{10}$$

For all candidate nodes $d \in A$, one of them is selected as the next node d^* to move. The node selection procedure of EACS is the same as Eq. (5) of ACS as shown in Eq. (11). For a random number $rand(\cdot)$, if $rand(\cdot) \le p_{ns}$, the node with the maximum value of $\widetilde{\tau(c,d)}^{\alpha}\,\widetilde{\eta(c,d)}^{\beta}$ among $d \in A$ is selected as the next node to increase intensification; otherwise, a node is determined by the roulette wheel selection to increase the diversity.

$$d^* = \begin{cases} \arg\max\limits_{\hat{d}\in A}\left\{\widetilde{\tau(c,d)}^{\alpha}\,\widetilde{\eta(c,d)}^{\beta}\right\}, & if\ rand(\cdot) \le p_{ns} \\ \text{roulette}_{\text{wheel}}\limits_{\hat{d}\in A}\left\{\dfrac{\widetilde{\tau(c,d)}^{\alpha}\,\widetilde{\eta(c,d)}^{\beta}}{\sum_{\hat{d}\in A}\widetilde{\tau(c,d)}^{\alpha}\,\widetilde{\eta(c,d)}^{\beta}}\right\}, & o.w. \end{cases} \tag{11}$$

The difference between this approach and ACS is that $\tau(c,d)$ and $\eta(c,d)$ are not used directly but are standardized as $\widetilde{\tau(c,d)} = \frac{\tau(c,d)}{\sum_{d\in A}\tau(c,d)}$ and $\widetilde{\eta(c,d)} = \frac{\eta(c,d)}{\sum_{d\in A}\eta(c,d)}$, respectively. In fact, $\tau(c,d)$ and $\eta(c,d)$ depend on the solution route until node c, the order of c in the route, the processing times and setup times of operations, and the assigned machines, tools, and TADs of c, d, u, and v. Conventional ACS compensates for the variation by using the adjusting parameters Q_τ and Q_η. However, the effect is questionable because there are no absolute criteria for the magnitudes of $\tau(c,d)$ and $\eta(c,d)$. The unstable magnitudes of (c,d) and $\eta(c,d)$ may also make it difficult to determine the parameters α and β, which are the degrees of contribution of the pheromone trails and desirability, respectively. If the magnitudes of $\tau(c,d)$ and $\eta(c,d)$ vary from IPPS to IPPS and from iteration to iteration, then determining the appropriate α and β becomes another optimization problem. Our proposed standardization approach can maintain $\widetilde{\tau(c,d)}$ and $\widetilde{\eta(c,d)}$ within [0,1], so it is a good way to overcome this issue.

Algorithm 4 OC: Order change procedure of EACS

Require: $(O, C \cup D)$: IPPS problem;
$s = \{O_{j,o}^{m,t,k}\}$: a sequence of operations;
select an operation $O_{j,o}^{m,t,k} \in s$ arbitrarily ;
find $O_{j,p}^{m',t',k'} \in s$ that must precede $O_{j,o}^{m,t,k}$ immediately in $(O, C \cup D)$;
find $O_{j,s}^{m'',t'',k''} \in s$ that must succeed $O_{j,o}^{m,t,k}$ immediately in $(O, C \cup D)$;
move $O_{j,o}^{m,t,k}$ between $O_{j,p}^{m',t',k'}$ and $O_{j,s}^{m'',t'',k''}$ arbitrarily;

Fig. 5 Pseudo-code for the order change heuristic of EACS

Algorithm 5 MTTC: Machine, tool, and TAD change procedure of EACS

Require: $s = \{O_{j,o}^{m,t,k}\}$: a sequence of operations;
for $O_{j,o}^{m,t,k} \in s$ **do**
  if $rand(\cdot) < p_{mttc}$ **then**
    find (m', t', k') that induces the minimum processing time of $O_{j,o}$
    among alternatives;
    $O_{j,o}^{m,t,k} \leftarrow O_{j,o}^{m',t',k'}$;
  end if
end for

Fig. 6 Pseudo-code for the machine, tool, and TAD change heuristic of EACS

4.4 Solution improving heuristics

Through preliminary tests, we found that the following three factors reduce the performance of ACO for IPPS; (1) premature convergence due to lack of diversification, (2) slow improvement of solution quality due to various flexibilities, and (3) infeasibility of solutions due to capacity constraints in large sized problems. To overcome those, we introduce three greedy heuristics: **OC**, **MTTC**, and **TUNING**. The detailed procedures are expressed as pseudo code in Figs. 5, 6 and 7, respectively, where n_s is the size of solution s; O_i is the ith operation of solution s; $O_{j,o_u}^{m_u,t_u,k_u}$ is the operation that precede $O_{j,o}^{m,t,k}$ immediately in the job j; and O_{j_v,o_v}^{m,t_v,k_v} is the operation that precede $O_{j,o}^{m,t,k}$ immediately on the machine m.

In ACO, ant agents tend to follow the most popular solution route due to pheromone trails. **OC** increases the diversity of the solution route by artificially altering the sequence while maintaining precedence relations between operations. **MTTC** is a heuristic that randomly selects some operations, and then changes the machine, tool, and TAD of minimal processing time among alternatives. It is effective in improving the solution quality by allowing ant agents to retain excellent combinations of resources. **TUNING** coincides resources of successor to ones of predecessor. This procedure not only increases the feasibility of the solution by increasing reuse of resources, but also reduces sequence-dependent setup times.

Algorithm 6 TUNING: Machine, tool, and TAD tuning procedure of EACS

Require: $s = \{O_{j,o}^{m,t,k}\}$: a sequence of operations;
1: select two integers f and l ($f < l$) arbitrarily in $[1, n_s]$
2: **for** $O_{j,o}^{m,t,k} \in \{O_f, \cdots, O_l\}$ **do**
3: find $O_{j,o_u}^{m_u,t_u,k_u} \in s$ and $O_{j_v,o_v}^{m,t_v,k_v} \in s$
4: **if** $O_{j,o}$ can be performed on machine m_u **then**
5: $O_{j,o}^{m,t,k} \leftarrow O_{j,o}^{m_u,t,k}$;
6: **end if**
7: **if** $O_{j,o}$ can be performed on machine m using t_v **then**
8: $O_{j,o}^{m,t,k} \leftarrow O_{j,o}^{m,t_v,k}$;
9: **end if**
10: **if** $O_{j,o}$ can be performed on machine m using t on TAD k_v **then**
11: $O_{j,o}^{m,t,k} \leftarrow O_{j,o}^{m,t,k_v}$;
12: **end if**
13: **end for**

Fig. 7 Pseudo-code for the machine, tool, and TAD tuning heuristic of EACS

4.5 Pheromone update rule of EACS

As discussed in Sect. 4.1, the biggest issue in applying ACO to IPPS is excessive evaporation of pheromone and the computation time required for pheromone update. To overcome it, we propose a simple approach that increases the pheromone by a constant without allowing evaporation. As the proposed updating rule only increases a fixed amount of pheromone, it may cause performance degradation of the algorithm due to unbalance in the magnitudes of $\tau(c,d)$ and $\eta(c,d)$ when applying the route construction rule. However, the standardization approach avoids such an imbalance.

The pheromone updating function of EACS is presented in Eq. (12), where the constant parameter $\Delta\tau$ is a constant incremental amount of the pheromone. The local update ($\Delta\tau_{local}$) and global update ($\Delta\tau_{global}$) are the same except for the amount of $\Delta\tau$. This simple updating function saves much computational effort because it updates only a few edges belonging to one solution route, rather than entire pheromone trails.

$$\tau_i(c,d) = \max\left\{\tau_{max}, \tau_{i-1}(c,d) + \Delta\tau\right\}, \quad if (c,d) \in s. \tag{12}$$

4.6 Objective function

It is reasonable to use the reciprocal of makespan $C_{MAX}(s)$ as an objective function if the objective of IPPS is to minimize makespan. However, the IPPS under consideration in this study has several constraints such as magazine capacity and tool capacity, so it is appropriate to add penalties for infeasible solution as in Kim et al. (2007). Equation (13) represents the penalty imposed on the objective function.

$$f(s) = \frac{100,000}{C_{max}(s) + c_t \sum_{\forall t} NST(t)^{\gamma_t} + c_m \sum_{\forall m} NSS(m)^{\gamma_m}} \tag{13}$$

Here, $C_{max}(s)$ represents the makespan for a solution route s, $NST(t)$ denotes the number of shortage of tools for the tool t, i.e., $NST(t) = \max\left\{n_t^{TOOL} - C_t^{TOOL}, 0\right\}$,

where n_t^{TOOL} is the number of the tool t used and C_t^{TOOL} is the capacity of the tool t, and $NSS(m)$ is the number of shortage of slots in the tool magazine of the machine m, i.e., $NSS(m) = \max\left\{n_m^{SLOT} - C_m^{SLOT}, 0\right\}$, where n_m^{SLOT} is the number of slots to equip tools on the machine m and C_m^{slot} is the capacity of tool slots on the machine m. c_t, c_m, γ_t and γ_m are the adjusting parameters that determine the influence of the penalty on the objective function.

5 Experiment environment

5.1 Benchmark problem sets

There is no exact existing benchmark problem for the IPPS considered in this study, so we constructed 31 benchmark problems by modifying those of Kim et al. (2007), who used an IPPS that is most similar to our IPPS. Table 1 summarizes the identification of each problem, the list of jobs involved, and the total number of operations involved in the problem. The problem set is divided into small-sized problems (P1–P12), medium-sized problems (P13–P20), large-sized problems (P21–P28) and very-large-sized problems (P29–P31) depending on the number of jobs involved. For a detailed network representation, see Kim et al. (2007).

Table 2 summarizes the information related to each job, machine, and tool. The job column provides information on the number of operations (#O) and the number of OR relations (#OR) included in each job. The machine column summarizes the capacity of slots on each machine. The tool column lists the capacity of each tool and the number of required slots for the tool t, i.e., r_t^{SLOT}. The sequence-dependent setup times $SCT(m)$, $ULT(m)$, and $TCT(t)$ and transportation time $TRT\left(m_1, m_2\right)$ used in the experiments are also summarized in Table 3.

5.2 Meta-heuristics and experiment environment for comparative study

We compare the performance of the proposed EACS with the three different meta-heuristics proposed for IPPS: the modified PSO (mPSO) by Miljković and Petrović (2017), the feasible sequence oriented discrete PSO (FSDPSO) by Dou et al. (2018), and enhanced ACO (E-ACO) by Zhang and Wong (2014). Since all three meta-heuristics have been proposed recently and the operating mechanisms are different, we believe that reasonable comparisons are possible. Among those, the most recently proposed FSDPSO was developed to overcome the difficulty of applying conventional continuous PSOs such as mPSO to discrete combinatorial problems such as IPPS. FSDPSO supports to discrete problems by updating particles using crossover and mutation operators of GA while maintaining the basic mechanism of PSO which improves particle swarm by correlation of current solution, iteration best solution and global best solution. Since the IPPSs applied by the methods are not the same as the IPPS we considered, we have slightly modified those to handle our problem. The algorithms of EACSNS and EACS are the same. The difference is that EACSNS

Table 1 Basic information on benchmark problems

Problem	Jobs	Total number of operations
P1	1, 2, 3, 10, 11, 12	89
P2	4, 5, 6, 13, 14, 15	100
P3	7, 8, 9, 16, 17, 18	121
P4	1, 4, 7, 10, 13, 16	99
P5	2, 5, 8, 11, 14, 17	102
P6	3, 6, 9, 12, 15, 18	109
P7	1, 4, 8, 12, 15, 17	103
P8	2, 6, 7, 10, 14, 18	96
P9	3, 5, 9, 11, 13, 16	111
P10	4, 5, 6, 10, 11, 12	105
P11	7, 8, 9, 13, 14, 15	76
P12	1, 2, 3, 16, 17, 18	105
P13	1, 2, 3, 5, 6, 10, 11, 12, 15	142
P14	4, 7, 8, 9, 13, 14, 16, 17, 18	168
P15	1, 4, 5, 7, 8, 10, 13, 14, 16	150
P16	2, 3, 6, 9, 11, 12, 15, 17, 18	160
P17	1, 2, 4, 7, 8, 12, 15, 17, 18	155
P18	3, 5, 6, 9, 10, 11, 13, 14, 16	155
P19	4, 5, 6, 7, 8, 9, 10, 11, 12	159
P20	1, 2, 3, 13, 14, 15, 16, 17, 18	151
P21	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15	189
P22	4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18	221
P23	1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17	201
P24	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18	211
P25	1, 2, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18	199
P26	2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18	207
P27	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	205
P28	1, 2, 3, 7, 8, 9, 13, 14, 15, 16, 17, 18	212
P29	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18	262
P30	1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18	266
P31	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	310

treats all setup times as sequence-independent, so $sct(\cdot)$, $tct(\cdot)$, and $ult(\cdot)$ are always applied. Transportation time is applied to all algorithms in common.

Table 4 summarizes the parameters applied to the methods and EACS. Operational parameters of the comparative methods are assigned as the recommended values in each paper (Zhang and Wong 2014; Miljković and Petrović 2017; Dou et al. 2018), where n_{iter} is the number of iterations; n_{CA} is the size of colony or swarm; n_{repeat} is the number of repetitions of the experiments; W_{min} and W_{min} are the initial and the final values of weighting parameters, respectively; C_1 and C_2

Table 2 Information related to each job, machine, and tool

Job		Machine			Tool		
J	#O	#OR	m	C_m^{SLOT}	t	C_t^{TOOL}	r_t^{SLOT}
1	12	2	1	28	1	6	1
2	14	1	2	31	2	7	2
3	19	2	3	38	3	6	2
4	16	2	4	28	4	6	2
5	18	2	5	36	5	10	1
6	20	2	6	37	6	6	2
7	21	4	7	29	7	10	3
8	20	4	8	28	8	9	2
9	20	3	9	29	9	7	1
10	11	1	10	26	10	5	2
11	15	2			11	7	1
12	18	2			12	10	2
13	18	3			13	9	3
14	13	2			14	8	2
15	15	2			15	6	2
16	21	3			16	10	2
17	22	4			17	9	1
18	17	3			18	6	2
					19	8	2
					20	7	3

Table 3 Information on setup times and transportation time

m_1	m_2											
	$SCT(m)$	$ULT(m)$	$TCT(m)$	$TRT(m_1, m_2)$								
				1	2	3	4	5	6	7	8	9	10
1	12	12	5	0	4	10	18	10	8	14	18	16	22
2	12	12	4	4	0	6	14	6	4	10	14	12	18
3	10	10	4	10	6	0	8	8	6	4	8	10	12
4	8	8	3	18	14	8	0	8	10	4	8	10	4
5	8	8	3	10	6	8	8	0	2	12	16	18	12
6	12	12	5	8	4	6	10	2	0	10	14	16	14
7	12	12	4	14	10	4	4	12	10	0	4	6	8
8	10	10	4	18	14	8	8	16	14	4	0	2	4
9	8	8	3	16	12	10	10	18	16	6	2	0	6
10	8	8	3	22	18	12	4	12	14	8	4	6	0

Table 4 Values of the parameters used in the meta-heuristics for comparative study

Common		mPSO		FSDPSO		E-ACO		EACSNS & EACS	
Param.	Values	Param.	Values	Param.	Values	Param.	Values	Param.	Values
n_{iter}	2000	W_{min}	0.4	ω	1.0	Q, C	300, 150	$\Delta\tau_{local}$	0.005
n_{CA}	30	W_{max}	1.2	C_1	2.0	D	15	$\Delta\tau_{global}$	0.01
n_{repeat}	50	C_1	2.0	C_2	2.0	τ_{min}, τ_{max}	1.0, 20.0	τ_{max}	20.0
c_t	200	C_2	2.0	k_1	0.5	τ_0	10	τ_0	10
c_m	200	$p_{crossover}$	0.6	k_2	0.005	α, β	1, 2	α, β	1, 2
γ_t	2	$p_{mutation}$	0.1	p_m	0.1	ρ_{gu}, ρ_{lu}	0.15, 0.15	p_{ns}	0.8
γ_m	2	p_{shift}	0.1			p_{ns}	0.9	p_{tune}	0.8
								p_{mttc}	0.1

are self-recognition and social component acceleration coefficients, respectively; $p_{crossover}$, $p_{mutation}$, and p_{shift} are selection probabilities for GA operations adopted at mPSO; ω is an inertia weight for velocity control, p_m is fragment mutation probability; k_1 is scale factor for adjusting p_m; k_2 is the basic value of p_m; Q and D are scale factors for pheromone calculation; C is a scale factor for heuristic desirability calculation; τ_{min} and τ_{max} are the lower and the upper bound of pheromone level, respectively; τ_0 is an initial value of pheromone level; α and β are respective weights of pheromone trail and heuristic desirability, respectively; ρ_{gu} and ρ_{lu} are evaporation rates for the global update and the local update, respectively; and p_{ns} is node selection probability. The determination of parameters for EACS will be dealt with in detail in the next subsection. All algorithms were implemented using the Julia language Version 1.0 and the experiments were performed on an Intel® Core™ i7-6700 CPU @3.40 GHz with 16.0 GB.

6 Experimental results

6.1 Parameter determination

A full factorial experiment was performed to determine the optimal operating parameters of EACS. We applied 3^3 factorial design with 27 treatments with 3 levels for p_{tune}, p_{mttc}, and p_{ns}, respectively. Other parameters were determined by preliminary tests. Observations were made on makespan C_{max} and computation time t_{comp} for P1, P13, P21, P29, and P31, which are representative problems of each size level. Table 5 summarizes the results of this experiment. The value of each cell represents an average value of C_{max} or t_{comp} by repeating 20 times. In the $\overline{C_{max}}$ column, the sum of $\overline{C_{max}}$s of all problems (sum) and the performance rank (rank) based on it are added separately.

Table 6 shows the results of an analysis of variance (ANOVA) to test whether the parameters and their interactions affect the sum of $\overline{C_{max}}$ s. All three parameters were

Table 5 $\overline{C_{max}}$ and $\overline{t_{comp}}$ according to p_{tune}, p_{mttc}, and p_{ns} for the selected IPPS benchmark problems

p_{tune}	p_{mttc}	p_{ns}	$\overline{C_{max}}$							$\overline{t_{comp}}$				
			P1	P13	P21	P29	P31	Sum	Rank	P1	P13	P21	P29	P31
0.7	0.05	0.6	475.7	570.3	696.9	936.0	1092.9	3771.6	22	17.1	30.8	44.6	69.4	86.1
		0.7	476.7	558.8	687.2	919.7	1100.6	3742.9	18	16.8	30.0	43.9	68.5	85.5
		0.8	478.6	536.6	680.9	898.9	1091.7	3686.6	10	16.6	29.8	43.4	67.8	83.9
	0.1	0.6	480.7	551.4	688.8	920.8	1107.3	3748.9	19	17.2	31.0	45.2	70.1	87.6
		0.7	463.2	540.4	678.2	902.6	1092.7	3676.9	6	17.2	30.7	44.8	69.8	86.6
		0.8	469.5	545.0	669.3	898.0	1077.8	3659.5	3	16.9	30.3	44.2	69.1	85.8
	0.15	0.6	473.9	554.1	677.7	913.1	1113.6	3732.2	16	17.7	31.6	45.9	71.2	88.0
		0.7	465.0	544.9	670.4	900.0	1078.5	3658.7	2	17.4	31.0	45.3	70.2	87.0
		0.8	475.6	535.3	660.2	900.2	1089.6	3660.9	4	17.2	30.8	44.7	69.6	86.3
0.8	0.05	0.6	472.7	565.1	692.2	924.4	1133.8	3788.2	25	12.9	21.8	31.9	50.1	62.6
		0.7	480.6	567.4	689.8	919.7	1122.6	3780.0	23	12.7	21.5	31.4	49.6	62.0
		0.8	480.1	554.5	666.1	915.4	1066.2	3682.2	7	12.6	21.2	30.9	48.9	61.3
	0.1	0.6	478.8	552.4	674.6	920.6	1091.1	3717.4	13	13.1	22.2	32.4	50.8	63.3
		0.7	474.5	551.6	676.5	913.8	1090.0	3706.4	12	12.9	21.8	31.8	50.4	62.8
		0.8	***471.7***	***540.5***	***667.4***	***899.3***	***1052.8***	***3631.5***	***1***	12.8	21.5	31.6	49.7	62.0
	0.15	0.6	477.6	552.9	673.7	918.2	1132.4	3754.7	21	13.3	22.6	32.9	51.3	64.1
		0.7	477.0	537.4	663.1	900.2	1083.4	3661.1	5	13.2	22.2	32.4	51.0	63.2
		0.8	474.5	541.4	660.9	908.3	1098.0	3683.1	8	13.0	21.9	31.8	50.5	62.8
0.9	0.05	0.6	481.4	581.1	687.4	945.0	1111.5	3806.3	26	8.2	13.4	18.5	28.6	36.0
		0.7	479.7	567.1	681.0	916.5	1108.5	3752.7	20	8.2	13.3	18.3	28.4	35.6
		0.8	476.7	567.8	676.2	903.2	1101.0	3724.8	15	8.1	13.2	18.1	28.2	35.3
	0.1	0.6	484.4	564.9	679.1	917.3	1139.5	3785.0	24	8.5	13.8	19.0	29.3	36.5
		0.7	478.9	556.1	674.1	917.1	1097.1	3723.3	14	8.3	13.7	18.7	29.2	36.5
		0.8	469.0	548.9	671.0	909.9	1084.6	3683.3	9	8.3	13.5	18.5	28.7	36.1
	0.15	0.6	481.3	554.5	684.1	945.3	1170.5	3835.7	27	8.8	14.2	19.5	30.0	37.8
		0.7	477.3	547.3	667.0	915.0	1130.4	3736.9	17	8.6	14.1	19.3	29.9	37.6
		0.8	471.1	545.4	661.5	905.3	1109.9	3693.2	11	8.6	13.9	19.1	29.6	37.1
Ave.			475.8	553.1	676.1	914.2	1102.5	3721.6						
SD			5.0	11.5	9.9	12.7	24.2	51.0						

Bold text and italics cells: the best combination of the operating parameters

statistically significant at the 0.05 significance level, but the interactions of them were not significant. The right-hand side of Table 7 is the ANOVA table after pooling the non-significant interactions. The boxplot for each factor is shown in Fig. 8a–c to determine the optimal level of the parameters. EACS shows similar performance at 0.7 and 0.8 of p_{tune}. We assigned 0.8 to p_{tune} to improve calculation time. p_{mttc} was determined to be 0.1 which shows the most stable solution quality. p_{ns} was determined to be 0.8 which shows the best solution quality. This combination of the parameters is consistent with the result of the rank in Table 5. Although the summary is not presented, ANOVA for the $\overline{t_{comp}}$ was also performed. All parameters and several interactions were

Table 6 ANOVA tables for the sum of $\overline{C_{max}}$s

Source	Before pooling				After pooling			
	df	MSS	F-ratio	*p* value	*df*	MSS	F-ratio	*p* value
p_{tune}	2	5188	12.65	0.00	2	5188	9.82	0.00
p_{mttc}	2	5021	12.24	0.00	2	5021	9.50	0.00
p_{ns}	2	19,619	47.82	0.00	2	19,619	37.13	0.00
$p_{tune} \times p_{mttc}$	4	657	1.60	0.26				
$p_{tune} \times p_{ns}$	4	329	0.80	0.56				
$p_{mttc} \times p_{ns}$	4	835	2.03	0.18				
Residuals	8	410			20	528		

df degree of freedom, *MSS* mean sum-of-square

statistically significant. However, as Fig. 8d shows, the computation time decreases sharply as p_{tune} increases. This is because the route improvement using the procedure TUNNING is much faster than the search of the new route using the route construction rule. However, the larger the p_{tune}, the worse the solution quality, so p_{tune} is determined as the arbitrated value.

6.2 Effect of sequence-dependent setups

To confirm the assertion of Wan et al. (2011) that the optimal schedule can be improved by considering sequence-independent setups independently, an experiment was conducted on the benchmark problems. The experimental environment was the same as Table 4. Figures 9 and 10 show the best solution of EACSNS and EACS for P21 in a Gantt chart. In the figure, the y-axis represents machine identification and the x-axis is time. Each box represents schedules of an operation, where blue boxes are tool change times, red boxes are setup change times, yellow boxes are unloading times, and gray boxes are transportation time. The color of the texted boxes represents the job identification and the text in the box denotes (j, o, m, t, k) of the operation. The thick lined black rounded boxes in Fig. 6 show the examples for which the schedule was improved by applying the sequence-dependent setup times, in other words, the sequence-dependent setup times can be reduced by EACS.

The experimental results for $\overline{C_{max}}$ and $\overline{t_{comp}}$ are summarized in Table 7. EACS improves $\overline{C_{max}}$ up to 22% over EACSNS. The degree of improvement depends on the problem settings and the algorithm such as the magnitude of the setup times, the performance of the algorithm, the size of the ant colony, and the number of iterations. Nevertheless, the improvement of the makespan is clear and it positively affects to manufacturing cost.

Table 7 Makespan improvement caused by considering the sequence-dependent setup times for the benchmark problems

Method	EACSNS			EACS			Improvement rate (%)	
Prob.	C^*_{max}	$\overline{C_{max}}$	SD	C^*_{max}	$\overline{C_{max}}$	SD	C^*_{max}	$\overline{C_{max}}$
P1	583	593.8	5.7	442	472.3	15.8	24.19	20.46
P2	601	611.9	5.2	460	480.1	12.6	23.46	21.55
P3	527	561.0	14.0	423	465.0	20.3	19.73	17.12
P4	527	541.0	6.5	440	476.3	15.1	16.51	11.96
P5	467	491.4	9.5	395	420.2	17.7	15.42	14.50
P6	613	629.3	9.0	470	518.4	23.5	23.33	17.62
P7	583	603.4	7.6	462	502.5	16.9	20.75	16.72
P8	593	611.6	6.6	426	478.6	18.1	28.16	21.75
P9	516	552.5	13.9	394	452.3	26.0	23.64	18.14
P10	605	621.3	8.5	472	505.0	16.1	21.98	18.72
P11	537	556.5	11.0	433	468.0	18.1	19.37	15.90
P12	457	482.5	10.6	349	394.1	17.5	23.63	18.32
P13	619	640.0	11.4	507	538.0	16.4	18.09	15.93
P14	607	644.0	18.1	521	575.5	23.9	14.17	10.64
P15	544	573.3	13.8	463	517.7	21.3	14.89	9.69
P16	652	673.2	13.7	541	584.1	22.3	17.02	13.24
P17	625	664.1	20.4	537	584.6	23.6	14.08	11.97
P18	612	639.7	12.5	482	537.0	23.8	21.24	16.07
P19	637	690.5	16.6	576	620.3	19.6	9.58	10.16
P20	489	523.4	17.8	388	451.3	19.7	20.65	13.76
P21	701	735.5	20.5	606	661.4	21.7	13.55	10.07
P22	721	769.4	22.4	659	709.2	23.4	8.60	7.82
P23	660	688.5	16.9	587	636.6	21.3	11.06	7.54
P24	755	797.3	27.5	673	726.4	24.1	10.86	8.90
P25	762	796.5	20.4	682	724.3	23.5	10.50	9.07
P26	695	725.3	20.9	589	656.2	25.8	15.25	9.53
P27	762	804.3	23.1	675	735.6	27.1	11.42	8.55
P28	650	700.4	23.6	588	637.7	19.5	9.54	8.94
P29	882	953.9	36.8	808	904.5	40.1	8.39	5.18
P30	880	949.9	37.3	816	900.1	40.2	7.27	5.25
P31	1038	1141.1	55.4	974	1086.1	55.5	6.17	4.82
						Min	6.17	4.82
						Max	28.16	21.75
						Ave.	16.21	12.90

6.3 Performance comparison for other meta-heuristics

The final series of experiments were conducted to compare the performance with existing meta-heuristics for IPPS. Four algorithms were used for the comparative study. mPSO (Miljković and Petrović 2017) and FSDPSO (Dou et al. 2018)

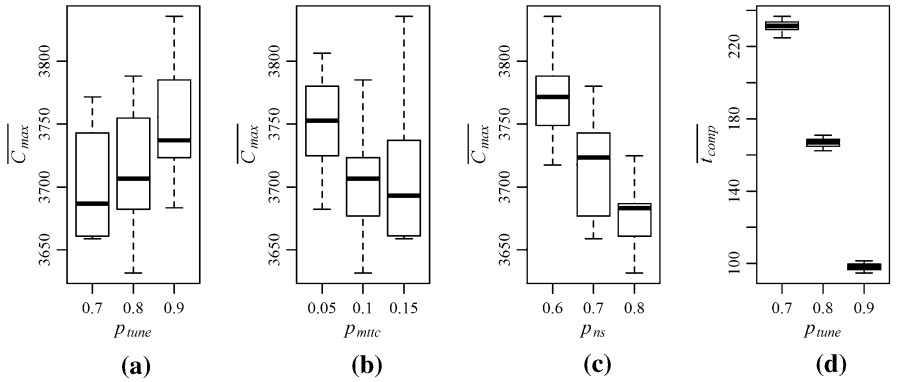

Fig. 8 Boxplots for $\overline{C_{max}}$ and $\overline{t_{comp}}$ according to p_{tune}, p_{mttc}, and p_{ns}

are based on PSO and E-ACO (Zhang and Wong 2014) and EACS are based on ACO. The experiment was performed in the same way as the parameter setting in Table 4. As an exception, we increased the swarm size n_K of FSDPSO to 60 for a fair comparison, because its computation time per iteration is very short compared to other meta-heuristics. The experimental results are summarized in Table 8. In Table 8, makespan and computation time are averaged for fifty repetitions, such as $\overline{C_{max}}$ and $\overline{t_{comp}}$, respectively. C^*_{max} is the best makespan and r_{inf} indicates the percentage of infeasible solutions among 50 repetitions. The last two columns compare C^*_{max} and $\overline{C_{max}}$ of the current best algorithm FSDPSO versus EACS.

According to Table 8, with regards to solution quality, EACS presented the smallest C^*_{max} and $\overline{C_{max}}$ for all problems except P4. This is easily confirmed by the boxplot in Fig. 11. Compared with the mPSO and E-ACO, EACS provides better solutions for all benchmark problems with less computation times. Even EACS is superior to the current best algorithm FSDPSO in solution quality. The decreases in C^*_{max} and $\overline{C_{max}}$ of EACS compared with those of FSDPSO are 22% and 21%, respectively, and the increases in computation time is negligible. In fact, if we adjust p_{tune} of EACS to 0.9, we can reduce $\overline{t_{comp}}$ by about 40% while slightly increasing $\overline{C_{max}}$ as shown in Table 5. Moreover, all solutions of the EACS and FSDPSO were feasible, but mPSO and E-ACO frequently derived infeasible solutions for large-sized problems. The effectiveness of **OC**, **MTTC**, and **TUNING**, which are three greedy heuristics proposed in Sect. 4.4, are summarized separately in Appendix 2.

Table 9 summarizes the results of hypothesis tests using a single-sided two sample t test for the experiment. Although the makespans of E-ACO for P31 did not satisfy the normality, the test was performed because the difference is so clear in the boxplot of Fig. 8. Some of test pairs do not satisfy the equality of variances, so all the tests were performed assuming unequal variances. As a result of the hypothesis test at a significance level 0.05, the solution quality of EACS was superior to that of all existing meta-heuristics even including EACSNS.

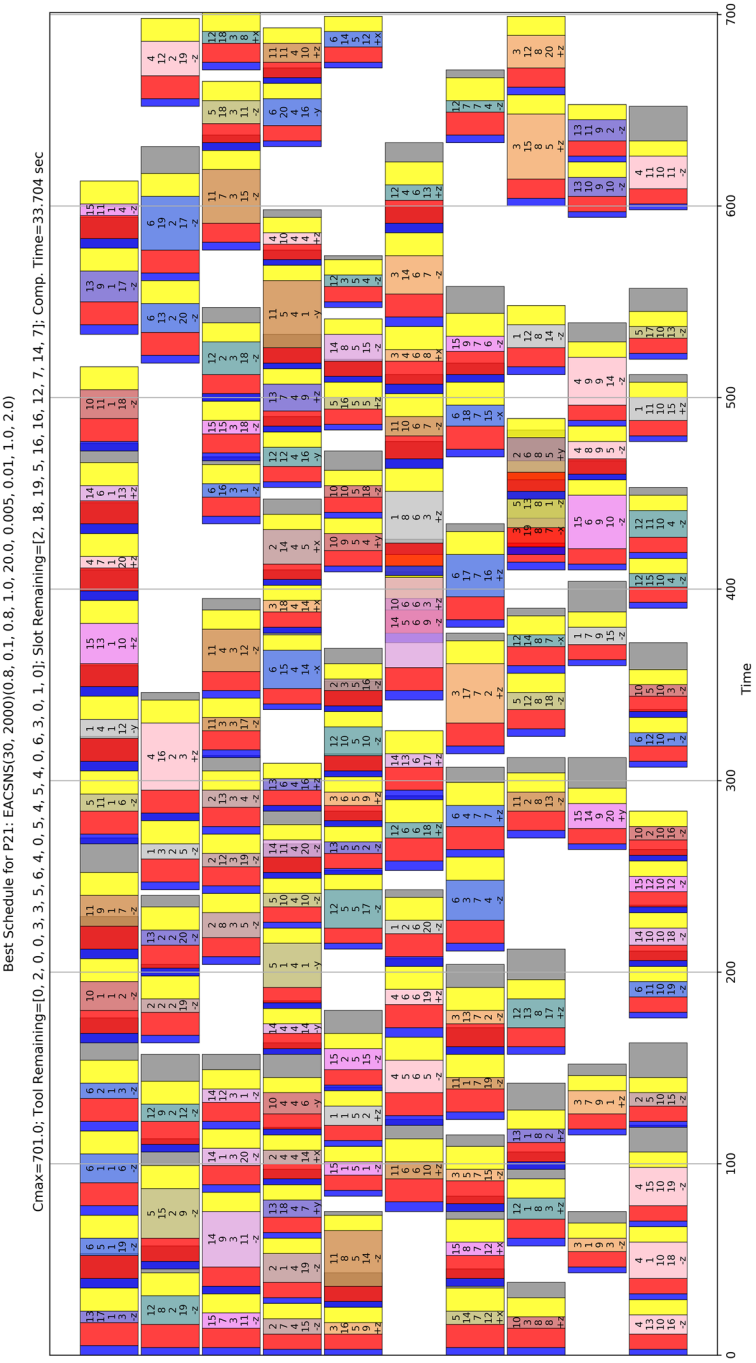

Fig. 9 Gantt chart of a solution of EACSNS for P21

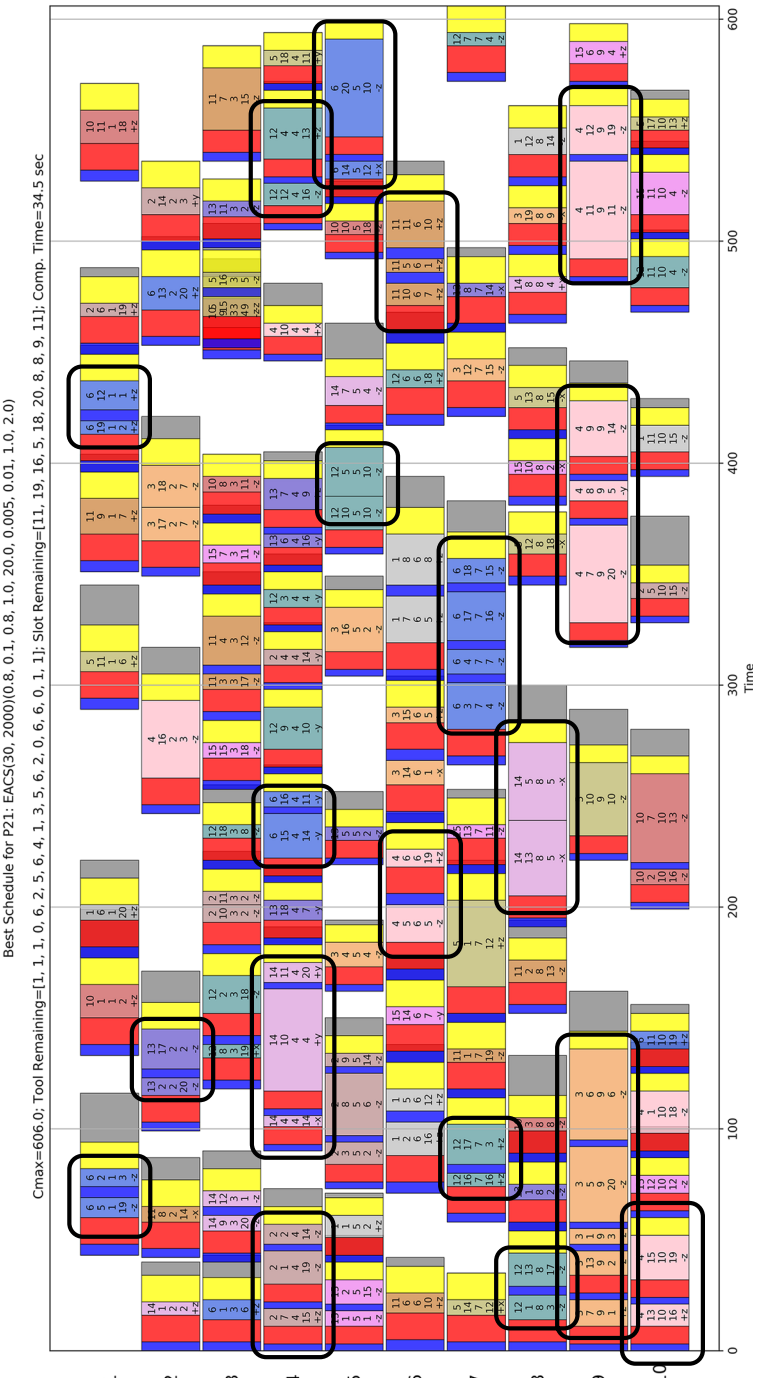

Fig. 10 Gantt chart of a solution of EACS for P21

Table 8 Performance comparison for various types of meta-heuristics for IPPS

Method	mPSO (Miljković and Petrović 2017)				FSDPSO (Dou et al. 2018)				E-ACO (Zhang and Wong 2014)				EACS				Decreased rate (%)	
Prob.	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$
P1	756	808.3	17.5	0	**481**	**549.1**	14.3	0	676	736.9	58.4	0	*442*	*472.3*	14.2	0	8.1	14.0
P2	752	818.8	19.5	0	**485**	**554.5**	16.0	0	689	742.0	68.1	0	*460*	*480.1*	15.4	0	5.2	13.4
P3	758	826.4	23.4	0	**471**	**533.3**	19.2	0	705	742.6	87.0	0	*423*	*465.0*	18.9	0	10.2	12.8
P4	666	738.4	19.3	0	*435*	**488.7**	16.0	0	641	675.1	64.0	0	440	*476.3*	14.8	0	− 1.1	2.5
P5	664	738.4	19.8	0	**446**	**505.3**	16.3	0	614	662.4	70.9	0	*395*	*420.2*	15.8	0	11.4	16.9
P6	835	892.4	21.0	0	**561**	**626.0**	17.4	0	788	826.4	79.9	0	*470*	*518.4*	17.3	0	16.2	17.2
P7	792	840.9	19.8	0	**489**	**568.3**	16.0	0	703	762.0	72.0	0	*462*	*502.5*	16.0	0	5.5	11.6
P8	753	818.5	18.6	0	**483**	**544.3**	15.7	0	694	742.1	65.3	0	*426*	*478.6*	14.8	0	11.8	12.1
P9	704	792.7	21.4	0	**453**	**519.5**	18.3	0	669	714.4	77.2	0	*394*	*452.3*	17.2	0	13.0	12.9
P10	799	865.0	18.9	0	**541**	**605.0**	15.8	0	755	791.9	66.6	0	*472*	*505.0*	15.3	0	12.8	16.5
P11	727	803.9	20.6	0	**459**	**512.8**	17.3	0	671	718.7	74.9	0	*433*	*468.0*	16.5	0	5.7	8.7
P12	679	738.8	20.4	0	**421**	**475.5**	17.0	0	610	666.2	75.2	0	*349*	*394.1*	16.2	0	17.1	17.1
P13	852	930.9	26.9	0	**584**	**682.1**	22.4	0	811	848.8	110.2	0	*507*	*538.0*	23.3	0	13.2	21.1
P14	891	960.7	31.7	0	**605**	**670.1**	26.9	0	801	848.9	132.5	0	*521*	*575.5*	28.3	0	13.9	14.1
P15	796	876.7	28.4	0	**503**	**585.5**	24.0	0	724	772.4	110.2	0	*463*	*517.7*	23.8	0	8.0	11.6
P16	927	984.0	30.1	0	**654**	**727.5**	25.4	0	858	902.9	134.1	0	*541*	*584.1*	28.2	0	17.3	19.7
P17	869	975.7	29.2	0	**610**	**693.5**	24.2	0	810	871.6	121.6	0	*537*	*584.6*	25.8	0	12.0	15.7
P18	844	913.1	29.2	0	**596**	**662.3**	25.2	0	766	835.0	122.0	0	*482*	*537.0*	25.2	0	19.1	18.9
P19	938	1013.6	29.7	0	**686**	**749.5**	25.3	0	879	922.5	125.3	0	*576*	*620.3*	26.2	0	16.0	17.2
P20	769	837.2	28.5	0	**498**	**567.6**	24.2	0	686	741.6	119.1	0	*388*	*451.3*	24.5	0	22.1	20.5
P21	989	1103.9	34.9	8	**723**	**810.1**	30.2	0	933	980.0	161.5	0	*606*	*661.4*	33.8	0	16.2	18.4
P22	1017	1196.1	40.8	62	**731**	**831.5**	37.1	0	934	1041.1	196.5	0	*659*	*709.2*	41.8	0	9.8	14.7
P23	941	1062.3	37.1	26	**636**	**737.3**	32.9	0	870	926.5	167.3	0	*587*	*636.6*	36.0	0	7.7	13.7

Table 8 (continued)

Method	mPSO (Miljković and Petrović 2017)				FSDPSO (Dou et al. 2018)				E-ACO (Zhang and Wong 2014)				EACS				Decreased rate (%)	
Prob.	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$
P24	1027	1174.9	38.9	52	**792**	**870.7**	34.9	0	1009	1064.2	191.9	0	*673*	*726.4*	41.1	0	15.0	16.6
P25	1068	1235.8	36.7	52	**747**	**830.7**	32.0	0	978	1061.0	171.2	0	*682*	*724.3*	37.2	0	8.7	12.8
P26	996	1123.7	38.3	32	**705**	**795.0**	34.9	0	934	991.9	182.2	0	*589*	*656.2*	39.0	0	16.5	17.5
P27	1030	1155.9	37.7	62	**772**	**861.6**	33.8	0	941	1056.8	178.7	0	*675*	*735.6*	38.1	0	12.6	14.6
P28	961	1134.5	39.2	46	**681**	**754.0**	35.5	0	910	970.8	186.2	0	*588*	*637.7*	40.3	0	13.7	15.4
P29	1150	1712.0	47.5	44	**901**	**1000.3**	45.6	0	1163	1309.1	261.5	15	*808*	*904.5*	55.5	0	10.3	9.6
P30	1194	1731.6	48.3	54	**901**	**994.2**	46.5	0	1186	1314.1	263.0	33	*816*	*900.1*	56.3	0	9.4	9.5
P31	1500	2243.2	56.2	90	**1016**	**1146.6**	57.6	0	1307	1543.3	328.9	50	*974*	*1086.1*	69.6	0	4.1	5.3
															Min		− 1.15	2.54
															Max		22.09	21.12
															Ave.		11.66	14.28

Bold text: the best solution among the conventional meta-heuristics

Italics cell: the overall best solution for the all compared meta-heuristics

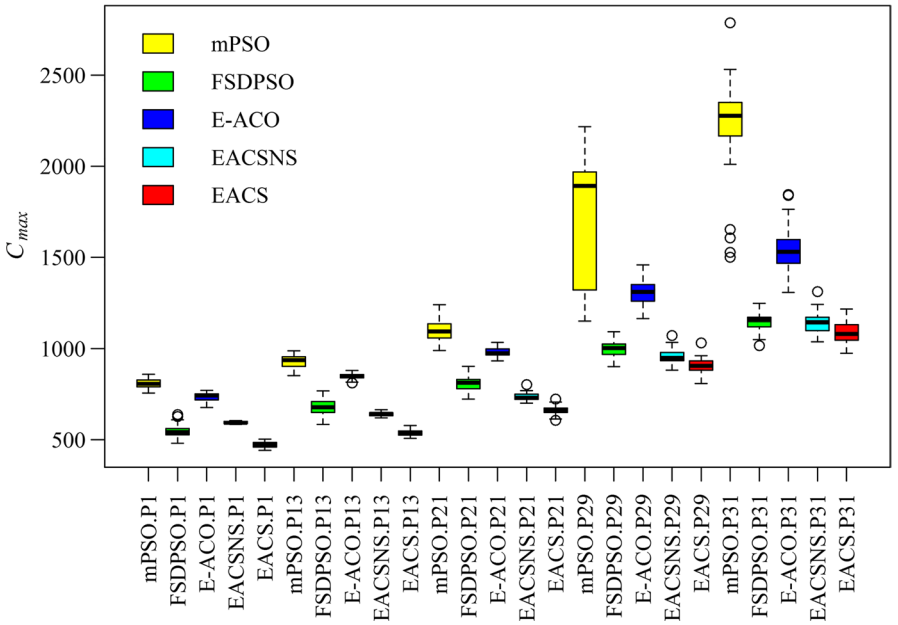

Fig. 11 Boxplots of C_{max} obtained by various methods for the selected IPPS problems

Table 9 Hypothesis test for $\mu_{C_{max}}$ at a significance level of 0.05(**)

Hypothesis	$H_1: \mu_{C_{max},EACS} < \mu_{C_{max},method}$ $H_0: \mu_{C_{max},EACS} = \mu_{C_{max},method}$							
Method	mPSO		FSDPSO		E-ACO		EACSNS	
Prob.	p value	Decision	p value	Decision	p value	Decision	p value	Decision
P1	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1
P13	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1
P21	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1
P29	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1	0.0000**	Accept H_1
P31	0.0000**	Accept H_1	0.0000**	Accept H_1	**0.0000****	Accept H_1	0.0000**	Accept H_1

Bold text: non-normal data by Shapiro–Wilk normality test at significance level of 0.01

7 Discussion and conclusion

The contributions of this study can be summarized in two ways. The first is that the flexibilities, constraints, and setups considered in the IPPS to date have been aggregated into a comprehensive IPPS problem. By doing so, the IPPS is closer to the real manufacturing environment. Secondly, we proposed a new ACO approach to solve the more complicated IPPS. Existing ACOs have limitations when they are applied to large-sized problems. The newly proposed EACS can reduce the computational

effort required to maintain the pheromone trails. Furthermore, it can improve the performance of the ACO by enhancing the capability of each ant agent. Nevertheless, there are some issues that need to be discussed.

First, although EACS improves the efficiency of ACO, it continues to require a large amount of computational effort. Various flexibilities considered in IPPS generate many alternative operations. Every ant agent in ACO should determine the next visiting node among those alternatives on every node in all iterations, which results in a large computation time. If the sequence-dependent setups are considered, it is even worse. We expect that development of a greedy heuristic such as Lin–Kernighan in the symmetric traveling salesman problem will help this obstacle. Second, a new design of pheromone trails is necessary. The key factors for determining the schedule in IPPS are job, operation, and machine. Thus, the pheromone trails should have three-dimensional structures that can handle these three factors simultaneously. However, both the existing ACO and our EACS run with two-dimensional pheromone trails based on job and operation. This is only because the repository size of the pheromone trails due to many alternatives. Parallel computing or distributed repository might be a solution for this problem.

Acknowledgements This research was funded by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015R1D1A1A01060391).

Appendix 1: Mathematical programming model of IPPS

The mathematical programming model for IPPS has been developed by Li et al. (2010) and Nourali et al. (2012). Nourali et al. (2012) proposed a mixed integer linear programming for assembly jobshop considering sequence-dependent setup times. However, the assembly jobshop problem is different from IPPS, but it does not consider loading and unloading, tool change, TAD change, and transportation time. On the other hand, Li et al. (2010) deals with IPPS but does not consider sequence-dependent setup times. Therefore, we present a new mixed integer nonlinear programming model for the proposed IPPS through the improvement of these two MILPs.

Sets and indices

j	job identification (ID) ($j \in J$)
o	operation ID of job j ($o \in O_j$)
m	machine ID ($m \in M$)
t	tool ID ($t \in T$)
k	TAD ID ($k \in K$)
p_j	process route ID of job j ($p_j \in P_j$)
J	set of jobs; $J = \{1, 2, 3, \ldots, n^{JOB}\}$, n^{JOB} is the total number of jobs
O_j	set of operations of job j ($O_j \subseteq O$); $O_j = \{1, 2, 3, \ldots, n_j^{OP}\}$, n_j^{OP} is the total number of operations of the job j
M	set of machines; $M = \{1, 2, 3, \ldots, n^M\}$, n^M is the total number of machines

T set of tools; $T = \{1, 2, 3, \ldots, n^T\}$, n^T is the total number of tool types
K set of TADs; $K = \{-x, x, -y, y, -z, z\}$, $-x, x, -y, y, -z, z$ are directions of tool access
P_j set of alternative process routes of job j
O_{jp_j} ordered set of operation IDs in the process route p_j of job j ($O_j = \bigcup_{p_j \in P_j} O_{jp_j} \forall j \in J$)
$O_{j,o}$ the operation o of job j
$O_{j,o}^{m,t,k}$ the $O_{j,o}$ that is assigned the machine m, the tool t, and the TAD k
O_{jp_ji} the ith operation ID of O_{jp_j}; O_{jp_jf} is the first and O_{jp_jl} is the last element of O_{jp_j}
O_{jp_jimtk} operation O_{jp_ji} that is assigned machine m, tool t, and TAD k
M_{jp_ji} set of alternative machines for the operation O_{jp_ji}, $M_{jp_ji} \subseteq M$
T_{jp_jim} set of alternative tools for the operation O_{jp_ji} at the machine m; $T_{jp_jim} \subseteq T$
K_{jp_jim} set of alternative TADs for the operation O_{jp_ji} at the machine m; $K_{jp_jim} \subseteq K$.

Parameters

$R_{jp_jii'}$ 1, if the operation O_{jp_ji} must precede the operation $O_{jp_ji'}$; 0, otherwise
C_m^{SLOT} the number (capacity) of tool slots of the machine m
C_t^{TOOL} the available number (capacity) of the tool t
r_t^{SLOT} the number of required slots for the tool t
L a very large number.

Decision variables

X_{jp_jimtk} 1, if machine m, tool t, and TAD k are selected for the O_{jp_ji}; 0, otherwise
Z_{jp_j} 1, if process route p_j of job j is selected; 0, otherwise
$Y_{jp_jij'p_{j'}i'm}$ 1, if operation O_{jp_ji} precedes operation $O_{j'p_{j'}i'}$ immediately on machine m; 0, otherwise
U_{mt} 1, if tool t is installed on machine m; 0, otherwise
st_{jp_jimtk} starting time of O_{jp_jimtk}
ct_{jp_jimtk} earliest completion time of O_{jp_jimtk}
sct_{jp_jimtk} setup change time of O_{jp_jimtk}
tct_{jp_jimtk} tool change time of O_{jp_jimtk}
pt_{jp_jimtk} processing time of O_{jp_jimtk}
ult_{jp_jimtk} unload time of O_{jp_jimtk}
trt_{jp_jimtk} transportation time to the successive operation of O_{jp_jimtk}
C_{max} makespan.

MINLP for IPPS

$$\text{Minimize} \quad C_{max} \tag{14}$$

subject to

$$\sum_{p_j \in P_j} Z_{jp_j} = 1 \quad \forall j \in J \tag{15}$$

$$\sum_{m \in M_{jp_j i}} \sum_{t \in T_{jp_j im}} \sum_{k \in K_{jp_j im}} X_{jp_j imtk} = Z_{jp_j} \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j} \tag{16}$$

$$\begin{aligned} st_{jp_j imtk} + ct_{jp_j imtk} \le L\left(X_{jp_j imtk}\right) \quad & \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \\ & \forall m \in M_{jp_j i}, \forall t \in T_{jp_j im}, \forall k \in K_{jp_j im} \end{aligned} \tag{17}$$

$$\begin{aligned} & st_{jp_j imtk} + sct_{jp_j imtk} + tct_{jp_j imtk} + pt_{jp_j imtk} + ult_{jp_j imtk} - L\left(1 - X_{jp_j imtk}\right) \le ct_{jp_j imtk} \\ & \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_j i}, \forall t \in T_{jp_j im}, \forall k \in K_{jp_j im} \end{aligned} \tag{18}$$

$$\begin{aligned} & ct_{j' p_{j'} i' mt' k'} - L\left(\mathrm{Y}_{jp_j ij' p_{j'} i' m}\right) \le st_{jp_j imtk} \\ & \forall j, j' \in J, \forall p_j \in P_j, \forall p_{j'} \in P_{j'} \forall i \in O_{jp_j}, \forall i' \in O_{j' p_{j'}}, \\ & \forall m \in M_{jp_j i} \cap M_{j' p_{j'}}, \forall t \in T_{jp_j im}, \forall k \in K_{jp_j im}, \forall t' \in T_{j' p_{j'} i' m}, \forall k' \in K_{j' p_{j'} i' m} \end{aligned} \tag{19}$$

$$\begin{aligned} & ct_{jp_j imtk} - L\left(1 - \mathrm{Y}_{jp_j ij' p_{j'} i' m}\right) \le st_{j' p_{j'} i' mt' k'} \\ & \forall j, j' \in J, \forall p_j \in P_j, \forall p_{j'} \in P_{j'} \forall i \in O_{jp_j}, \forall i' \in O_{j' p_{j'}}, \\ & \forall m \in M_{jp_j i} \cap M_{j' p_{j'}}, \forall t \in T_{jp_j im}, \forall k \in K_{jp_j im}, \forall t' \in T_{j' p_{j'} i' m}, \forall k' \in K_{j' p_{j'} i' m} \end{aligned} \tag{20}$$

$$\begin{aligned} & \sum_{m \in M_{jp_j (i-1)}} \sum_{t \in T_{jp_j (i-1) m}} \sum_{k \in K_{jp_j (i-1) m}} \left(ct_{jp_j (i-1) mtk} + trt_{jp_j (i-1) mtk}\right) \le \sum_{m' \in M_{jp_j i}} \sum_{t' \in T_{jp_j im}} \sum_{k' \in K_{jp_j im}} st_{jp_j im' t' k'} \\ & \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j} - \left\{O_{jp_j f}\right\} \end{aligned} \tag{21}$$

$$\begin{aligned} & \sum_{m \in M_{jp_j i}} \sum_{t \in T_{jp_j im}} \sum_{k \in K_{jp_j im}} \left(ct_{jp_j imtk} + trt_{jp_j imtk}\right) - L\left(1 - R_{jp_j ii'}\right) \le \sum_{m' \in M_{jp_j i'}} \sum_{t' \in T_{jp_j i' m}} \sum_{k' \in K_{jp_j i' m}} st_{jp_j i' m' t' k'}, \\ & \forall i < i', j \in J, \forall p_j \in P_j, \forall i, i' \in O_{jp_j} \end{aligned} \tag{22}$$

$$ult_{jp_jlmtk} \geq ULT(m) \quad \forall j \in J, \forall p_j \in P_j, \forall m \in M_{jp_jl}, \forall t \in T_{jp_jlm}, \forall k \in K_{jp_jlm} \tag{23}$$

$$ct_{jp_jlmtk} \leq C_{max} \quad \forall j \in J, \forall p_j \in P_j, \forall m \in M_{jp_jl}, \forall t \in T_{jp_jlm}, \forall k \in K_{jp_jlm} \tag{24}$$

$$X_{jp_jimtk} \geq U_{mt} \quad \forall j \in J, \forall p_j \in P_j, \forall m \in M_{jp_jl}, \forall t \in T_{jp_jlm}, \forall k \in K_{jp_jlm} \tag{25}$$

$$\sum_{t \in T} r_t^{SLOT} \times U_{mt} \leq C_m^{SLOT} \quad \forall m \in M \tag{26}$$

$$\sum_{m \in M} U_{mt} \leq C_t^{TOOL} \quad \forall t \in T \tag{27}$$

$$st_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{28}$$

$$sct_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{29}$$

$$tct_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{30}$$

$$pt_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{31}$$

$$ult_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{32}$$

$$trt_{jp_jimtk} \geq 0 \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{33}$$

$$X_{jp_jimtk} \in \{0,1\} \quad \forall j \in J, \forall p_j \in P_j, \forall i \in O_{jp_j}, \forall m \in M_{jp_ji}, \forall t \in T_{jp_jim}, \forall k \in K_{jp_jim} \tag{34}$$

$$Z_{jp_j} \in \{0,1\} \quad \forall j \in J, \forall p_j \in P_j \tag{35}$$

$$\mathrm{Y}_{jp_jij'p_{j'}i'm} \in \{0,1\} \quad \forall j, j' \in J, \forall p_j \in P_j, \forall p_{j'} \in P_{j'} \forall i \in O_{jp_j}, \forall i' \in O_{j'p_{j'}}, \forall m \in M_{jp_ji} \cap M_{j'p_{j'}} \tag{36}$$

$$U_{mt} \in \{0,1\} \quad \forall m \in M, \forall t \in T \tag{37}$$

IPPS is a problem that minimizes makespan of the objective function (14). Constraints (15) allow only one process route to be selected in each job. Constraints (16) make sure that each operation of a selected process route selects only a combination of (m, t, k). Constraints (17) make time schedules of all dummy operations zero. Constraints (18) ensure that each operation is completed by consuming

relevant sequence-dependent setup times and processing time. Constraints (19) and (20) ensure that two or more operations are not performed simultaneously on the same machine. Constraints (21) consider transportation time for workpiece movement in the same job. Constraints (22) ensure that all precedence relations between operations are satisfied. Constraints (23) ensure that the completed workpiece is unloaded at the last operation of each job. Constraints (24) determine makespan. Constraints (25) prevent duplicate installation of the same tool on a machine. Constraints (26) and (27) are constraints on machine's slot capacity and tool capacity, respectively. Constraints (28)–(37) are the possible ranges of time schedules and decision variables.

Appendix 2: Validation of effectiveness of OC, MTTC, and TUNING

Additional experiments were performed to verify the effectiveness of the three greedy heuristics OC, MTTC, and TUNING proposed in Sect. 4.4. OC, MTTC, and TUNING were inserted into the procedures of mPSO, FSDPSO, and E-ACO. Then, the five representative problems P1, P13, P21, P29 and P31 were repeated 10 times for each algorithm. Table 10 summarizes the experimental results before and after applying the three greedy heuristics. Three greedy heuristics reduced $\overline{C_{max}}$ by 4.5% and 0.8% on average for mPSO and FSDPSO, respectively. On the other hand, for E-ACO, it resulted in a dramatic improvement by reducing $\overline{C_{max}}$ by 27.4% and $\overline{t_{comp}}$ by 75.4%. This phenomenon occurs because mPSO and FSDPSO contain procedures like OC, MTTC, and TUNING, but E-ACO does not. The results of this comparative study demonstrate that these three greedy heuristics are effective in improving the solution and reducing the computation time. Another notable point is that the performance of our proposed EACS is still better than others, although other meta-heuristics have been improved by OC, MTTC, and TUNING.

Table 10 Analysis of effectiveness of three greedy heuristics for various types of ACOs

Method	mPSO (Miljković and Petrović 2017)				FSDPSO (Dou et al. 2018)				E-ACO (Zhang and Wong 2014)				EACS			
Prob.	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)	C^*_{max}	$\overline{C_{max}}$	$\overline{t_{comp}}$	r_{inf} (%)
Heuristics without **OC**, **MTTC**, and **TUNING** (50 repetitions)																
P1	756	808.3	17.5	0	**481**	**549.1**	14.3	0	676	736.9	58.4	0	*442*	*472.3*	14.2	0
P13	852	930.9	26.9	0	**584**	**682.1**	22.4	0	811	848.8	110.2	0	*507*	*538.0*	23.3	0
P21	989	1103.9	34.9	8	**723**	**810.1**	30.2	0	933	980.0	161.5	0	*606*	*661.4*	33.8	0
P29	1150	1712.0	47.5	44	**901**	**1000.3**	45.6	0	1163	1309.1	261.5	15	*808*	*904.5*	55.5	0
P31	1500	2243.2	56.2	90	**1016**	**1146.6**	57.6	0	1307	1543.3	328.9	50	*974*	*1086.1*	69.6	0
Heuristics applying **OC**, **MTTC**, and **TUNING** (10 repetitions)																
P1	741	778.3	20.7	0	*428*	**494.6**	20.4	0	484	495.3	15.4	0	446	*467.9*	14.2	0
P13	894	939.6	29.9	0	**602**	**644.7**	31.2	0	559	572.0	26.7	0	*519*	*537.4*	23.9	0
P21	1002	1097.8	41.0	20	**736**	**805.7**	41.7	0	673	703.6	38.6	0	*639*	*666.2*	34.8	0
P29	1219	1485.0	55.7	60	**980**	**1053.4**	61.6	0	892	1008.9	63.0	50	*853*	*900.8*	54.5	0
P31	1917	2111.4	66.1	100	**1124**	**1223.2**	76.0	0	1037	1224.1	80.1	100	*1024*	*1093.2*	67.9	0
Improved rate (%)																
P1	2.0	3.7	− 18.3		11.0	9.9	− 42.5		28.4	32.8	73.7		− 0.9	0.9	0.1	
P13	− 4.9	− 0.9	− 10.9		− 3.1	5.5	− 39.4		31.1	32.6	75.7		− 2.4	0.1	− 2.6	
P21	− 1.3	0.6	− 17.5		− 1.8	0.5	− 37.8		27.9	28.2	76.1		− 5.4	− 0.7	− 3.0	
P29	− 6.0	13.3	− 17.3		− 8.8	− 5.3	− 34.9		23.3	22.9	75.9		− 5.6	0.4	1.9	
P31	− 27.8	5.9	− 17.5		− 10.6	− 6.7	− 31.9		20.7	20.7	75.6		− 5.1	− 0.7	2.5	
Ave	− 7.6	4.5	− 16.3		− 2.7	0.8	− 37.3		26.3	27.4	75.4		− 3.9	0.0	− 0.2	

Bold text: the best solution among the conventional meta-heuristics

Italics cell: the overall best solution for the all compared meta-heuristics

References

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res 246:345–378

Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187:985–1032. https://doi.org/10.1016/j.ejor.2006.06.060

Azab A, ElMaraghy HA, Samy SN (2009) Reconfiguring process plans: a new approach to minimize change. In: Changeable and reconfigurable manufacturing systems, pp 179–194

Chandra MB, Baskaran R (2012) A survey: ant colony optimization based recent research and implementation on several engineering domain. Expert Syst Appl 39:4618–4627. https://doi.org/10.1016/j.eswa.2011.09.076

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1:53–66. https://doi.org/10.1109/4235.585892

Dou J, Li J, Su C (2018) A discrete particle swarm optimisation for operation sequencing in CAPP. Int J Prod Res 56:3795–3814. https://doi.org/10.1080/00207543.2018.1425015

Eren T (2010) A bicriteria m-machine flowshop scheduling with sequence-dependent setup times. Appl Math Model 34:284–293. https://doi.org/10.1016/J.APM.2009.04.005

Guo YW, Li WD, Mileham AR, Owen GW (2009) Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach. Int J Prod Res 4714:3775–3796. https://doi.org/10.1080/00207540701827905

Kim YK, Kim JY, Shin KS (2007) An asymmetric multileveled symbiotic evolutionary algorithm for integrated FMS scheduling. J Intell Manuf 18:631–645. https://doi.org/10.1007/s10845-007-0037-5

Leung CWW, Wong TNN, Mak KLL, Fung RYKYK (2010) Integrated process planning and scheduling by an agent-based ant colony optimization. Comput Ind Eng 59:166–180. https://doi.org/10.1016/j.cie.2009.09.003

Li WD, McMahon CA (2007) A simulated annealing-based optimization approach for integrated process planning and scheduling. Int J Comput Integr Manuf 20:80–95. https://doi.org/10.1080/09511920600667366

Li WD, Ong SK, Nee AYC (2002) Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts. Int J Prod Res 40:1899–1922. https://doi.org/10.1080/00207540110119991

Li X, Gao L, Shao X et al (2010) Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. Comput Oper Res 37:656–667. https://doi.org/10.1016/j.cor.2009.06.008

Liu J, MacCarthy BL (1997) A global MILP model for FMS scheduling. Eur J Oper Res 100:441–453. https://doi.org/10.1016/S0377-2217(96)00055-0

Liu X, Ni Z, Qiu X (2016) Application of ant colony optimization algorithm in integrated process planning and scheduling. Int J Adv Manuf Technol 84:1–13. https://doi.org/10.1007/s10845-010-0407-2

Miljković Z, Petrović M (2017) Application of modified multi-objective particle swarm optimisation algorithm for flexible process planning problem. Int J Comput Integr Manuf 30:271–291. https://doi.org/10.1080/0951192X.2016.1145804

Moon C, Kim J, Hur S (2002) Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Comput Ind Eng 43:331–349. https://doi.org/10.1016/S0360-8352(02)00078-5

Nourali S, Imanipour N, Shahriari MR (2012) A mathematical model for integrated process planning and scheduling in flexible assembly job shop environment with sequence dependent setup times. Int J Math Anal 6:2117–2132

Petrović M, Vuković N, Mitić M, Miljković Z (2016) Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. Expert Syst Appl 64:569–588. https://doi.org/10.1016/j.eswa.2016.08.019

Rachamadugu R, Stecke KE (1994) Classification and review of FMS scheduling procedures. Prod Plan Control 5:2–20. https://doi.org/10.1080/09537289408919468

Reddy SVB (1999) Operation sequencing in CAPP using genetic algorithms. Int J Prod Res 37:1063–1074. https://doi.org/10.1080/002075499191409

Shao X, Li X, Gao L, Zhang C (2009) Integration of process planning and scheduling—a modified genetic algorithm-based approach. Comput Oper Res 36:2082–2096. https://doi.org/10.1016/j.cor.2008.07.006

Shen X-N, Yao X (2015) Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inf Sci (NY) 298:198–224. https://doi.org/10.1016/j.ins.2014.11.036

Srinivas PS, Raju VR, Rao CSP (2012) Optimization of process planning and scheduling using ACO and PSO algorithms. Int J Emerg Technol Adv Eng 2:343–354

Stützle T, Dorigo M (1999) ACO algorithms for the traveling salesman problem. In: Miettinen K, Mākelā M, Neittaanmāki P, Periaux J (eds) Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, genetic programming and industrial applications. Wiley, New York, pp 1–23

Tan W, Khoshnevis B (2000) Integration of process planning and scheduling—a review. J Intell Manuf 11:51–63

Wan SY, Wong TN, Zhang S, Zhang L (2011) Integrated process planning and scheduling with setup time consideration by ant colony optimization. In: Proceedings of the 41st International conference on computers and industrial engineering, pp 998–1003

Wang J, Fan X, Zhang C, Wan S (2014) A graph-based ant colony optimization approach for integrated process planning and scheduling. Chin J Chem Eng 22:748–753. https://doi.org/10.1016/j.cjche.2014.05.011

Zhang SC, Wong TN (2013) An enhanced ant colony optimization approach for integrating process planning and scheduling based on multi-agent system. In: Proceedings of the 5th IESM conference. Rabat, Morocco

Zhang S, Wong TN (2014) Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. J Intell Manuf 29:1–17. https://doi.org/10.1007/s10845-014-1023-3

Zhang L, Wong TN (2016) Solving integrated process planning and scheduling problem with constructive meta-heuristics. Inf Sci (NY) 340–341:1–16. https://doi.org/10.1016/j.ins.2016.01.001

Chunghun Ha is currently an Associate Professor of Department of Industrial Engineering at Hongik University. He received a B.S. in Electronics Engineering from Yonsei University and a master and doctoral degree in industrial engineering from Texas A & M University. He worked as an engineer at Samsung Advanced Institute of Technology (SAIT) and Samsung Electronics. His research interests include optimization, scheduling, reliability and yield modeling, and demand forecasting.