Applied Intelligence (2022) 52:1752–1769
https://doi.org/10.1007/s10489-021-02336-z

An ant colony-based algorithm for integrated scheduling on batch machines with non-identical capacities

Zhao-hong Jia[1,2] · Yu-fei Cui[2] · Kai Li[3]

Accepted: 6 March 2021 / Published online: 28 May 2021

Abstract
In this paper, a production–distribution scheduling problem with non-identical batch machines and multiple vehicles is considered. In the production stage, n jobs are grouped into batches, which are processed on m parallel non-identical batch machines. In the distribution stage, there are multiple vehicles with identical capacities to deliver jobs to customers after the jobs are processed. The objective is to minimize the total weighted tardiness of the jobs. Considering the NP-hardness of the studied problem, an algorithm based on ant colony optimization is presented. A new local optimization strategy called LOC is proposed to improve the local exploitation ability of the algorithm and further search the neighborhood solution to improve the quality of the solution. Moreover, two interval candidate lists are proposed to reduce the search for the feasible solution space and improve the search speed. Furthermore, three objective-oriented heuristics are developed to accelerate the convergence of the algorithm. To verify the performance of the proposed algorithm, extensive experiments are carried out. The experimental results demonstrate that the proposed algorithm can provide better solutions than the state-of-the-art algorithms within a reasonable time.

Keywords Parallel batch machines · Non-identical machine capacities · Production and distribution · Ant colony optimization algorithm · Total weighted tardiness

1 Introduction

As a new branch of scheduling, batch scheduling problems (BSPs) [1] extensively exist in industrial manufacturing systems, such as metal manufacturing, pharmaceuticals, aeronautics, semiconductor manufacturing, and logistics, among others [2]. Different from the classical scheduling problems, one typical feature of scheduling on batch processing machines (BPMs) is that several jobs can be processed as a batch on one BPM simultaneously as long as the total size of jobs in the batch does not exceed the capacity of the BPM.

In recent years, the coordination of production scheduling and product transportation has become increasingly important in supply chain management. With the advancement of the make-to-order (MTO) business model in which products are customized and delivered from factory to customer and no finished product inventory is held between them, the production and distribution are therefore more closely linked and jointly scheduled. The integrated scheduling models are often used in time-sensitive supply-chain management, such as the production and delivery of perishable products [3] and industrial adhesive material [4]. In such examples from practical life, because of the time-sensitive characteristics of these products, orders should be delivered to the customers directly without intermediate inventory. Additionally, the delay in the delivery of these products may not only incur a tardiness penalty due to customer dissatisfaction and potential loss of reputation, but also lead to failure of the supply chains. However, with the increasingly fierce competition in the case

✉ Zhao-hong Jia
zhjia@mail.ustc.edu.cn

Yu-fei Cui
yfcui0808@163.com

Kai Li
hfutlk@139.com

1 Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Hefei, China

2 School of Computer Science and Technology, Anhui University, Hefei, 230039, China

3 School of Management, Hefei University of Technology, Hefei, 230009, China

of MTO manufacturing, productivity related to due dates has become more significant for manufacturers. Therefore, the integrated scheduling problem involving due-date considerations becomes very crucial for most of today's the businesses, and how to effectively integrate the production and delivery stages at the operational level to lower operational costs and improve customer service become very important to company success.

A large number of companies globally rely on third-party logistics (3PL) providers for their daily transportation of products and other logistics needs. 3PL providers usually follow a fixed daily or weekly service plan to serve their customers. The objective of integrated scheduling is generally to achieve an overall optimization for the supply chain and the obtained schedule can offer effective guidance for operations management.

There are two different modes related to vehicle capacity. In the first mode, it is assumed that each job occupies one unit of the capacity, without loss of generality. On the one hand, if the sizes of jobs are identical, each job is assumed to occupy one unit of the capacity. On the other hand, if the jobs have non-identical sizes, each job is assumed to take up one standard-size pallet for delivery convenience, regardless of the sizes of jobs. Hence, each vehicle can transport a fixed number of jobs in the first mode. In the second mode, the amount of capacity occupied by each job is assumed to equal the size of the job, which is suitable for addressing BSPs with unequal-size jobs. Therefore, the second mode is employed in addressing the studied problem.

In this paper, the problem of integrated scheduling of production and distribution is investigated, and the objective of the studied problem is to minimize the total weighted tardiness (TWT) of the jobs. Furthermore, an ant colony optimization (ACO) algorithm with a novel local optimization is proposed to group the jobs into batches in the production stage. Moreover, a heuristic is developed to transport the completed jobs to customers during the distribution phase. The main contributions of the proposed algorithm are the following.

(1) A new local optimization strategy called LOC is proposed to improve the local exploitation ability of the algorithm and further search the neighborhood solution to improve the quality of the solution.
(2) Two interval candidate lists (ICLs) are proposed to reduce the search for the feasible solution space and improve the search speed.
(3) Three objective-oriented heuristics are developed to accelerate the convergence of the algorithm.

The rest of this paper is organized as follows. In Section 2, the literature on batch scheduling and the integrated scheduling model is reviewed. In Section 3, the studied problem is formulated. Thereafter, an ACO algorithm with a novel local optimization based on efficient solution construction is proposed in Section 4. In Section 5, results and analysis on comparative experiments are provided. Conclusions and future research directions are presented in Section 6.

A list of all abbreviations used in this paper can be found in Appendix A.

2 Literature review

With the emergence of manufacturing, the BSP has been widely investigated [5–7]. Furthermore, makespan, one of the common objectives of scheduling problems, is usually used to evaluate the productivity of companies from many different scenarios. Reference [8] studied BSP with non-identical job sizes on a single machine and proposed an artificial bee colony (ABC) approach to minimize the makespan. Reference [9] designed two improved heuristics: (1) the first-fit longest processing time (FFLPT) rule is improved by considering identical job size and (2) the best-fit longest processing time (BFLPT) rule is improved by using an enumeration scheme. The objective of the problem was to minimize makespan on a BPM with non-identical job sizes and unequal job processing times. Reference [10] dealt with the problem of scheduling jobs with arbitrary sizes on the parallel BPMs with non-identical capacities and presented a heuristic based on the first-fit-decreasing (FFD) rule, as well as a meta-heuristic based on max-min ant system (MMAS), to minimize the makespan. Reference [11] proposed a branch-and-bound algorithm to minimize the makespan of the problem of scheduling jobs with unit size, different processing times, and arbitrary release dates on parallel BPMs. Reference [12] studied the problem of scheduling jobs with dynamic arrival times on parallel BPMs with arbitrary capacities to minimize makespan and proposed two meta-heuristics based on ACO to tackle it.

The objective of total (weighted) completion time (TCT/TWCT) is related to measure customer satisfaction in general. Therefore, with the rise of MTO manufacturing, an increasing number of BSP researchers have begun to consider this objective. Reference [13] proposed a dynamic programming approach to minimize the TCT of the problem of scheduling jobs on an unbounded BPM. Reference [14] investigated the problem of scheduling jobs in incompatible families and dynamic arrivals on a single BPM to minimize the TCT. They put forward a decomposed branch-and-bound algorithm according to the characteristics of dynamic job arrivals. Reference [15] studied the problem of serial batch scheduling on uniform parallel machines to minimize TCT and presented a polynomial-time procedure with the complexity of $O(m^2 \cdot n^{m+2})$. Reference [16] proposed two ACO-based algorithms with different coding methods to minimize the TCT of the BSP. The two coding methods are based on job sequence and batch sequence, respectively. Reference [17] investigated the problem of minimizing the total weighted completion time on parallel BPMs with identical capacities, non-identical job sizes, and unequal weights. They presented an ACO-based meta-heuristic, as well as a mixed-integer programming (MIP) formulation, to address it.

Minimizing total (weighted) tardiness, one of the important objectives that reflect the production performance of the time-sensitive jobs, has widely been considered in scheduling problems with order due dates. Reference [18] studied the scheduling problem of minimizing earliness–tardiness (E/T) on a single BPM, in which jobs have non-identical sizes and a common due date. They presented a heuristic called the minimum attribute ratio of the batch (MARB), a hybrid genetic algorithm (GA), and a mathematical model to solve the problem. Reference [19] dealt with scheduling problems in two-stage hybrid flow shops in which jobs have different due dates and each machine has identical capacities. They proposed a MIP model and three iterative algorithms to address the problem. Reference [20] provided a mathematical model considering dynamic starting conditions and developed heuristic algorithms to solve the scheduling problem for minimizing TWT under the constraint of unequal release times, incompatible job families, non-identical job sizes, and heterogeneous BPMs. Reference [21] studied the scheduling problem on non-identical parallel BPMs with job release times and non-identical job sizes to minimize the TWT and presented several heuristics, a MIP model, dynamic programming methods, and simulated annealing (SA) algorithms to resolve the problem.

The above studies only considered the production stage of products but ignored the transportation process. With the development of the logistics industry, the transportation of products plays an important role in this field. Reference [22] studied the vehicle scheduling problem of minimizing the required transportation time and presented two meta-heuristic methods, variable neighborhood search (VNS) and greedy randomized adaptive search procedure (GRASP), to solve the problem. Reference [23] proposed a novel dynamic public transport vehicle allocation scheme based on the emergent intelligence (EI) technique and built mathematical models for estimation of resources, utilization, and reliability parameters for solving public transport system problems. Reference [24] dealt with transportation vehicle scheduling problems in a supply-chain network and presented a collaborative transportation scheduling strategy, as well as a meta-heuristic algorithm called chemical reaction optimization(CRO), to minimize the total transportation cost. Reference [25] investigated the problem of minimizing the total transportation cost in urban cold chain transportation logistics. They presented an improved GA to address it. Reference [26] dealt with the vehicle routing problem of minimizing the traveling cost. The hybridization of the particle swarm optimization (PSO) and adaptive large neighborhood search (ALNS) algorithms was first developed for solving the problem.

In the past, the optimized objectives of BSPs with a single phase were related to production efficiency, and included makespan, total completion time, and total delay time, among other objectives. With the rise of the service and manufacturing industries, two-stage integrated scheduling problems have played an increasingly important role in reducing operating costs and improving service level and customer satisfaction in those industries. In addition, to be closer to real world applications, the completion time of an order extended from the completion time of processing in a single production stage to the delivery time to the customer in the transportation phase. This problem was often faced by manufacturers that made time-sensitive products such as fashion apparel and toys, which was sold only during specific seasons. Therefore, it was necessary for these manufacturers to make the production and distribution scheduling decisions. In addition, the customers often set due dates on the orders they placed with the manufacturer and there was typically a penalty imposed on the manufacturer if the orders were not completed and delivered to the customers on time. Hence, the manufacturer was incentivized to meet the due dates as close as possible.

With the rise of supply-chain management technology, a growing number of studies on production scheduling with simultaneous order distribution have appeared. Reference [27] presented a branch-and-bound algorithm for small-scale problems and a tabu search and a hybrid GA for large-scale problems of minimizing the total tardiness of a set of jobs to be scheduled on identical parallel machines for which jobs can only be delivered on certain fixed delivery dates. Reference [28] considered the production-distribution scheduling problem of minimizing total weighted delivery times on parallel BPMs with non-identical job sizes and equal processing time. They proposed a deterministic heuristic and two hybrid meta-heuristics based on MMAS to solve the problem. Reference [29] studied an integrated production and distribution problem considering homogenous vehicles with capacity constraints to minimize the total weighted delivery times of orders and presented an improved large neighborhood search algorithm with a local search. Reference [30] developed an approximation algorithm with a worst-case ratio of $3/2$ to minimize the total delivery times of jobs for the problem that considers both production and job delivery simultaneously with the consideration of machine availability. Reference

[31] dealt with the problem of minimizing total weighted delivery time of scheduling a set of jobs with identical processing time, non-identical sizes, and unequal weights on parallel BPMs with arbitrary capacities. They presented two heuristic algorithms and an algorithm based on ACO to address the problem. Reference [32] investigated several integrated problems of production, inventory, and delivery. After being processed, the orders were delivered to the customers by transporters at fixed delivery departure times. They considered two classes of problems based on whether the order delivery was splittable or not. For every class of problems, they analyzed the computational complexity, and pointed that each problem is NP-hard. For a detailed review on integrated scheduling, the reader is referred to [33, 34].

Since the completion time of each job in the production stage is the start time of jobs in the transportation phase for two-phase integration scheduling problems, the processing completion time of each job is diverse for different production scheduling schemes. Thus, in the transportation stage, jobs that are transported on each delivery are different and the distribution scheduling scheme obtained is also different. In addition, it is difficult to coordinate the integrated scheduling of the two stages of production and transportation, especially considering the parallel BPM environment in the production stage. Hence, the relationship between the production and transportation stages is rarely studied. In summary, studies on integrated scheduling considering TWT, especially on parallel non-identical BPMs with different job sizes, unequal job weights, and arbitrary job due dates, are still far fewer those of traditional objectives. Since the objective of TWT is widely applicable in the service and manufacturing industries, the integrated scheduling problems considering TWT are still worth further study.

3 Problem formulation

The problem studied in this paper can be denoted by $P_m \mid p\text{-}batch, delivery, p_j = p, w_j, s_j, d_j, S_i, CV, DT_l, VA_l \mid \sum_{j=1}^{n} w_j \cdot T_j$. It is assumed that all parameters are integers and known in advance. The assumptions of this problem are the following.

There are a set of n jobs, denoted by $J = \{J_j | j = 1, 2, ..., n\}$, to be grouped into b batches that are scheduled on m parallel BPMs, denoted by $M = \{M_1, M_2, \ldots, M_m\}$. Each job $J_j \in J$ has four attributes, i.e., processing time $p_j = p$, size s_j, weight w_j, and due date d_j. The generated batch set is denoted by $B = \{B_u | u = 1, 2, \ldots, b\}$. Each batch $B_u \in B$ generally includes more than one job. All jobs in the same batch are processed simultaneously without interruption. Note that b is not determined in advance. The capacity of the machines are non-identical. Each machine M_i has a capacity S_i. There are z different machine capacity, denoted by $S^1, S^2, \cdots, S^z$ $(1 \le z \le m)$. Without loss of generality, it can be assumed that $S^1 < S^2 < \cdots < S^z$. The number of machines whose capacity equals to S^ξ $(1 \le \xi \le z)$ is denoted by m_ξ, where $m_1 + m_2 + \cdots + m_z = m$ and $S_i \in \{S^1, S^2, \cdots, S^z\}$. The size of each job is not more than the maximum capacity of machines, i.e., S^z. All machines and jobs are available at time zero. In the studied problem, some jobs can only be processed on the machines with large capacity due to the constraint of machine capacity. Moreover, the total size of all jobs in one batch cannot exceed the capacity of the machine that processes the batch, and the k-th batch scheduled on M_i is denoted by B_{ki}. The processing time of the batch B_{ki}, denoted by P_{ki}, is equal to p.

Once the processing of a job is finished, this job is delivered to the customer by one of the vehicles at $d(l = 1, 2, \ldots, d)$ different departure times. At the l-th departure time, denoted by $DT_l (l = 1, 2, \ldots, d)$, there are VA_l vehicles to be used for transportation, and each vehicle has the identical capacity CV.

To describe the studied problem more accurately, the MIP model of this problem is presented in Appendix B.

In [1], the scheduling problem of minimizing makespan on a single BPM with arbitrary job sizes was studied and the proof of strong NP-hardness of the problem was provided. Since the due dates and weights of the jobs, as well as the capacity of machines, are all non-identical in the studied problem, which is more complicated than the problem in [1], the problem investigated in this paper is also NP-hard.

4 Proposed algorithm

Considering that the studied problem is NP-hard, a meta-heuristic based on ACO is proposed to address the studied problem. The ACO algorithm is inspired by the communication mechanism of ant foraging behavior, where individual ants communicate with each other by secreting pheromones. As the ants move, they leave pheromones in their path, and the ones behind choose their path by sensing the concentration of pheromones. Since the collective behavior of the ant colony shows the phenomena of positive feedback, self-organization, strong robustness, and distributed calculating. ACO has been successfully used to handle a great deal of complex discrete combinatorial optimization problems.

In this paper, the problem under study can be solved in two stages. In the production stage, the jobs are grouped into batches that are scheduled on the parallel BPMs. In the transportation stage, an effective heuristic method is proposed to obtain the optimal schedule.

This section contains the following contents. First, the solution construction scheme is introduced. Second, the way the pheromones update is presented. Third, to obtain the best solution, the local optimization strategy is developed. Fourth, a detailed description of the proposed algorithm is presented.

4.1 Solution construction

Encoding is the first step in ACO being applied to solve the combinatorial optimization problem. In this study, each solution is encoded as a $2 \times n$ vector, where n represents the number of jobs. For the production phase of the integrated scheduling problem, jobs must be grouped into batches before the jobs can be processed on the BPMs. Once a job is assigned to a batch, the batch index and machine index of the job are determined simultaneously.

To clearly describe the state of each job being accessed, an access list, denoted by AL, is employed to record the access state of each job, and the initial value of AL is set to $(1, 2, \ldots, n)$, indicating that all jobs are not scheduled. The number at the corresponding position on the access list becoming zero denotes that the job has been scheduled. After all jobs are processed, $AL = (0, 0, \ldots, 0)$, indicating that one schedule to be delivered in the stage of production is built.

4.1.1 Interval candidate list

In the solution construction (SC) algorithm of the studied problem, since the selection of the first job in the new empty batch on the machine has a great influence on the quality of solution. ICLs based on four attributes, i.e., capacity of batches, size of jobs, due dates of jobs, and departure times, are designed to reduce the search for the feasible solution space and improve the search speed. Specifically, there are three steps to construct the interval candidate list ICL_{lki}^1: (1) Build a set of jobs that meet the size of current batch B_{ki} on the machine M_i; (2) divide an interval into different sub-intervals according to the departure time; and (3) assign jobs to different sub-intervals according to their due dates. ICL_{lki}^1 is used to randomly select a job from the sub-intervals in order from front to back and assign it to the new empty batch B_{ki} on machine M_i. In addition, the process of constructing interval candidate list ICL_{lki}^2 is to remove jobs that do not meet the remaining space of the current batch B_{ki} on machine M_i based on ICL_{lki}^1. ICL_{lki}^2 is used to select a job from the sub-intervals in order from front to back and assign it to the current batch B_{ki} on machine M_i according to (7). ICL_{lki}^1 and ICL_{lki}^2 are defined as (1) and (2), respectively.

$$ICL_{lki}^1 = \{J_j | s_j \leq S_{ki} \wedge d_j \geq DT_l \wedge d_j \leq DT_{l+1}\} \tag{1}$$

$$ICL_{lki}^2 = \{J_j | s_j \leq (S_{ki} - \sum_{h \in B_{ki}} s_h) \wedge d_j \geq DT_l \wedge d_j \leq DT_{l+1}\} \tag{2}$$

where J_j in (1) and (2) represents any one unscheduled job.

4.1.2 Pheromone trails

Pheromone trails are of great importance in constructing solutions by ants. Each ant selects the unscheduled jobs and assigns the selected jobs into one batch based on the state-transition probability. Moreover, pheromones usually represent past experiences of the ants in building solutions. Moreover, an unsuitable definition of pheromones generally leads to poor solutions. In this study, pheromone ψ_{hj} represents the desirability to place jobs J_h and J_j in the same batch, and θ_{jki} represents the average desirability of candidate job J_j being assigned into batch B_{ki}. To improve the accuracy of the algorithm, θ_{jki} is defined as follows:

$$\theta_{jki} = \frac{\sum_{J_h \in B_{ki}} \psi_{hj}}{| B_{ki} |} \tag{3}$$

where ψ_{hj} denotes the expectation of candidate jobs J_j and J_h being grouped into the same batch; $| B_{ki} |$ denotes the number of jobs in B_{ki}.

4.1.3 Heuristic information

As a crucial component of the ACO algorithm, heuristic information enables guiding the search direction during the search process. Specifically, it can provide the problem-specific knowledge, thereby accelerating the convergence of the algorithm. In the problem described herein, jobs with large weight, tight due date, and small size should be processed with a higher priority. Thus, η_j^1 is defined as (4). Additionally, since the job with a tight due date should be selected with a higher probability, η_j^2 is defined as (5). To reduce the number of batches and make the jobs able to be processed as early as possible, each batch should be as full as possible to reduce the idle space of the batch on the machine. Therefore, the third heuristic information η_{jki}^3 is defined as (6):

$$\eta_j^1 = \frac{w_j}{d_j \cdot s_j + \gamma} \tag{4}$$

$$\eta_j^2 = \frac{1}{d_j} \tag{5}$$

$$\eta_{jki}^{3}=\frac{1}{S_i-(\sum_{J_h\in B_{ki}} s_h+s_j)+1} \tag{6}$$

where γ is a constant.

4.1.4 Decision rules

To build each batch, two decision rules are required, namely decision making in selecting the machine and job, respectively. In the decision rule for machine selection, the machine with the smallest completion time is selected to decrease the makespan of production. If more than one machine has the same smallest completion time, the machine with the smallest index number is selected. In the decision rule for job selection, when an empty batch is constructed on the selected machine, one unscheduled job is randomly selected from ICL_{lki}^{1} and added into the current batch as the first job first. Then, the unscheduled jobs are chosen from ICL_{jki}^{2} and added into the current batch one by one according to the state-transition probability, defined as (7):

$$p_{jki}=\begin{cases}\frac{(\theta_{jki})^{\alpha}\cdot(\eta_j^1)^{\beta 1}\cdot(\eta_j^2)^{\beta 2}\cdot(\eta_{jki}^3)^{\beta 3}}{\sum_{J_h\in ICL_{lki}^2}(\theta_{hki})^{\alpha}\cdot(\eta_h^1)^{\beta 1}\cdot(\eta_h^2)^{\beta 2}\cdot(\eta_{hki}^3)^{\beta 3}}, & \text{if } J_j\in ICL_{lki}^2\\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

According to (7), the selection of jobs is influenced by four parameters, i.e., α, β_1, β_2, and β_3, determining the relative importance of pheromone trails and heuristic information.

4.2 Update of pheromone trails

Pheromone update is a vitally significant step in ACO to explore and exploit better solutions and search the solution space efficiently. For ant colonies, pheromone update is used to dynamically adjust the search direction based on the ants' previous experiences. There are generally two types of strategies for updating pheromones, namely the local and global updates. In the local update, only the best solution found by the colony in the current generation is used to update pheromones. However, the best solution found from the first generation to the current generation is employed in the global update of pheromones. To decrease the complexity of the proposed algorithm and elevate the quality of solutions, only the global update strategy of pheromones is utilized in this study. Moreover, pheromones evaporate at a fixed rate to hinder them from accumulating indefinitely. Specifically, the pheromone trails are updated according to (8):

$$\psi_{hj}(t+1)=(1-\rho)\cdot\psi_{hj}(t)+\Delta\psi_{hj}(t)\cdot m_{hj}(t) \tag{8}$$

where ρ denotes the fixed evaporation rate, ranging from 0 to 1. $m_{hj}(t)$ represents the number of times that jobs J_h and J_j are grouped into the same batch at the current iteration t. $\Delta\psi_{hj}(t)$, representing the increment of pheromones generated by the global best solution at iteration t, is defined as (9):

$$\Delta\psi_{hj}(t)=\begin{cases}\frac{Q}{TWT^*(t)}, & \text{if jobs } J_h \text{ and } J_j \text{ are grouped into the same batch}\\ 0, \text{otherwise.}\end{cases} \tag{9}$$

In the t-th generation, if jobs J_h and J_j are assigned in the same batch, $\Delta\psi_{hj}(t)=Q/TWT^*(t)$, where Q is an input parameter, then $TWT^*(t)$ denotes the objective value of the global best solution at the t-th generation; that is, the minimum TWT is found at the t-th generation. If J_h and J_j are not grouped in the same batch, $\Delta\psi_{hj}(t)=0$.

4.3 Local optimization strategy

Ants can search for a feasible solution under the guidance of historical and heuristic information. However, it is difficult to explore the entire solution space exhaustively due to the complexity of the problem. Thus, it is quite possible to find a better neighbor solution of the constructed solution. To improve the local exploitation ability of the algorithm, and further search the neighborhood solution to improve the quality of the solution, a swap-based local optimization strategy, called LOC, is proposed. In LOC, the jobs in different batches are exchanged so that the jobs can be finished before or close to their due dates as much as possible. Before introducing the LOC, the following definition is provided.

Definition 1 For the studied problem, the sequence number of batches on a machine is called the rank of the batch. Furthermore, the first batch on each machine is called the first-rank batch, the second batch on each machine is called the second-rank batch, and so on.

Since jobs being processed and transported earlier benefits the objective, the jobs with larger weights and smaller due dates should be processed as early as possible. In other words, these jobs should be assigned lower-rank batches. In LOC, one job with smaller weight and larger due dates in one lower-rank batch is exchanged with one job with larger weight and smaller due date in the higher-rank batch under the constraint of machine capacity.

To make LOC more readable, an example is given as follows. In the production stage, suppose there are two machines and five jobs, the information of which is listed in Table 1. In Table 1, the first row represents the indices of the jobs, while the second, third, and fourth rows denote the weights, sizes, and delivery times of each job, respectively. In the transportation stage, it is assumed that there are two departure times, denoted by 3 and 6, there are two available vehicles, the capacity of which are five at every departure time, and the interval between the two departure times is three.

The distribution of jobs in batches before and after using the LOC algorithm are displayed in Figs. 1 and 2, respectively.

From Figs. 1 and 2, it can be found that job J_2 is in the second-rank batch and J_5 is in the first-rank batch. Moreover, $w_2 > w_5$ and $d_2 < d_5$. Therefore, J_2 is exchanged with J_5 after LOC is executed, causing the objective value of TWT to be reduced from 34 to 20.

The LOC algorithm described as Algorithm 1.

Algorithm 1 LOC.

```
1: Input one production schedule.
2: for $h_1 \leftarrow 1$ to $b-1$ do
3:     for $h_2 \leftarrow h_1+1$ to $b$ do
4:         for $(J_{e1} \in B_{h_1} \wedge J_{e2} \in B_{h_2})$ do
5:             if $((d_{e2} < d_{e1}) \wedge (w_{e2} > w_{e1}) \wedge (s_{e2} \leq S_{B_{h_1}} - \sum_{J_s \in B_{h_1}} s_s + s_{e1}) \wedge (s_{e1} \leq S_{B_{h_2}} - \sum_{J_r \in B_{h_2}} s_r + s_{e2}))$ then
6:                 Swap $J_{e1}$ and $J_{e2}$
7:             end if
8:         end for
9:     end for
10: end for
11: Output improved production schedule.
```

4.4 Description of proposed algorithm

To make the proposed algorithm easier to understand, the processes of algorithm SC and job transportation (JT) are described as Algorithms 2 and 3, respectively.

Algorithm SC consists of four parts. To begin, some parameters are initialized. Second, an empty batch is built on the selected machine and the first job for the current batch is chosen. Third, the unscheduled jobs are selected and grouped into the current batch based on the state-transition probability p_{jki}. Fourth, the solution to the production sub-problem is output.

Table 1 Infomation of jobs

j	1	2	3	4	5
w_j	3	5	2	4	1
s_j	1	3	2	3	4
d_j	4	2	3	1	5

Algorithm 2 SC.

```
1: $AL \leftarrow (1, 2, \ldots, n), ICL^1_{lki} \leftarrow \emptyset, ICL^2_{lki} \leftarrow \emptyset, C_i \leftarrow 0$.
   /* Initialization of access list, interval candidate list, and completion times of machines, respectively */
2: while $AL \neq (0, 0, \ldots, 0)$ do
3:     Sort machines in non-descending order of their completion times and select machine $M_i$ with the smallest completion time.
4:     Build new empty batch $B_{ki}$ on $M_i$, update $ICL^1_{lki}$.
5:     If $ICL^1_{lki} \neq \emptyset$, select a job randomly from $ICL^1_{lki}$, update $AL$.
6:     Update $ICL^2_{lki}$ for $B_{ki}$ according to (2).
7:     while $ICL^2_{lki} \neq \emptyset$ do
8:         Select jobs according to (7) from $ICL^2_{lki}$ and add selected jobs into $B_{ki}$.
9:         Update $ICL^2_{lki}$ and $AL$.
10:    end while
11:    Update $C_i$
12: end while
13: Output production schedule.
```

Algorithm 3 JT.

```
1: $Flag_j \leftarrow 0 (j = 1, ..., n), TWT \leftarrow 0$.
2: for $l \leftarrow 1$ to $d$ do
3:     $J_{trp} \leftarrow \emptyset$.
4:     for $j \leftarrow 1$ to $n$ do
5:         if $(c_j \leq DT_l) \wedge (Flag_j == 0)$ then
6:             $J_{trp} \leftarrow J_{trp} \cup J_j$
7:         end if
8:     end for
9:     Sort jobs in $J_{trp}$ in ascending order of their due dates.
10:    while $J_{trp} \neq \emptyset$ do
11:        Assign jobs in $J_{trp}$ into vehicles one by one on condition that constraint of vehicle capacity is satisfied.
12:        if $(d_j \leq DT_l)$ then
13:            $TWT \leftarrow TWT + w_j \cdot (DT_l - d_j)$
14:        end if
15:        $Flag_j \leftarrow 1$
16:        $J_{trp} \leftarrow J_{trp} \setminus J_j$
17:    end while
18: end for
19: Output solution.
```

Let $Flag_j$ denote the shipped states of jobs; that is, $Flag_j = 0$ indicates that job J_j is not shipped, while $Flag_j = 1$ means that J_j is shipped. Let c_j represent the

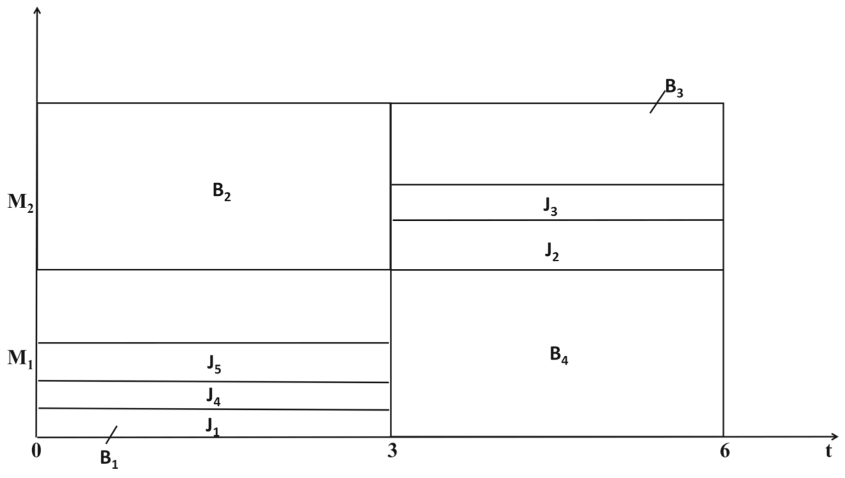

Fig. 1 Gantt chart of feasible schedule before using LOC

completion time of job J_j. J_{trp} denotes the set of jobs that are transported. Algorithm JT is described as Algorithm 3.

The proposed algorithm, i.e., ant colony optimization with local optimization (LACO), is described as Algorithm 4. In line 1, the parameters, such as the maximum iteration I_{max}; index of the current iteration, g; evaporation rate of pheromones, ρ; global best solution GS; number of ants, N_{ant}; and pheromone matrix, among others, are initialized separately. In lines 2–11, the best solution is generated. In line 4, Algorithm SC is called to construct a production schedule. In line 5, Algorithm LOC is called to improve the production schedule. In line 6, Algorithm JT is called to generate the solution. In lines 7 and 9, GS, the best solution from the first generation to the current generation, and pheromone trails, are updated separately. In line 12, the global best solution GS is output.

Algorithm 4 LACO.

```
 1: Parameter initialization:$I_{max}$, $g \leftarrow 1$, $GS \leftarrow +\infty$, the
    pheromone matrix;
 2: while $g \leq I_{max}$ do
 3:     for each ant do
 4:         Call SC to build production schedule;
 5:         Call LOC to improve production schedule;
 6:         Call JT to generate solution;
 7:         Update $GS$;
 8:     end for
 9:     Update pheromone matrix according to (8);
10:     $g++$;
11: end while
12: Output $GS$.
```

4.5 Complexity analysis

For each iteration of the proposed LACO algorithm, the computation time consists of three separate parts, i.e., building the batches, local optimization, and transporting the jobs, the complexities of which are $O(n^2)$, $O(b^2)$, and $O(d \cdot n^2)$, respectively. Since the maximum number of batches, b, is not more than the number of jobs, n, and the local optimization strategy is executed by every ant, the computational complexity of the LACO algorithm is $O(I_{max} \times N_{ant} \times d \times n^2)$, where I_{max}, N_{ant}, and d denote the maximum number of iterations, number of ants, and number of departure times, respectively.

5 Computational experiments

To evaluate the performance of the LACO algorithm comprehensively, it is compared with three other representative algorithms, i.e., PSO [35], the random-keys genetic algorithm (RKGA) [36], and SA [37]. To verify the effectiveness of the local optimization strategy in LACO, WACO (LACO without local optimal strategy) is also implemented as one of the benchmark algorithms. Moreover, to apply the benchmark algorithms to the problem studied in this paper, several modifications are required.

To make a fair comparison, the strategy of job transportation in all algorithms is the same. Additionally, the method of batching in this paper is incorporated into the benchmark algorithm SA, meanwhile, machine environment in [37] is modulated according to the studied problem.

All algorithms are programmed in C++ using the software Visual Studio 2017 and executed on a Windows 7 PC with an Intel Core i3 3.20GHz processor and 8 Gb of RAM.

5.1 Experimental designs

In this paper, eight categories of test instances are determined by the combination of four factors such as

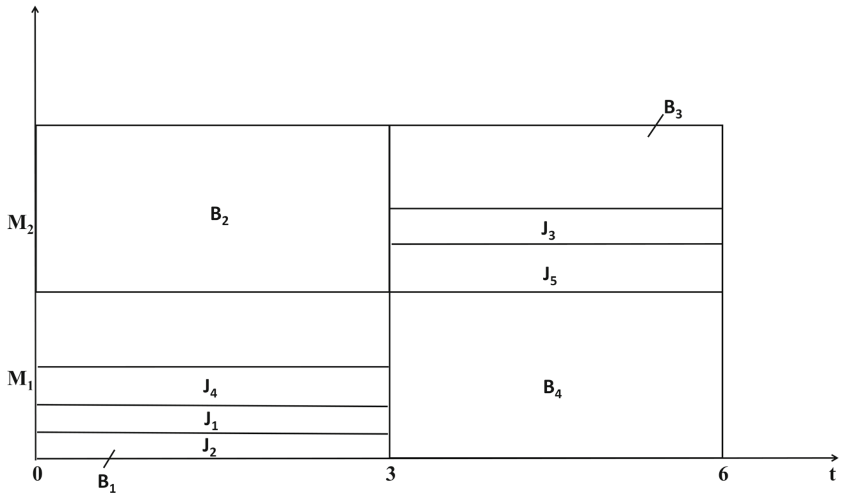

Fig. 2 Gantt chart of improved schedule after using LOC

number of jobs (n), number of machines (m), tightness of due dates (A), and departure modes (DW). Specifically, two combinatorial values of n and m are considered, i.e., ($n = 100, m = 5$) and ($n = 200, m = 10$), denoted by $NM1$ and $NM2$, respectively. Parameter A is set according to [38] to adjust the urgency degrees of the due dates of jobs. The value of A is selected from set $\{0.5, 1\}$, denoted by $A1$ and $A2$, respectively. Two kinds of departure modes are designed, denoted by $DW1$ and $DW2$, respectively. Hence, eight categories of test instances are generated totally, denoted by $NM1A1DW1$, $NM1A1DW2$, $NM1A2DW1$, $NM1A2DW2$, $NM2A1DW1$, $NM2A1DW2$, $NM2A2DW1$, and $NM2A2DW2$, respectively. For each category, ten instances are generated randomly, according to the settings in Table 2, where P and U represent poisson distribution and uniform distribution, respectively.

To test the effect of the number of jobs on the performance of the LACO algorithm, two levels of the number of jobs are considered. The processing times of jobs (p_j), the weights of jobs (w_j), the types of machine capacities (S^{ξ}), and the number of machines (m) are set according to the method in [31]. The due date of each job (d_j) is generated by $U[2p, A \times C_{max}]$, where C_{max} denotes the makespan obtained by the first fit (FF) rule. The sizes of jobs are set according to the method in [39]. In the distribution stage, the capacity of each vehicle (CV) is set to 80. The vehicles are set to depart at d different departure times, where d is determined until all jobs have been transported in the transportation stage. At the l-th departure time, denoted by $DT_l (l = 1, 2, \cdots, d)$, the corresponding number of vehicles is denoted by VA_l. Additionally, in $DW1$, the number of vehicles for departure times with odd and even indexes is generated by $U[1, 2]$ and $U[3, 4]$ for 100 jobs, respectively. Correspondingly, the number of vehicles for departure times with odd and even indexes is generated by $U[3, 4]$ and $U[4, 5]$ for 200 jobs, respectively. In $DW2$, the number of vehicles for each departure time is set by $U[2, 3]$ for 100 jobs and $U[4, 5]$ for 200 jobs, respectively.

Table 2 Factors and levels of the test instances

Factors	Levels
n	$N1 = 100, N2 = 200$
$m(m_1, m_2)$	$5(3, 2), 10(6, 4)$
p_j	3
s_j	$s_j \sim P(\lambda_q), \lambda_1 = 5, \lambda_2 = 12.5$
w_j	$U[1,10]$
A	$A1 = 0.5, A2 = 1$
d_j	$U[2p, A \times C_{max}]$
S^{ξ}	$S^1 = 10, S^2 = 25$
CV	80
DT_l	$DT_1 = 6 \times 1, DT_2 = 6 \times 2, \ldots, DT_d = 6 \times d$
$VA_{1,3,\ldots,d-1}$	$U[1, 2]$ for $N1DW1$, $U[3, 4]$ for $N2DW1$
$VA_{2,4,\ldots,d}$	$U[3, 4]$ for $N1DW1$, $U[4, 5]$ for $N2DW1$
$VA_{1,2,\ldots,d}$	$U[2, 3]$ for $N1DW2$, $U[4, 5]$ for $N2DW2$

5.2 Parameter settings

In the proposed LACO algorithm, the setting of several parameters has great influence on solution quality and computation time. Therefore, to achieve better performance, it is necessary to determine appropriate values for these parameters. The LACO parameters include N_{ant}, I_{max}, ρ, α, β_1, β_2, and β_3. To balance between solution quality and computation time, the values of several parameters are set, according to the preliminary experiments, as follows: $N_{ant} = 20$, $I_{max} = 200$, $\rho = 0.5$, and $Q = 1$. Meanwhile, to compare the algorithms fairly, the number of particles, number of chromosomes, and number of iterations at the same temperature are set to 20 in PSO, RKGA, and SA, respectively; that is, the number of fitness function evaluations in each iteration for all algorithms is 20. Moreover, the maximum number of iterations for all algorithms is set to 200. Therefore, the numbers of fitness function evaluations of all the algorithms are the same and equal to 4,000.

To determine the effect of the values of the other four key parameters on the performance of LACO, the Taguchi design-of-experiment (DOE) method is employed. The combinations of selected values of these parameters are listed in Table 3.

To determine the effect of the four parameters, i.e., α, β_1, β_2, and β_3, on pheromone trails and heuristic information, the values of pheromones and heuristic information are set within $(0, 1)$. Moreover, the input parameter γ is set to 5 to ensure the value of heuristic information η_j^1 is within $(0, 1)$. As shown in Table 3, five levels of values are set for each parameter. As a result, an orthogonal array $L_{25}(5^4)$ is selected. To test the robustness of the proposed LACO algorithm and simplify the simulated experiments, one group of representative instances, denoted by $NM2A1DW1$, is chosen, in which 200 jobs with urgent due dates are processed on 10 machines and transported in the first mode of departure. Furthermore, five out of 10 instances are randomly selected and each selected instance is independently tested 10 times. Finally, the average TWT value obtained on each instance is regarded as the response variable (RV). Apparently, the smaller the value of RV, the better the combinatorial values of the four parameters. The orthogonal array and obtained RV values are listed in Table 4.

The average RV values on the numbers in the rows with the same level of each parameter in Table 4 are listed in Table 5 and illustrated in Fig. 3. In Table 5, $Delta$ represents the difference between the maximum and the minimum numbers in the rows for each column. The numbers in the row of $Ranks$ are determined according to decreasing order of $Delta$. The smaller the value of $Ranks$, the greater the influence of the corresponding parameter on the objective value.

From Fig. 3 and Table 5, it can be seen that β_1 is the most significant one among the four parameters. Moreover, a smaller value of β_1 can degrade the performance of the algorithm. This is because the jobs with loose due dates and larger weights being given higher priorities to be processed and transported, which can result in greater delays for the jobs with tight due dates. As for β_3, if it is too large the idle space in each batch on the machine increases. On the one hand, if the value of α is set to be too large, the algorithm converges faster without guaranteeing better solution quality. On the other hand, a smaller value of α can cause the solution space searched by the algorithm to decrease. Therefore, to ensure better performance of LACO, an appropriate value for α is required. It can be seen that the influence of β_2 on the algorithm is insignificant, so

Table 3 Combinations of parameter values

Parameters	Factor levels				
	1	2	3	4	5
α	1/5	1/3	1	3	5
β_1	1/5	1/3	1	3	5
β_2	1/5	1/3	1	3	5
β_3	1/5	1/3	1	3	5

Table 4 Orthogonal array and RV values

Test indexes	Paramter values				RV
	α	β_1	β_2	β_3	
1	1	1	1	1	31212.1
2	1	2	2	2	31234.5
3	1	3	3	3	31241.5
4	1	4	4	4	31272.4
5	1	5	5	5	31242.6
6	2	1	2	3	33770.8
7	2	2	3	4	34041.2
8	2	3	4	5	33804.6
9	2	4	5	1	29426.9
10	2	5	1	2	29329.0
11	3	1	3	5	34315.6
12	3	2	4	1	31349.1
13	3	3	5	2	30610.5
14	3	4	1	3	29268.6
15	3	5	2	4	30033.6
16	4	1	4	2	31572.1
17	4	2	5	3	32343.5
18	4	3	1	4	33283.1
19	4	4	2	5	32513.2
20	4	5	3	1	29196.6
21	5	1	5	4	33846.8
22	5	2	1	5	34086.3
23	5	3	2	1	30037.9
24	5	4	3	2	29459.9
25	5	5	4	3	29433.9

Table 5 Average response values

Levels	α	β_1	β_2	β_3
1	31240.62	32943.48	31435.82	30244.52
2	32074.50	32610.92	31518	30441.2
3	31115.48	31795.52	31650.96	31211.66
4	31781.70	30388.2	31486.42	32495.42
5	31372.96	29847.14	31494.06	33192.46
Delta	959.02	3096.34	215.14	2947.94
Ranks	3	1	4	2

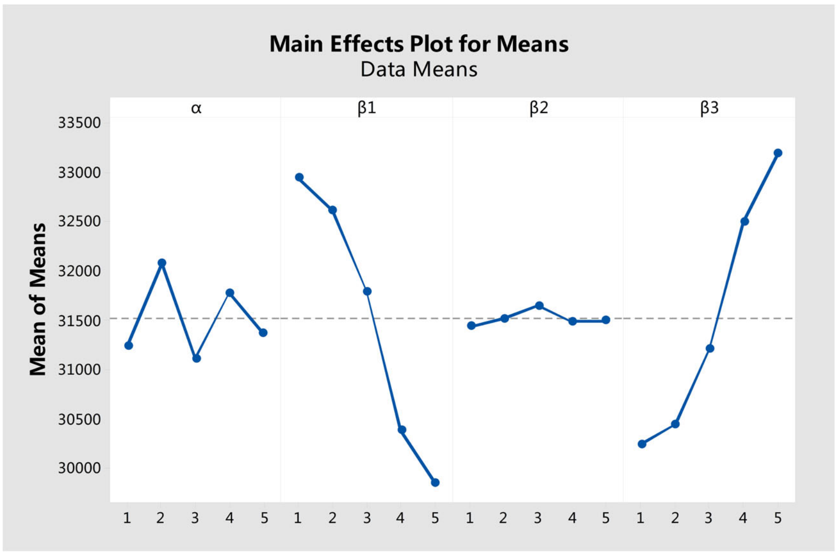

Fig. 3 Trends of factor levels of α, $\beta 1$, $\beta 2$, and $\beta 3$

only a reasonable value is preferred. According to the above analysis, the values of the four parameters are set to $\alpha = 1$, $\beta_1 = 5$, $\beta_2 = 0.2$, and $\beta_3 = 0.2$, which are used in the following experiments.

The parameters of all the benchmark algorithms are set as in Table 6. As shown in Table 6, for each algorithm, the maximum iteration number I_{max} is set to 200. In PSO, n_{pop} denotes the number of particles, and c_1 and c_2 denote the cognitive and social learning coefficients, respectively. w and α denote the inertia weight and decrement factor, respectively. Meanwhile, the range of velocity and position for each particle are set within [-4.0,4.0] and [0.0,4.0], respectively. There are three main parameters in RKGA, i.e., population size N, crossover probability p_c, and mutation probability p_m. In SA, K and T_{min} denote the iterative number under the same temperature and lower bound of the temperature, respectively. At the same time, θ denotes the cooling rate. Additionally, the values of other parameters of the benchmark algorithms are set according to their original papers [35–37].

Table 6 Parameter settings

PSO	RKGA	SA	LACO
$n_{pop} = 20$	$N = 20$	$K = 20$	$N_{ant} = 20$
$I_{max} = 200$	$I_{max} = 200$	$I_{max} = 200$	$I_{max} = 200$
$c_1 = 1$	$p_c = 0.5$	$\theta = 0.95$	$\rho = 0.5$
$c_2 = 1$	$p_m = 0.2$	$T_{min} = 10^{-6}$	$Q = 1$
$w = 0.6$			$\alpha = 1$
$\alpha = 0.99$			$\beta_1 = 5$
$V_{max} = 4.0$			$\beta_2 = 0.2$
$V_{min} = -4.0$			$\beta_3 = 0.2$
$X_{max} = 4.0$			
$X_{min} = 0.0$			

5.3 Evaluation metrics

To guarantee the fairness of comparison between the algorithms, the following performance metrics are adopted.

(1)Relative distance (R_{Alg}): This metric is used to measure the quality of the obtained solution [31]. Specifically, R_{Alg} indicates the distance between the obtained solution and best objective value, denoted by TWT^{Best}, among those obtained by all algorithms, including LACO, WACO, PSO, RKGA, and SA, to each test instance. R_{Alg} is formulated as (10):

$$R_{Alg}(\%) = \frac{TWT^{Alg} - TWT^{Best}}{TWT^{Best}} \times 100 \tag{10}$$

where TWT^{Alg} denotes the average value of each algorithm on its 10 runs Alg for each instance. It is clear that the smaller the value of R_{Alg}, the closer the objective value of algorithm Alg to the TWT^{Best}.

(2) Run time (t): This metric is generally used to compare the time performance of different algorithms under the same execution environment.

5.4 Experimental results and analysis

The experimental results of different combinations are shown in Tables 7–14. In each table, the first column represents the indexes of instances in the corresponding instance group. Each algorithm is run on each instance 10 times and the average values of $\bar{R}$ and $\bar{t}$ are calculated to measure the performance of the algorithms. The last row denotes the average results on the 10 instances of each instance group. Note that the best result of $\bar{R}$ on each test instance is shown in bold type.

Table 7 Average results on $NM1A1DW1$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**4.703**	1.99	9.260	1.78	6.126	0.71	23.263	1.26	26.672	4.60
2	**2.269**	2.03	8.261	1.78	15.819	0.36	25.544	1.25	21.215	4.79
3	**3.493**	2.06	8.501	1.77	5.873	0.53	14.227	1.26	16.935	5.03
4	**2.227**	2.00	6.053	1.73	13.612	0.30	19.029	1.30	23.501	4.65
5	**3.769**	2.09	9.285	1.78	15.420	0.30	35.003	1.36	31.510	4.47
6	**2.998**	2.06	8.771	1.75	7.555	0.30	21.785	1.43	17.659	4.47
7	6.798	2.16	12.162	1.89	**6.512**	0.29	21.216	1.48	22.009	4.45
8	**2.363**	2.05	6.005	1.74	5.874	0.31	24.083	1.55	23.415	4.55
9	**0.988**	1.99	6.168	1.68	12.527	0.29	23.163	1.39	29.647	4.58
10	**4.797**	1.93	9.291	1.64	6.168	0.30	17.584	1.34	19.763	4.46
avg	**3.440**	2.04	8.376	1.75	9.549	0.37	22.490	1.36	23.233	4.61

Table 8 Average results on $NM1A1DW2$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**4.229**	1.97	9.535	1.79	13.034	0.29	27.011	1.32	23.397	4.33
2	**2.238**	2.01	7.916	1.80	14.839	0.26	27.501	1.35	25.144	4.33
3	**3.864**	2.04	8.562	1.77	16.566	0.25	28.416	1.36	26.895	4.27
4	**2.936**	2.00	6.886	1.73	12.330	0.26	19.874	1.32	21.611	4.37
5	**3.741**	2.09	9.135	1.78	15.631	0.26	27.273	1.32	34.745	4.32
6	**4.673**	2.04	7.272	1.76	12.353	0.26	24.763	1.32	18.166	4.29
7	**1.090**	2.15	6.298	1.89	3.101	0.26	15.676	1.32	24.185	4.28
8	**5.059**	2.04	10.279	1.76	15.729	0.26	36.162	1.33	40.382	4.31
9	**4.202**	1.98	8.303	1.69	15.610	0.25	28.511	1.34	35.842	4.37
10	**2.772**	1.92	6.888	1.65	15.718	0.26	19.122	1.33	23.241	4.25
avg	**3.481**	2.02	8.107	1.76	13.491	0.26	25.431	1.33	27.361	4.31

Table 9 Average results on $NM1A2DW1$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**12.576**	1.63	17.102	1.50	139.932	0.26	99.644	1.51	115.051	4.51
2	**5.007**	1.68	14.779	1.57	141.469	0.26	105.906	1.37	104.736	4.51
3	**3.662**	1.62	7.767	1.53	47.463	0.26	54.745	1.37	52.471	4.54
4	**2.529**	1.62	8.581	1.45	62.153	0.26	63.880	1.36	55.019	4.53
5	**6.985**	1.57	18.371	1.44	134.663	0.26	144.363	1.36	179.419	4.53
6	**8.876**	1.63	13.618	1.47	100.065	0.30	110.788	1.37	95.220	4.48
7	**6.494**	1.62	11.950	1.44	71.290	0.34	92.833	1.36	95.961	4.58
8	**2.692**	1.67	8.431	1.49	48.371	0.34	38.477	1.35	37.866	4.56
9	**16.502**	1.63	19.040	1.44	128.189	0.33	101.893	1.36	117.366	4.52
10	**4.975**	1.60	7.250	1.45	44.737	0.33	54.844	1.38	35.246	4.54
avg	**7.030**	1.63	12.689	1.48	91.833	0.29	86.737	1.38	88.836	4.53

Table 10 Average results on $NM1A2DW2$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**9.730**	1.64	20.180	1.48	185.383	0.30	148.401	1.36	136.171	4.35
2	**8.938**	1.70	15.852	1.55	163.912	0.30	134.958	1.40	180.675	4.36
3	**4.328**	1.67	6.968	1.48	73.435	0.30	77.429	1.41	78.469	4.37
4	**4.605**	1.67	7.580	1.42	63.600	0.30	61.620	1.35	65.054	4.35
5	**6.386**	1.63	8.768	1.35	153.491	0.32	140.411	1.35	148.131	4.37
6	**8.644**	1.66	11.034	1.41	107.866	0.30	104.268	1.34	115.570	4.33
7	**5.147**	1.61	6.399	1.39	78.253	0.32	103.740	1.35	97.913	4.38
8	**17.604**	1.66	27.150	1.44	78.797	0.33	68.873	1.34	79.432	4.37
9	**15.430**	1.64	22.341	1.42	131.848	0.31	99.676	1.34	108.801	4.47
10	**4.746**	1.60	8.145	1.41	60.386	0.33	83.446	1.38	83.315	4.47
avg	**8.556**	1.65	13.442	1.43	109.697	0.31	102.282	1.36	109.353	4.38

Table 11 Average results on $NM2A1DW1$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**2.025**	7.33	8.531	6.71	38.914	0.82	30.034	5.01	34.779	13.13
2	**1.224**	7.43	5.618	6.78	16.104	0.85	21.094	4.48	27.763	12.82
3	**0.825**	7.34	5.682	6.63	10.988	0.86	22.496	4.43	22.922	12.77
4	**1.492**	7.06	4.256	6.39	17.097	0.82	18.693	4.38	13.769	12.69
5	**1.427**	7.24	8.482	6.53	32.351	0.86	22.059	4.42	24.853	12.74
6	**2.384**	7.02	9.003	6.34	35.006	1.09	35.927	4.38	34.604	12.90
7	**1.445**	7.16	6.346	6.37	12.066	0.92	29.035	4.37	31.031	12.76
8	**2.111**	7.03	7.502	6.31	19.205	0.92	20.235	4.38	20.120	12.79
9	**1.074**	7.24	5.096	6.60	16.015	0.90	25.151	4.42	25.155	12.69
10	**1.049**	7.25	7.277	6.50	20.083	0.90	31.640	4.39	32.965	12.96
avg	**1.506**	7.21	6.779	6.52	21.783	0.89	25.636	4.47	26.796	12.82

Table 12 Average results on $NM2A1DW2$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**2.647**	7.60	9.312	6.78	41.858	0.82	35.531	4.63	29.885	13.01
2	**2.516**	7.52	8.244	6.83	22.074	0.84	32.497	4.47	39.032	13.23
3	**1.724**	7.36	8.881	6.72	16.430	0.98	26.677	4.32	28.087	13.17
4	**1.663**	7.06	7.897	6.46	28.795	0.94	25.292	4.32	27.639	13.00
5	**1.360**	7.09	8.543	6.43	34.021	1.05	25.445	4.36	21.352	13.72
6	**1.992**	7.00	7.080	6.44	34.821	0.95	34.173	4.32	36.200	12.67
7	**1.642**	6.96	7.209	6.43	18.319	0.95	29.548	4.32	34.221	13.79
8	**1.559**	6.94	8.731	6.30	24.000	0.92	25.720	4.31	26.285	13.48
9	**1.339**	7.12	7.265	6.50	18.526	0.94	19.212	4.41	22.363	13.27
10	**2.189**	7.03	7.542	6.37	23.826	0.92	32.313	4.38	34.402	13.18
avg	**1.863**	7.17	8.070	6.53	26.267	0.93	28.641	4.38	29.947	13.25

Table 13 Average results on $NM2A2DW1$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**2.061**	5.82	14.201	5.52	156.672	0.95	105.838	4.69	91.345	13.77
2	**6.978**	5.73	20.529	5.48	187.252	0.94	169.273	4.68	70.727	13.74
3	**10.158**	5.36	19.651	5.16	92.757	1.06	100.889	4.51	94.766	14.83
4	**14.840**	5.35	23.462	5.32	202.906	0.96	131.175	4.35	80.021	13.71
5	**11.525**	5.35	20.568	5.31	239.653	0.97	134.543	4.38	156.362	12.75
6	**7.799**	5.30	13.012	5.18	149.900	1.02	140.139	4.84	100.772	12.86
7	**4.743**	5.36	9.410	5.14	138.569	0.97	138.750	4.74	82.431	12.85
8	**3.496**	5.56	8.169	5.38	96.680	0.94	67.629	4.79	51.791	12.75
9	**3.727**	5.36	12.099	5.07	122.783	0.95	102.976	4.85	103.993	12.56
10	**8.779**	5.77	15.219	5.35	152.450	0.93	115.128	4.94	148.994	12.63
avg	**7.411**	5.50	15.632	5.29	153.962	0.97	120.634	4.68	98.120	13.25

Table 7 shows the experimental results on the instances with $n = 100$, $m = 5$, $A = 0.5$, and $DW1$, denoted by $NM1A1DW1$. Table 8 lists the results on instance group $NM1A1DW2$. It can be observed from Tables 7 and 8 that the LACO algorithm can obtain the smallest $\bar{R}$ of any other benchmark algorithm, except for the seventh instance in $NM1A1DW1$, no matter what the departure mode is. This shows that the quality of the solution obtained by LACO is better than that of each other benchmark algorithm. In other words, the solution found by LACO is much closer to the best solution that can be found by all algorithms for each instance. From Tables 7 and 8, it can be found that the performance of each algorithm on instances with the first mode, i.e., $DW1$, is different from that on the mode of $DW2$. Specifically, the average values of $\bar{R}$ in Table 7 are slightly lower than those in Table 8. Although the computation time of PSO is obviously less than that of LACO, the solution quality of PSO is much worse than that of LACO. This may be because the strategies adopted in LACO, such as the local optimization, are relatively more deliberate.

It can be seen from Tables 7 and 8 that the difference between the runtime of LACO and that of WACO is very small, meaning that the computation time of the local optimization strategy is negligible. Furthermore, the solution quality of LACO slightly outperforms that of WACO. This means that the performance of LACO is better than that of WACO as a whole.

Tables 9 and 10 present the results obtained by the algorithms for instances with 100 jobs, five machines, A=1, and combined with two modes of departure. It can be observed from Tables 9 and 10 that the average values of $\bar{R}$ obtained by LACO in Tables 9 and 10 are larger than the corresponding average values of $\bar{R}$ in Tables 7 and 8, respectively. The reason may be that the delays of most jobs become smaller when the value of A increases, which

Table 14 Average results on $NM2A2DW2$

Ins.	LACO		WACO		PSO		RKGA		SA	
	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$	$\bar{R}(\%)$	$\bar{t}(s)$
1	**6.553**	5.89	19.087	5.49	165.493	1.06	114.931	4.55	93.545	13.55
2	**10.058**	5.80	18.709	5.38	202.388	1.09	157.146	4.53	147.573	13.73
3	**4.357**	5.43	10.622	5.11	77.879	0.99	79.077	4.44	72.988	13.78
4	**7.157**	5.44	15.278	5.25	174.148	0.96	114.878	4.37	134.080	13.03
5	**16.796**	5.47	26.908	5.27	265.996	0.97	144.983	4.37	152.314	12.48
6	**14.551**	5.40	18.679	5.17	166.052	0.99	129.176	4.45	145.604	12.82
7	**11.045**	5.37	18.692	5.11	162.038	0.96	140.767	4.37	149.451	13.86
8	**3.892**	5.58	7.452	5.35	97.598	0.96	75.742	4.38	82.717	13.72
9	**3.260**	5.33	10.640	5.03	121.849	0.97	96.811	4.38	94.475	13.68
10	**2.135**	5.63	6.018	5.32	124.055	0.96	96.794	4.42	92.523	13.30
avg	**7.980**	5.53	15.208	5.25	155.750	0.99	115.030	4.43	116.527	13.40

results in a decrease in objective value. According to (10), the average values of $\bar{R}$ become slightly larger.

Tables 11–14 provide the experimental results on 200 jobs and 10 machines with $A1DW1$, $A1DW2$, $A2DW1$, and $A2DW2$. It can be observed from the four tables that, as the number of jobs increases, all algorithms take slightly longer to find the solutions, which is still acceptable. At the same time, the average values of $\bar{R}$ obtained by the LACO algorithm are still better than those of the other benchmark algorithms. Additionally, according to the experimental results in Tables 11 and 13, it can be seen that the average values obtained by all algorithms in Table 11 are slightly better than those in Table 13. The reason for this phenomenon may be due to the different setting of parameter A. Furthermore, since the value of parameter A in Table 10 is less than that in Table 12, there are more jobs with tight due dates in instance $A1DW1$ than in instance $A2DW1$, which results in the target values obtained in instance $A1DW1$ being greater than those in instance $A2DW1$. According to (10), one can conclude that the average $\bar{R}$ values of instance $A1DW1$ are slightly less than those of instance $A2DW1$. The comparative results in Tables 12 and 14 are consistent with those in Tables 11 and 13, respectively.

It is remarkable that the average values of $\bar{R}$ obtained by all algorithms on instances with a $DW1$ mode are slightly better than those of all algorithms on a $DW2$ mode, according to Tables 7–14, which means that the first departure mode is superior to the second mode for the studied problem. Moreover, it can be seen from each table that the average values of $\bar{R}$ obtained by LACO are slightly better those of WACO, due to the WACO algorithm being unable to find a better schedule of assigning jobs on machines than the LACO algorithm. In other words, in the solution found by WACO, more jobs with loose due dates or small weights are processed in advance, leading to the jobs with tight due dates or large weights being delayed for processing. Therefore, the objective values obtained by the WACO algorithm is slightly larger than that of the LACO algorithm.

In all, according to the results of comparative experiments, the LACO algorithm is able to find better solutions than any other benchmark algorithm, although the computation time it takes is slightly longer than that of the WACO algorithm, PSO algorithm, and RKGA. Moreover, the computation time of the LACO algorithm is acceptable.

6 Conclusions

In this paper, an algorithm based on ACO, called LACO, is proposed to address the integrated production distribution scheduling problem on parallel BPMs and to minimize the TWT of jobs. In the production stage, the jobs are processed on the BPMs with non-identical capacities. In the delivery phase, the jobs having finished their processing are transported by vehicles with equal capacity. A new local optimization strategy called LOC is proposed to improve the local exploitation ability of the algorithm, and further search the neighborhood solution to improve the quality of the solution. Two interval candidate lists are proposed to reduce the search for the feasible solution space and improve the search speed. Additionally, three objective-oriented heuristics are developed to accelerate the convergence of the algorithm. To verify the performance of the proposed algorithm, several algorithms are compared with LACO on test instances designed deliberately. Experimental results show that the proposed LACO can offer better solutions than the other four benchmark algorithms in addressing the studied problem.

In future research, the studied problem can be extended to some other variations that are interesting and closer to reality. However, the extended problems may be more complex. For example, the production phase can be extended to more complex environments, such as unequal processing speeds of machines and jobs with dynamic release times. Another more realistic extension is to extend the equal capacity of vehicles to non-identical vehicle capacity. Additionally, other objectives could also be considered, such as the maximum tardiness and total revenue.

Appendix A: Abbreviations

Abbreviation	Method
ABC	Artificial bee colony
ACO	Ant colony optimization
AL	Access list
ALNS	Adaptive large neighborhood search
BFLPT	Best fit longest processing time
BPM	Batch processing machine
BSP	Batch scheduling problem
CRO	Chemical reaction optimization
DOE	Design of experiment
EI	Emergent intelligence
E/T	Earliness-tardiness
FF	First fit
FFD	First-fit-decreasing
FFLPT	First-fit longest processing time
GA	Genetic algorithm
GRASP	Greedy randomized adaptive search procedure

GS	Global best solution
ICL	Interval candidate list
JT	Job transportation
LOC	Local optimization strategy
LACO	Ant colony optimization with local optimization
MARB	Minimum attribute ratio of the batch
MIP	Mixed-integer programming
MMAS	Max-min ant system
MTO	Make-to-order
PSO	Particle swarm optimization
RKGA	Random-keys genetic algorithm
RV	Response variable
SA	Simulated annealing
SC	Solution construction
TCT/TWCT	Total (weighted) completion time
TWT	Total weighted tardiness
VNS	Variable neighborhood search
WACO	LACO without local optimal strategy

Appendix B: Mixed-integer programming model

To present the MIP model of the studied problem, the parameters and decision variables are listed as follows:

Parameters:

j	index of each job, $j = 1, 2, \dots, n$
i	index of each machine, $i = 1, 2, \dots, m$
k	index of each batch, $k = 1, 2, \dots, b$
l	index of each departure time, $l = 1, 2, \dots, d$
p_j	processing time of job J_j
s_j	size of job J_j
d_j	due date of job J_j
S_i	capacity of machine M_i
w_j	weight of job J_j
CV	capacity of vehicle
DT_l	the l-th departure time
VA_l	number of vehicles depart at the l-th departure time
D_j	departure time of job J_j
P_{ki}	processing time of the k-th batch on machine M_i
CT_{ki}	completion time of the k-th batch on machine M_i
ST_{ki}	start time of processing for the k-th batch on machine M_i
c_j	completion time of job J_j
C_i	completion time of machine M_i

Decision variables:

$$X_{ki} = \begin{cases} 1, & \text{If batch } B_k \text{ is assigned to machine } M_i \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

$$Y_{jki} = \begin{cases} 1, & \text{If job } J_j \text{ is assigned to the } k\text{th batch on machine } M_i \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

$$Z_{jl} = \begin{cases} 1, & \text{If job } J_j \text{ is transported at the } l\text{th departure time } DT_l \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The MIP model of this problem is formulated as follows:

$$\text{Minimize} \quad TWT = \sum_{j=1}^{n} w_j \cdot T_j \tag{14}$$

Subject to.

$$\sum_{i=1}^{m} X_{ki} = 1 \quad k = 1, 2, \dots, b \tag{15}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{b} Y_{jki} = 1 \quad j = 1, 2, \dots, n \tag{16}$$

$$\sum_{j=1}^{n} s_j \cdot Y_{jki} \le S_i \quad k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{17}$$

$$P_{ki} = p_j \cdot Y_{jki} \quad j = 1, 2, \dots, n; k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{18}$$

$$X_{ki} \ge Y_{jki} \quad j = 1, 2, \dots, n; k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{19}$$

$$ST_{ki} = CT_{k-1,i} \quad k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{20}$$

$$CT_{ki} = ST_{ki} + P_{ki} \quad k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{21}$$

$$ST_{1i} = 0 \quad i = 1, 2, \dots, m \tag{22}$$

$$CT_{ki} \le DT_d \quad k = 1, 2, \dots, b; i = 1, 2, \dots, m \tag{23}$$

$$\sum_{l=1}^{d} Z_{jl} = 1 \quad j = 1, 2, \dots, n \tag{24}$$

$$\sum_{j=1}^{n} s_j \cdot Z_{jl} \le CV \cdot VA_l \quad l = 1, 2, \dots, d \tag{25}$$

$$DT_l = D_j \cdot X_{ki} \cdot Y_{jki} \cdot Z_{jl} \quad j = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, b; i = 1, 2, \ldots, m; l = 1, 2, \ldots, d \qquad (26)$$

$$\sum_{k=1}^{b} X_{ki} \geq 1 \quad i = 1, 2, \ldots, m \qquad (27)$$

$$T_j = max\{0, D_j - d_j\} \quad j = 1, 2, \ldots, n \qquad (28)$$

$$X_{ki}, Y_{jki}, Z_{jl} \in \{0, 1\} \qquad (29)$$

The objective function (14) is used to minimize the TWT of jobs. Constraints (15) ensure that each batch can only be assigned to one machine. Constraints (16) ensure that each job can only be assigned into one batch on one machine. Constraints (17) indicate that the total size of all jobs in one batch cannot exceed the capacity of the machine that processes this batch. Constraints (18) define the processing time of the batch. Constraints (19) guarantee that each job can only be assigned into a batch after it has been created. Constraints (20) define the start processing time of a batch. Constraints (21) formulate that the completion time of each batch equals the sum of the start time and processing time of the batch. Constraints (22) define that the start time of the first batch on each machine is zero. Constraints (23) ensure that the completion time of each batch does not exceed the last delivery time. Constraints (24) indicate that each job can only be transported once. Constraints (25) guarantee that the sum of the size of jobs transported at each departure does not exceed the total capacity of the corresponding vehicles. Constraints (26) determine the departure time of each job. Constraints (27) ensure that the number of the batches processed on each machine is not less than one. Constraints (28) define the tardiness time of each job. Constraint (29) define the binary restriction on the decision variables.

Acknowledgements This work is supported by the National Natural Science Foundation under grants 71971002, 71871076 and 61872002, the Humanity and Social Science Youth Foundation of Ministry of Education of China under grant 15YJC630041, the Natural Science Foundation of Anhui Provincial Department of Education under grant KJ2015A062.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Uzsoy R (1994) Scheduling a single batch processing machine with non-identical job sizes. Int J Prod Res 32(7):1615–1635
2. Mathirajan M, Sivakumar AI (2006) A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int J Adv Manuf Technol 29(9-10):990–1001
3. García JM, Lozano S (2005) Production and delivery scheduling problem with time windows. Comput Ind Eng 48(4):733–742
4. Devapriya P, Ferrell W, Geismar N (2006) Optimal fleet size of an integrated production and distribution scheduling problem for a perishable product. In: 2006 IIE annual conference. Institute of Industrial and Systems Engineers
5. Muter I (2020) Exact algorithms to minimize makespan on single and parallel batch processing machines. Eur J Oper Res 285(2):470–483
6. Beldar P, Costa A (2018) Single machine batch processing problem with release dates to minimize total completion time. Int J Ind Eng Comput 9(3):331–348
7. Cheng B, Wang Q, Yang S, Hu X (2013) An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes. Appl Soft Comput 13(2):765–772
8. Al-Salamah M (2015) Constrained binary artificial bee colony to minimize the makespan for single machine batch processing with non-identical job sizes. Appl Soft Comput 29:379–385
9. Li X, Li Y, Wang Y (2017) Minimising makespan on a batch processing machine using heuristics improved by an enumeration scheme. Int J Prod Res 55(1):176–186
10. Jia ZH, Li K, Leung JYT (2015) Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities. Int J Prod Econ 169:1–10
11. Ozturk O, Begen MA, Zaric GS (2017) A branch and bound algorithm for scheduling unit size jobs on parallel batching machines to minimize makespan. Int J Prod Res 55(6):1815–1831
12. Jia Z, Li X, Leung JYT (2017) Minimizing makespan for arbitrary size jobs with release times on p-batch machines with arbitrary capacities. Futur Gener Comput Syst 67:22–34
13. Bellanger A, Oulamara A, Kovalyov MY (2010) Minimizing total completion time on a batching machine with job processing time compatibilities. Electron Notes Discrete Math 36:1295–1302
14. Yao S, Jiang Z, Li N (2012) A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals. Comput Oper Res 39(5):939–951
15. Li SS, Zhang YZ (2014) Serial batch scheduling on uniform parallel machines to minimize total completion time. Inf Process Lett 114(12):692–695
16. Xu R, Chen HP, Shao H, Wang SS (2010) Two kinds of ant colony algorithms to minimize the total completion time for batch scheduling problem. Comput Integr Manuf Syst 16(6):1255–1264
17. Jia ZH, Zhang H, Long WT, Leung JYT, Li K, Li W (2018) A meta-heuristic for minimizing total weighted flow time on parallel batch machines. Comput Ind Eng 125:298–308
18. Li Z, Chen H, Xu R, Li X (2015) Earliness–tardiness minimization on scheduling a batch processing machine with non-identical job sizes. Comput Ind Eng 87:590–599
19. Yu JM, Huang R, Lee DH (2017) Iterative algorithms for batching and scheduling to minimise the total job tardiness in two-stage hybrid flow shops. Int J Prod Res 55(11):3266–3282
20. Gokhale R, Mathirajan M (2014) Minimizing total weighted tardiness on heterogeneous batch processors with incompatible job families. Int J Adv Manuf Technol 7(9-12):1563–1578
21. Chou FD (2013) Minimising the total weighted tardiness for non-identical parallel batch processing machines with job release times and non-identical job sizes. Eur J Ind Eng 7(5):529–557
22. Anokić A, Stanimirović Z, Stakić D, Davidović T (2019) Metaheuristic approaches to a vehicle scheduling problem in sugar beet transportation. Oper Res :1–33
23. Chavhan S, Gupta D, Chandana B, Chidambaram RK, Khanna A, Rodrigues JJ (2020) A novel emergent intelligence technique for public transport vehicle allocation problem in a dynamic transportation system. IEEE Trans Intell Transp Syst

24. Islam MR, Mahmud MR, Pritom RM (2019) Transportation scheduling optimization by a collaborative strategy in supply chain management with tpl using chemical reaction optimization. Neural Comput Applic :1–26
25. Peng J (2019) Optimizing the transportation route of fresh food in cold chain logistics by improved genetic algorithms. Int J Metrol Qual Eng 10:14
26. Pitakaso R, Sethanan K, Jamrus T (2020) Hybrid pso and alns algorithm for a software and mobile application for transportation in the ice manufacturing industry. Comput Ind Eng :106461
27. Mensendiek A, Gupta JN, Herrmann J (2015) Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. Eur J Oper Res 243(2):514–522
28. Jia ZH, Zhuo XX, Leung JYT, Li K (2019) Integrated production and transportation on parallel batch machines to minimize total weighted delivery time. Comput Oper Res 102:39–51
29. Liu L, Liu S (2020) Integrated production and distribution problem of perishable products with a minimum total order weighted delivery time. Mathematics 8(2):146
30. Liu P, Lu X (2016) Integrated production and job delivery scheduling with an availability constraint. Int J Prod Econ 176:1–6
31. Jia ZH, Huo SY, Li K, Chen HP (2019) Integrated scheduling on parallel batch processing machines with non-identical capacities. Eng Optim 1–16
32. Li F, Chen ZL, Tang L (2017) Integrated production, inventory and delivery problems: Complexity and algorithms. INFORMS J Comput 29(2):232–250
33. Chen ZL (2010) Integrated production and outbound distribution scheduling: review and extensions. Oper Res 58(1):130–148
34. Wang DY, Grunder O, Moudni AE (2015) Integrated scheduling of production and distribution operations: a review. Int J Ind Syst Eng 19(1):94–122
35. Hulett M, Damodaran P, Amouie M (2017) Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. Comput Ind Eng 113:425–436
36. Zhou S, Xie J, Du N, Pang Y (2018) A random-keys genetic algorithm for scheduling unrelated parallel batch processing machines with different capacities and arbitrary job sizes. Appl Math Comput 334:254–268
37. Li K, Xiao W, Yang S (2019) Minimizing total tardiness on two uniform parallel machines considering a cost constraint. Expert Systems With Applications 123:143–153
38. Jia ZH, Gao LY, Zhang XY (2020) A new history-guided multi-objective evolutionary algorithm based on decomposition for batching scheduling. Expert Syst Appl 141:112920
39. Jia ZH, Li YJ, Li K, Chen HP (2020) Weak-restriction bi-objective optimization algorithm for scheduling with rejection on non-identical batch processing machines. Appl Soft Comput 86:105914

Zhao-hong Jia is a Professor in Anhui University. She received her M.S. degree in Computer Science in 2003 from Anhui University and her Ph.D. degree in Management Science and Technology in 2008 from University of Science and Technology of China. Her research interests include intelligent computation and its applications, multi-objective optimization and operations research.

Yu-fei Cui is currently working toward his M.S. degree in Computer Science and Technology in Anhui University. His research interest includes computational intelligence.

Kai Li is a Professor in Hefei University of Technology. He received his M.S. and Ph.D. degrees from Hefei University of Technology, in 2005 and 2009, respectively. His research interests include operations research, manufacturing system optimization and service system optimization.