Journal of Scheduling (2020) 23:649–664
https://doi.org/10.1007/s10951-019-00619-5

Less is more: variable neighborhood search for integrated production and assembly in smart manufacturing

Shaojun Lu[1,3] · Jun Pei[1,2,3] · Xinbao Liu[1,3] · Xiaofei Qian[1,3] · Nenad Mladenovic[4,5] · Panos M. Pardalos[2]

Published online: 7 September 2019

Abstract

This paper investigates an integrated production and assembly scheduling problem with the practical manufacturing features of serial batching and the effects of deteriorating and learning. The problem is divided into two stages. During the production stage, there are several semi-product manufacturers who first produce ordered product components in batches, and then these processed components are sent to an assembly manufacturer. During the assembly stage, the assembly manufacturer will further process them on multiple assembly machines, where the product components are assembled into final products. Through mathematical induction, we characterize the structures of the optimal decision rules for the scheduling problem during the production stage, and a scheme is developed to solve this scheduling problem optimally based on the structural properties. Some useful lemmas are proposed for the scheduling problem during the assembly stage, and a heuristic algorithm is developed to eliminate the inappropriate schedules and enhance the solution quality. We then prove that the investigated problem is NP-hard. Motivated by this complexity result, we present a less-is-more-approach-based variable neighborhood search heuristic to obtain the approximately optimal solution for the problem. The computational experiments indicate that our designed LIMA-VNS (less is more approach–variable neighborhood search) has an advantage over other metaheuristics in terms of converge speed, solution quality, and robustness, especially for large-scale problems.

Keywords Variable neighborhood search · Less is more · Deteriorating effect · Learning effect · Serial-batching scheduling · Assembly

1 Introduction

In recent years, smart manufacturing has been experiencing an explosive growth and has now become one of the dominant factors for enterprises to compete in the fiercer global competition (Yang et al. 2018). The production of many products, e.g., mobile phones, aircraft, and automobiles, needs to be first processed by several semi-product manufacturers, and then it is completed by assembly manufacturers, as is shown in Fig. 1. It is of great significance to make smart decisions for semi-product manufacturers and assembly manufacturers in the manufacturing system to reduce production duration and increase profit. To cope with global competition, some researchers have investigated optimization problems in assembly lines (Framinan et al. 2019), where they have proposed various solution methods and have summarized development trends of assembly production scheduling. In addition, Renna and Perrone (2015) investigated manufacturing systems with multiple manufacturers and stated that coordinating manufacturing is very important in today's world. Jia and Mason (2009) considered the problem of scheduling orders in a multi-machine manufacturing system. Most previous research has focused on general processing without considering practical specific

✉ Jun Pei
feiyijun.ufl@gmail.com
http://www.drpeijun.com

✉ Xinbao Liu
lxb@hfut.edu.cn

1 School of Management, Hefei University of Technology, Hefei, China

2 Department of Industrial and Systems Engineering, Center for Applied Optimization, University of Florida, Gainesville, USA

3 Key Laboratory of Process Optimization and Intelligent Decision-Making of Ministry of Education, Hefei, China

4 Department of Industrial Engineering, Khalifa University, Abu Dhabi, UAE

5 Ural Federal University, Yekaterinburg, Russia

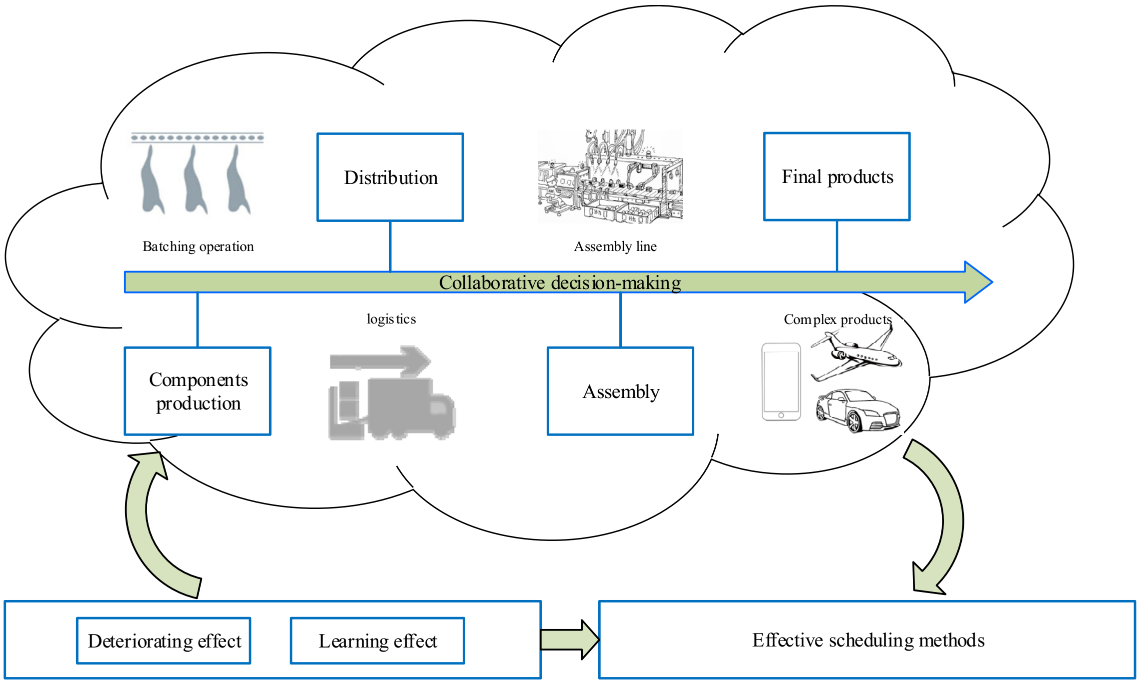

Fig. 1 The integrated scheduling problem in complex product manufacturing

production features. However, there are a number of practical issues in a real manufacturer system, such as serial-batching production, the learning effect, and the deteriorating effect, which make it more difficult for decision-makers in the manufacturing system to produce efficient schedules. Thus, in this paper, we address an integrated manufacturing optimization problem which covers various practical issues.

Classical scheduling problems assume that the processing times of jobs are constant values. However, the actual processing time of a job may increase or decrease with changing machine efficiency and human proficiency. These phenomena are called the deteriorating effect and the learning effect (Wang 2007). Gawiejnowicz (2008) systematically summarized various types of deterioration effects, and he also analyzed how these effects affect the optimal scheduling scheme in general scheduling problems. Azzouz et al. (2018) presented a concise overview of learning effects, provided a classification scheme for the different scheduling problems under learning effects, and discussed the relationship between some well-known leaning models. To solve real-life production scheduling problems, many other researchers have also taken the deteriorating effect and the learning effect into consideration. Wang and Cheng (2007) studied a single-machine scheduling problem and constructed time- and position-based job-processing time functions. Considering that there should be an upper limit to proficiency, Wang et al. (2017) investigated a single-machine scheduling problem with a truncated job-dependent learning effect. Yin et al. (2017) considered deteriorating jobs in a parallel machine scheduling problem with potential machine disruptions. Fu et al. (2018) studied scheduling problems with deteriorating and learning effects in an industry 4.0-based manufacturing system.

In a batching scheduling problem, jobs are grouped into batches and processed on the batching machine. This type of production mode can improve the efficiency of the machine, while a setup operation is required before the processing of each batch to ensure the security and quality of the production. Batching scheduling problems commonly exist in semi-product manufacturing (Yin et al. 2016; Ahmadizar and Farhadi 2015) and have drawn much attention in recent years. Mor and Mosheiov (2011) investigated a two-agent single-machine scheduling problem in the context of batch processing. For a batching scheduling problem with transportation, Yin et al. (2013) developed an efficient dynamic programming algorithm under an assumption on the relationships between the cost parameters. Differently, Yin et al. (2015) considered a single-machine batching scheduling problem with rejection costs and developed algorithms with significant improvements. Then, to solve the problem in more complex manufacturing systems, Yin et al. (2018) studied integrated production, inventory, and batch delivery scheduling problems with due date assignments and two competing agents, based on which effective heuristics were proposed. Kress et al. (2018) focused on the total setup cost minimization problem and proposed an approximately optimal algorithm to solve it in polynomial time. Alam and Raza (2018) investigated a batching scheduling

Table 1 Comparisons of previous related literature and the current study

Publication	Assembly	Batching	Objective	Setup time	Algorithms	Processing time	
						LE	DE
Potts et al. (1995)	√	×	Makespan	×	Heuristic algorithms	×	×
Hariri and Potts (1997)	√	×	Makespan	×	Branch and Bound	×	×
Allahverdi and Al-Anzi (2006)	√	×	Makespan	√	Heuristic algorithms	×	×
Shokrollahpour et al. (2011)	√	×	Weighted sum of makespan	×	Imperialist competitive algorithm	×	×
Wu et al. (2018)	√	×	Makespan	×	Metaheuristic	√	×
Luo et al. (2018)	√	×	Makespan	√	Branch and Bound	×	×
Pei et al. (2017)	×	√	Maximum earliness	√	Heuristic algorithms	√	√
Fan et al. (2018)	×	√	Makespan	√	Heuristic algorithms and metaheuristics	√	√
Lu et al. (2018)	×	√	Makespan	×	Heuristic algorithms and metaheuristics	×	√
Current study	√	√	Makespan	√	Heuristic algorithms and metaheuristics	√	√

DE deteriorating effect;*LE* learning effect; √ (×) denotes that the corresponding issue is (not) considered

problem in a parallel and distributed environment. Shahvari and Logendran (2018) solved a hybrid flow shop batching scheduling problem via two-stage-based hybrid algorithms.

In the assembly scheduling literature, some papers have analyzed makespan minimization problems within a certain factory, typically where the component production and assembly can be finished in a single production flow line (Liao et al. 2015). In these settings, the operator only needs to consider the optimal schedule of the product components on a single machine, and the assembly starts whenever a set of parts is finished. However, in a global manufacturing system, the product components are manufactured by several enterprises located in different places. After being processed by the semi-product manufacturer, the product components are delivered and assembled into the final product by an assembly manufacturer. Roh et al. (2014) investigated the response supply chain in global complexity and stated that improving production capability is a key to the success of firms in the global supply chain. Tao et al. (2017) analyzed the evolution of such a manufacturing system in the environment of information technology.

There is limited existing literature on integrated scheduling models considering the production, assembly, serial-batching processing, the deteriorating effect, and the learning effect simultaneously. Our previous research papers involved such issues as the batching and deteriorating effects, and we focused on the single-stage scheduling problems with multiple machines or a single machine. The comparison of this with existing publications is presented in Table 1. In Pei et al. (2017), we studied the batching scheduling problem with the deteriorating effect as well as the learning effect and proposed several effective heuristics, but the assembly process was not considered. Hence, those proposed methods cannot solve the problem in this work. Lu et al. (2018) investigateda coordinating production and maintenance scheduling problem under the deteriorating effect. The assembly process and the leaning effect are ignored in that model. This paper contributes by considering various significant issues in an integrated scheduling problem and proposes several useful structural properties, based on which effective heuristics and a metaheuristic are developed to make decisions for different participants in the manufacturing system. To the best of our knowledge, the proposed scheduling problem in this paper has not been studied in any existing literature. However, the problem exists in real-life manufacturing and solving the problem is significant for the realization of smart manufacturing, which is the motivation of this paper.

The main contributions of this work are stated as follows.

1. We formulated an integrated scheduling model which covers production and assembly. In addition, the features of serial-batching processing, the deteriorating effect, and the learning effect are taken into consideration in our model.
2. For the serial-batching scheduling problem in the production stage, several structural properties are proposed, based on which an optimal algorithm is developed.
3. For the assembly scheduling problem, we give some useful lemmas and design a heuristic which can improve the solution quality.
4. Since the integrated problem is proved to be NP-hard, we develop a LIMA-VNS (less is more approach–variable

Table 2 Notations

Notations	Definitions
N	The number of products
g	The number of semi-product manufacturers or product components
G	The number of machines in the assembly manufacturer
P_I	The product I, $I = 1, 2, \ldots, N$
R_I	The available time for P_I is the assembly stage, $I = 1, 2, \ldots, N$
M_m	The mth semi-product manufacturer, $m = 1, 2, \ldots, g$
AM_d	The dth assembly machine, $d = 1, 2, \ldots, G$
N_d	The number of products in A_d, $d = 1, 2, \ldots, G$
$p_{i,m}$	The mth product component for the P_I, $I = 1, 2, \ldots, N$, $m = 1, 2, \ldots, g$
$t_{i,m}$	The normal processing time of the $p_{i,m}$, $I = 1, 2, \ldots, N$, $m = 1, 2, \ldots, g$
$t_{i,m}^A$	The actual processing time of the $p_{i,m}$, $I = 1, 2, \ldots, N$, $m = 1, 2, \ldots, g$
c	The machine capacity
n_m	The number of batches in M_m, $m = 1, 2, \ldots, g$
$B_{e,m}$	The eth batch in M_m, $e = 1, 2, \ldots, n_m$, $m = 1, 2, \ldots, g$
$s_{e,m}$	The setup time for $B_{e,m}$, $e = 1, 2, \ldots, n_m$, $m = 1, 2, \ldots, g$
$t_{[j],m}$	The normal processing time of the jth component in M_m, $j = 1, 2, \ldots, N$, $m = 1, 2, \ldots, g$
$t_{[j],m}^A$	The actual processing time of the jth component in M_m, $j = 1, 2, \ldots, N$, $m = 1, 2, \ldots, g$
θ, δ	The deteriorating rates for setup operations
D_m	The delivery time between the mth semi-product manufacturer and the assembly manufacturer, $m = 1, 2, \ldots, g$
a_I	The normal processing time of the P_I, $I = 1, 2, \ldots, N$
$a_{[l],d}$	The normal processing time of the lth product in AM_d, $l = 1, 2, \ldots, N_d$, $d = 1, 2, \ldots, G$
$a_{[l],d}^A$	The actual processing time of the lth product in AM_d, $l = 1, 2, \ldots, N_d$, $d = 1, 2, \ldots, G$
$S_{[l],d}$	The setup time of the lth product in AM_d, $l = 1, 2, \ldots, N_d$, $d = 1, 2, \ldots, G$
$C_{\max}^m$	The maximum completion time for M_m, $m = 1, 2, \ldots, g$
$A_{\max}^d$	The maximum completion time for AM_d, $d = 1, 2, \ldots, G$
$A_{\max}$	The makespan for the assembly machines

neighborhood search) to solve the problem. The performance of the LIMA-VNS is validated via computational experiments.

The remainder of this paper is organized as follows: Sect. 2 gives the formulation of the considered integrated scheduling problem. In Sects. 3 and 4, we discuss the subproblems of the integrated model and develop heuristics to solve them, respectively. In Sect. 5, we design a metaheuristic to solve the integrated scheduling problem. Sect. 6 presents the results of computational experiments and analyzes the performance of the involved metaheuristics. Finally, Sect. 7 concludes and gives some future research directions.

2 Notations and problems statement

The used notations throughout this paper and their definitions are presented in Table 2.

The integrated scheduling problem is formulated as a two-stage model. During the component production stage, the product components are processed and delivered in batches. Each semi-product manufacturer ensures that the completion time is minimized by solving a serial-batching scheduling problem, which is denoted as Q1. Then, during the assembly stage, the assembly manufacturer aims at minimizing the makespan under the constraints of the component finishing time and transportation, which is denoted as Q2. In the manufacturing system, the semi-product manufacturers first make decisions and then the assembly manufacturers will try to minimize the makespan based on their decisions. During the above scheduling period, the decisions concern: (1) how to group product components into batches for each semi-product manufacturer, (2) how to sequence the batches, (3) how to assign products to the assembly machines, and (4) how to sequence the products on each assembly machine. The decisions (1) and (2) are for the semi-product manufacturer, while the decisions (3) and (4)

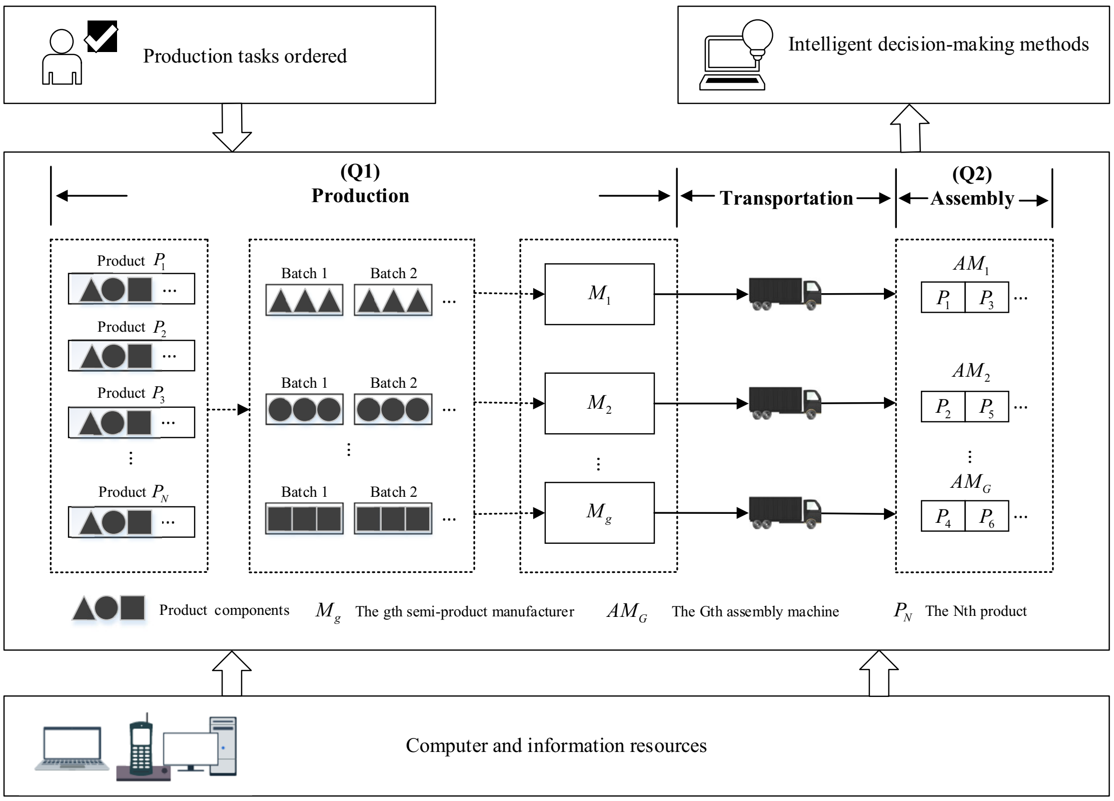

Fig. 2 The integrated scheduling problem with production, transportation, and assembly

are for the assembly manufacturers. The joint decisions can improve the efficiency of the whole manufacturing system. The two-stage formulation is consistent with the service-oriented global manufacturing system, where different manufacturing service providers finish the finial products together and want to gain more profit through improving the efficiency.

The structure of the integrated scheduling problem is depicted in Fig. 2. We assume that there are N products $\{P_1, P_2, \ldots, P_I, \ldots, P_N\}$ to be processed by g semi-product manufacturers $\{M_1, M_2, \ldots, M_m, \ldots, M_g\}$ and G machines of an assembly manufacturer. There are g semi-product manufacturers processing different product components, and each of them is necessary for the final products, i.e., the mth product component must be processed by the mth semi-product manufacturer. Hence, each product consists of g components $\{p_{I,1}, p_{i,2}, \ldots, p_{i,m}, \ldots, p_{i,g}\}$. During the production stage, the components are finished by the semi-product manufacturers. Each semi-product manufacturer has a single serial-batching machine. The machine capacity is denoted as c, which means that the number of jobs in a batch cannot exceed c. The setup time is required to start a new batch. Let $B_{e.m}$ denote the eth batch in the m th semi-product manufacturer. The setup time is formulated as in Cheng et al. (2011):

$$s_{e,m} = \theta T, \ e = 1, 2, \ldots, n_m, \ m = 1, 2, \ldots, g. \quad (1)$$

where θ is the deteriorating rate, T denotes the starting time, and n_m is the number of batches on the machine from the mth semi-product manufacturer. In addition, the sum-of-processing-times-based deterioration (Liu et al. 2013) is extended to serial-batching scheduling criteria. Denoting the normal processing time of the jth component on M_m as $t_{[j],m}$, the actual processing time of the jth component on M_m is formulated as follows:

$$t^A_{[j],m} = \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right) t_{[j],m},$$
$$j = 1, 2, \ldots, N, \ m = 1, 2, \ldots, g, \ \sum_{x=1}^{0} t^A_{[x],m} = 0. \quad (2)$$

After the production stage, the finished components are delivered to the assembly manufacturer and assembled into final products. The capacity limitation on the delivery is equal to that of the production machine. We assume that there are enough vehicles to deliver each batch as soon as it is finished in the production stage. The semi-product manufacturers are in different places, which results in different delivery times. We denote the delivery time between the mth semi-product manufacturer and the assembly manufacturer as D_m, $m = 1, 2, \ldots, g$. Then, for the assembly stage, each machine in the assembly manufacturer can only assemble one product at one time. Considering that human activities are of great significance in the assembly stage, we use the learning model proposed in Cheng et al. (2009) to formulate the actual assembly time of each product such that:

$$a^A_{[l],d} = \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d}\right)^{\alpha} a_{[l],d},$$

$$l = 1, 2, \ldots, N_d,\ d = 1, 2, \ldots, G,\ \sum_{x=1}^{0} \ln a_{[x],d} = 0. \tag{3}$$

where $a_{[l],d}$ denotes the normal processing time of the lth product on AM_d and $\alpha < 0$ is the learning index. In addition, the setup time for each product is formulated as:

$$S_{[l],d} = \delta T, \quad l = 1, 2, \ldots, N_d,\ d = 1, 2, \ldots, G \tag{4}$$

where δ is the deteriorating rate for the assembly process, T denotes the starting time, and N_d is the number of products in the dth assembly machine.

3 Structural properties and an optimal algorithm for Q1

In this section, we focus on the serial-batching scheduling problem in the production stage. Some structural properties are proposed, and an optimal rule is designed to schedule product components in each semi-product manufacturer. Moreover, the completion time of the product components is derived, which is very important for the decision making on the assembly stage.

Lemma 1 *In an optimal schedule for a certain semi-product manufacturer, all product components are sorted in the non-decreasing order of the normal processing times.*

Proof Suppose that there are two adjacent product components $p_{I_1,m}$ and $p_{I_2,m}$ in the mth semi-product manufacturer, where $I_1 = 1, 2, \ldots, N$ and $I_2 = 1, 2, \ldots, N$, we consider two cases in this proof. One is that the two product components $p_{I_1,m}$ and $p_{I_2,m}$ are from the same batch, and the other is that they are from different batches. For the first case, we assume that there exist two schedules $\pi = \left\{W_1, \underbrace{\ldots, p_{I_1,m}, p_{I_2,m}, \ldots}_{B_{e,m}}, W_2\right\}$ and $\pi^* = \left\{W_1, \underbrace{\ldots, p_{I_2,m}, p_{I_1,m}, \ldots}_{B_{e,m}}, W_2\right\}$, where W_1 and W_2 are the partial sequences. Let $S\big(p_{I_1,m}(\pi)\big)$ and j denote the starting time and the position of the product component $p_{I_1,m}$ in π. Then, we can calculate the completion time of the product component $p_{I_2,m}$ in π as

$$\begin{aligned}
&C\left(p_{I_2,m}(\pi)\right)\\
&= S\left(p_{I_1,m}(\pi)\right) + \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right) t_{I_1,m}\\
&\quad + \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m} + \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right) t_{I_1,m}}{\sum_{x=1}^{N} t_{[x],m}}\right) t_{I_2,m}\\
&= S\left(p_{I_1,m}(\pi)\right)\\
&\quad + \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right) \cdot \left(t_{I_1,m} + t_{I_2,m}\right)\\
&\quad + \frac{\left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right)}{\sum_{x=1}^{N} t_{[x],m}} t_{I_1,m} \cdot t_{I_2,m}
\end{aligned}$$

Similarly, the completion time of the product component $p_{I_1,m}$ in π^* is

$$\begin{aligned}
C\left(p_{I_1,m}(\pi^*)\right) &= S\left(p_{I_2,m}(\pi^*)\right) + \left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right)\\
&\quad \cdot \left(t_{I_1,m} + t_{I_2,m}\right)\\
&\quad + \frac{\left(1 + \frac{\sum_{x=1}^{j-1} t^A_{[x],m}}{\sum_{x=1}^{N} t_{[x],m}}\right)}{\sum_{x=1}^{N} t_{[x],m}} t_{I_1,m} \cdot t_{I_2,m}.
\end{aligned}$$

Since $S\big(p_{I_2,m}(\pi^*)\big) = S\big(p_{I_1,m}(\pi)\big)$, we obtain $C\big(p_{I_2,m}(\pi)\big) = C\big(p_{I_1,m}(\pi^*)\big)$.

For the second case, we take the similar expression as in the first case but define the two different schedules as $\pi = \left\{W_1, \underbrace{\ldots, p_{I_1,m}}_{B_{e,m}}, \underbrace{p_{I_2,m}, \ldots}_{B_{e+1,m}}, W_2\right\}$ and $\pi^* = \left\{W_1, \underbrace{\ldots, p_{I_2,m}}_{B_{e,m}}, \underbrace{p_{I_1,m}, \ldots}_{B_{e+1,m}}, W_2\right\}$.

Then, we derive

$$
\begin{aligned}
&C\left(p_{I_2,m}(\pi)\right)\\
&\quad=(1+\theta)\,S\left(p_{I_1,m}(\pi)\right)+(1+\theta)\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_1,m}\\
&\qquad+\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}+\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_1,m}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_2,m}\\
&\quad=(1+\theta)\,S\left(p_{I_1,m}(\pi)\right)\\
&\qquad+\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)\\
&\qquad\cdot\left(t_{I_1,m}+t_{I_2,m}\right)+\frac{\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)}{\sum_{x=1}^{N}t_{[x],m}}\\
&\quad t_{I_1,m}\cdot t_{I_2,m}+\theta\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_1,m}.
\end{aligned}
$$

and

$$
\begin{aligned}
&C\left(p_{I_1,m}(\pi^*)\right)\\
&\quad=(1+\theta)S\left(p_{I_2,m}(\pi^*)\right)+\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)\cdot\left(t_{I_1,m}+t_{I_2,m}\right)\\
&\qquad+\frac{\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)}{\sum_{x=1}^{N}t_{[x],m}}t_{I_1,m}\cdot t_{I_2,m}\\
&\qquad+\theta\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_2,m}.
\end{aligned}
$$

Hence, we have $C\left(p_{I_2,m}(\pi)\right)-C\left(p_{I_1,m}(\pi^*)\right)=\theta\left(1+\frac{\sum_{x=1}^{j-1}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)\left(t_{I_1,m}-t_{I_2,m}\right)$.

Combining the two cases, it can be derived $C\left(p_{I_2,m}(\pi)\right)\leq C\left(p_{I_1,m}(\pi^*)\right)$ when $t_{I_1,m}\leq t_{I_2,m}$. Thus, this lemma is proved. □

Lemma 2 *In an optimal schedule for a certain semi-product manufacturer, all batches contain c product components except for the first batch in each semi-product manufacturer.*

Proof Let $B_{e,g}$ and $B_{e+1,g}$ be two adjacent batches, where $e=1,\cdots,n_m-1$. We assume that the number of product components in $B_{e+1,g}$ is less than c, the last two product components in $B_{e,g}$ are $p_{I_1,m}$ and $p_{I_2,m}$, and the first product component in $B_{e+1,g}$ is $p_{I_3,m}$, i.e., $\pi=\left\{W_1,\underbrace{\ldots,p_{I_1,m},p_{I_2,m}}_{B_{e,m}},\underbrace{p_{I_3,m}\ldots}_{B_{e+1,m}},W_2\right\}$. Then, we obtain a new schedule π^* by moving the product component $p_{I_2,m}$ from $B_{e,m}$ to $B_{e+1,m}$ such that $\pi^*=\left\{W_1,\underbrace{\ldots,p_{I_1,m}}_{B_{e,m}},\underbrace{p_{I_2,m},p_{I_3,m}\ldots}_{B_{e+1,m}},W_2\right\}$. Suppose the $p_{I_1,m}$ is in the jth position, the completion times of $p_{I_2,m}$ in π and π^* are calculated as $S\left(p_{I_3,m}(\pi)\right)=(1+\theta)C\left(p_{I_1,m}(\pi)\right)+(1+\theta)\left(1+\frac{\sum_{x=1}^{j}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_2,m}$ and $S\left(p_{I_3,m}(\pi^*)\right)=(1+\theta)C\left(p_{I_1,m}(\pi)\right)+\left(1+\frac{\sum_{x=1}^{j}t_{[x],m}^{A}}{\sum_{x=1}^{N}t_{[x],m}}\right)t_{I_2,m}$. Then, since the other sequences are the same for the two schedules, it is easy to obtain $C_{max}^{m}(\pi)>C_{max}^{m}(\pi^*)$. Thus, it can be concluded that this lemma holds. □

Then, based on Lemmas 1 and 2, we develop the following schedule algorithm on product components sequencing, batching, and batch sequencing.

```
Algorithm 1
Step 1. For each semi-product manufacturer, sort the product
    components in the non-decreasing order of the normal processing
    times.
Step 2. If $N - c\cdot\lceil \frac{N}{c}\rceil < 0$, then, group the first $N - c\cdot(\lceil \frac{N}{c}\rceil - 1)$
    product components into a batch and remove them from the list.
Step 3. Group the first $c$ product components into a new batch and
    remove them from the list.
Step 4. Repeat Step 3 until the job list is empty.
Step 5. Process the batches in the generated order and process the
    product components in the non-decreasing order of the normal
    processing times.
```

Since the computational complexity of step 1 and the steps 2–5 is $O(N\log N)$ and $O(N)$, respectively, the total complexity of Algorithm 1 is $O(N\log N)$. Based on Lemmas 1 and 2, we know that Algorithm 1 can obtain the optimal schedule for each semi-product manufacturer.

Lemma 3 *For the problem Q1, the completion time of each semi-product manufacturer is:*

$$
\begin{aligned}
C_{\max}^{m}&=\sum_{x=1}^{N}t_{[x],m}\prod_{y=1}^{N}\left(1+\frac{1}{\sum_{x=1}^{N}t_{[x],m}}t_{[y],m}\right)-\sum_{x=1}^{N}t_{[x],m}\\
&-\sum_{x=1}^{f\left(\lceil\frac{N}{c}\rceil-1\right)}t_{[x],m}\prod_{y=1}^{N}\left(1+\frac{1}{\sum_{x=1}^{N}t_{[x],m}}t_{[y],m}\right)+\sum_{x=1}^{f\left(\lceil\frac{N}{c}\rceil-1\right)}t_{[x],m}\\
&+\sum_{z=1}^{\lceil N/c\rceil-1}(1+\theta)^{\lceil N/c\rceil-z}\Bigg[\sum_{x=1}^{f(z)}t_{[x],m}\prod_{y=1}^{f(z)}\left(1+\frac{1}{\sum_{x=1}^{N}t_{[x],m}}t_{[y],m}\right)-\sum_{x=1}^{f(z)}t_{[x],m}\\
&-\sum_{x=1}^{f(z-1)}t_{[x],m}\prod_{y=1}^{f(z-1)}\left(1+\frac{1}{\sum_{x=1}^{N}t_{[x],m}}t_{[y],m}\right)+\sum_{x=1}^{f(z-1)}t_{[x],m}\Bigg]
\end{aligned}
\tag{5}
$$

where

$$
f(z)=\begin{cases}N-c\cdot\left\lceil\frac{N}{c}\right\rceil+z\cdot c,\ N-c\cdot\left\lceil\frac{N}{c}\right\rceil<0, & z>0\\ zc, & N-c\cdot\left\lceil\frac{N}{c}\right\rceil=0, z>0\\ 0, & z=0.\end{cases}
$$

Proof Given an optimal schedule in each semi-product manufacturer such as $\pi = \left\{B_{1,m}, B_{2,m}, \ldots, B_{\left\lceil \frac{N}{c} \right\rceil,m}\right\}$, we can derive the number of jobs in each batch based on Lemmas 1 and 2, e.g., the number of jobs in $\left|B_{e,m}\right|$ is $\left|B_{e,m}\right| = \begin{cases} N - c \cdot \left\lceil \frac{N}{c} \right\rceil + c, N - c \cdot \left\lceil \frac{N}{c} \right\rceil < 0, \; e = 1 \\ c, \quad \text{other} \end{cases}$. Hence, it can be obtained that the total number of jobs in the first $e > 0$ batches is $\sum_{x=1}^{e} \left|B_{e,m}\right| = f(e) = \begin{cases} N - c \cdot \left\lceil \frac{N}{c} \right\rceil + e \cdot c, \; N - c \cdot \left\lceil \frac{N}{c} \right\rceil < 0 \\ ec, \quad N - c \cdot \left\lceil \frac{N}{c} \right\rceil = 0 \end{cases}$. Then, we can prove this lemma using mathematical induction. First, for $e = 1$, we have

$$C\left(B_{1,m}\right) = \sum_{x=1}^{f(1)} t_{[x],m} \prod_{y=1}^{f(1)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(1)} t_{[x],m}.$$

Assume that we have:

$$\begin{aligned} C\left(B_{e,m}\right) = & \sum_{x=1}^{f(e)} t_{[x],m} \prod_{y=1}^{f(e)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(e)} t_{[x],m} \\ & - \sum_{x=1}^{f(e-1)} t_{[x],m} \prod_{y=1}^{N} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) + \sum_{x=1}^{f(e-1)} t_{[x],m} \\ & + \sum_{z=1}^{e-1} (1+\theta)^{\lceil N/c \rceil - z} \left[\sum_{x=1}^{f(z)} t_{[x],m} \prod_{y=1}^{f(z)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(z)} t_{[x],m} \right. \\ & \left. - \sum_{x=1}^{f(z-1)} t_{[x],m} \prod_{y=1}^{f(z-1)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) + \sum_{x=1}^{f(z-1)} t_{[x],m} \right] \end{aligned} \tag{6}$$

for the batch $B_{e,m}$. Then, for the batch $B_{e+1,m}$, we derive its completion time as

$$\begin{aligned} & C\left(B_{e+1,m}\right) \\ & = \theta C\left(B_{e,m}\right) + C\left(B_{e,m}\right) \\ & \quad + \sum_{x=1}^{f(e+1)} t_{[x],m} \prod_{y=1}^{f(e+1)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(e+1)} t_{[x],m} \\ & \quad - \left[\sum_{x=1}^{f(e)} t_{[x],m} \prod_{y=1}^{f(e)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(e)} t_{[x],m}\right] \\ & = \sum_{x=1}^{f(e+1)} t_{[x],m} \prod_{y=1}^{f(e+1)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(e+1)} t_{[x],m} \\ & \quad - \sum_{x=1}^{f(e)} t_{[x],m} \prod_{y=1}^{N} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) + \sum_{x=1}^{f(e)} t_{[x],m} \\ & \quad + \sum_{z=1}^{e} (1+\theta)^{\left\lceil \frac{N}{c} \right\rceil - z} \left[\sum_{x=1}^{f(z)} t_{[x],m} \prod_{y=1}^{f(z)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) - \sum_{x=1}^{f(z)} t_{[x],m} \right. \\ & \quad \left. - \sum_{x=1}^{f(z-1)} t_{[x],m} \prod_{y=1}^{f(z-1)} \left(1 + \frac{1}{\sum_{x=1}^{N} t_{[x],m}} t_{[y],m}\right) + \sum_{x=1}^{f(z-1)} t_{[x],m} \right]. \end{aligned}$$

Thus, Eq. 6 also holds for the batch $B_{e+1,m}$. Combing the above results, we can conclude that this lemma holds. □

4 Structural properties and a local search strategy for Q2

Based on the results obtained in Sect. 3, the available time of each product in the assembly stage is derived. Further, under the case where products have been assigned to assembly machines, we proposed a local search strategy to improve the solution quality.

Lemma 4 *For the problem Q2, the available time of each product is*

$$R_I = \max_{1 \le m \le g} \left\{X_{i,e,m} C\left(B_{e,m}\right) + D_m\right\} \tag{7}$$

where $X_{i,e,m} = \begin{cases} 1, \; if p_{i,m} is assigned to B_{e,m} \\ 0, \; else \end{cases}$.

Lemma 5 *Let* $F(\lambda) = (1+\delta)(\lambda - 1) + (1 + c_0 ln\lambda + c_0 x)^{\alpha} - \lambda(1 + c_0 x)^{\alpha}$, *then* $F(\lambda) \ge 0$ *for* $\delta \ge 0$, $\lambda \ge 1$, $0 < c_0 < 1$,$x > 1$, *and* $\alpha < 0$.

Proof The lemmas can be easily proved by analyzing the first derivative and the second derivative of $F(\lambda)$ as the proof of Lemma 3 in Cheng et al. (2009), so we omit it here.

Lemma 6 *For the problem Q2, given the assignment of products, if there exist two adjacent products with* $R_{I_1} \le R_{I_2}$ *and* $a_{I_1} \le a_{I_2}$, *then the* P_{I_1} *should be processed preceded by* P_{I_2} *in an optimal schedule.*

Proof Suppose that $\varphi = \left\{E_1, P_{I_1}, P_{I_2}, E_2\right\}$ and $\varphi^* = \left\{E_1, P_{I_2}, P_{I_1}, E_2\right\}$ are two product schedules, where E_1 and E_2 are partial product sequences, $R_{I_1} \le R_{I_2}$, and $a_{I_1} \le a_{I_2}$. We assume that φ^* is an optimal schedule for the Q2 given the assignment of products. In addition, the position of P_{I_1} in φ is l. Denoting the completion time of the last product in E_1 as $A(E_1)$, the completion time of P_{I_1} and P_{I_2} in φ is

$$\begin{aligned} A\left(P_{I_1}(\varphi)\right) = & \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d}\right)^{\alpha} a_{I_1,d} \\ & + \delta \max\left\{A\left(E_1\right), R_{I_1}\right\} + \max\left\{A\left(E_1\right), R_{I_1}\right\} \\ = & \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d}\right)^{\alpha} a_{I_1,d} \\ & + (\delta + 1) \max\left\{A\left(E_1\right), R_{I_1}\right\} \end{aligned}$$

and:

$$\begin{aligned} A\left(P_{I_2}(\varphi)\right) = & (\delta+1)\max\left\{A\left(P_{I_1}(\varphi)\right), R_{I_2}\right\} + \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d} + \ln a_{I_1,d}\right)^{\alpha} a_{I_2,d} \\ = & (\delta+1)\max\left\{\left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d}\right)^{\alpha} a_{I_1,d} + (\delta+1)A(E_1), \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d}\right)^{\alpha} a_{I_1,d} \right. \\ & \left. + (\delta+1)R_{I_1}, R_{I_2}\right\} + \left(1 + \sum_{x=1}^{l-1} \ln a_{[x],d} + \ln a_{I_1,d}\right)^{\alpha} a_{I_2,d}. \end{aligned} \tag{8}$$

Then, for the schedule φ^*, we also calculate the completion time of P_{I_1} and P_{I_2} as follows:

$$\begin{aligned}
A\left(P_{I_2}\left(\varphi^*\right)\right) &= \left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_2,d} \\
&\quad + \delta\max\left\{A\left(E_1\right), R_{I_2}\right\} + \max\left\{A\left(E_1\right), R_{I_2}\right\} \\
&= \left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_2,d} \\
&\quad + (\delta+1)\max\left\{A\left(E_1\right), R_{I_2}\right\}
\end{aligned}$$

and:

$$\begin{aligned}
&A\left(P_{I_1}\left(\varphi^*\right)\right) = (\delta+1)\max\left\{A\left(P_{I_2}\left(\varphi^*\right)\right), R_{I_1}\right\} + \left(1+\sum_{x=1}^{l-1}\ln a_{[x],d} + \ln a_{I_2,d}\right)^{\alpha} a_{I_1,d} \\
&= (\delta+1)\max\{\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_2,d} + (\delta+1)A(E_1), \\
&\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_2,d} + (\delta+1)R_{I_2}, R_{I_1}\} \\
&+ \left(1+\sum_{x=1}^{l-1}\ln a_{[x],d} + \ln a_{I_2,d}\right)^{\alpha} a_{I_1,d}.
\end{aligned} \tag{9}$$

Set $\lambda = \frac{a_{I_2,d}}{a_{I_1,d}}$, $c_0 = \frac{1}{1+\sum_{x=1}^{l-1}\ln a_{[x],d}}$, and $x = \ln a_{I_1,d}$; then, based on Lemma 5, we obtain

$$\begin{aligned}
&\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d} + \ln a_{I_2,d}\right)^{\alpha} a_{I_1,d} + (\delta+1)\left[\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_2,d}\right] \\
&- \left(1+\sum_{x=1}^{l-1}\ln a_{[x],d} + \ln a_{I_1,d}\right)^{\alpha} a_{I_2,d} - (\delta+1)\left[\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} a_{I_1,d}\right] \\
&= a_{I_1,d}\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha} [(\delta+1)\left(\frac{a_{I_2,d}}{a_{I_1,d}}-1\right) \\
&+ \left(1+\frac{1}{1+\sum_{x=1}^{l-1}\ln a_{[x],d}}\ln\frac{a_{I_2,d}}{a_{I_1,d}} + \frac{1}{1+\sum_{x=1}^{l-1}\ln a_{[x],d}}\ln a_{I_1,d}\right)^{\alpha} \\
&- \frac{a_{I_2,d}}{a_{I_1,d}}\left(1+\frac{1}{1+\sum_{x=1}^{l-1}\ln a_{[x],d}}\ln a_{I_1,d}\right)^{\alpha}] \\
&= a_{I_1,d}\left(1+\sum_{x=1}^{l-1}\ln a_{[x],d}\right)^{\alpha}\left[(\delta+1)(\lambda-1) + (1+c_0\ln\lambda + c_0 x)^{\alpha} - \lambda(1+c_0 x)^{\alpha}\right] \geq 0.
\end{aligned}$$

Hence, we have:
The first term in Eq. (9)$\geq$ the first term in Eq. (8);
The second term in Eq. (9)$\geq$ the second term in Eq. (8);
The second term in Eq. (9)$\geq$ the third term in Eq. (8).
Thus, $A\left(P_{I_1}(\varphi^*)\right) \geq A\left(P_{I_2}(\varphi)\right)$, which conflicts with the assumption. The proof is completed.

Let $R_{[l],d}$ denote the available time of the lth product in AM_d. Based on Lemma 6, we design the following algorithm to improve the solution quality in the assembly stage.

Algorithm 2

```
Step 1. Set $l = 1$.
Step 2. If $R_{[l],d} \geq R_{[l+1],d}$ and $a_{[l],d} \geq a_{[l+1],d}$, then, go to step 3;
    else, go to step 5.
Step 3. If $R_{[l],d} = R_{[l+1],d}$ and $a_{[l],d} = a_{[l+1],d}$, then, go to step 5;
    else, swap the products in the $l$th and $(l+1)$th positions and go to
    step 4.
Step 4. If $l = 1$, then, go to step 5; else, set $l = l - 1$ and go to step
    2.
Step 5. Set $l = 1 + l$. If $l < N_d - 1$, then, go to Step 2; else, end the
    algorithm.
```

Based on Lemma 6, it is known that Algorithm 2 can improve the solution quality and eliminate some inappropriate solutions. The computation complexity will not exceed $O(l^2)$.

5 LIMA-VNS metaheuristic

If $R_I = 0$ and $\alpha = 0$, the studied problem is reduced to the NP-hard problem $P||A_{\max}$ (Coffman et al. 1978). Then, we can infer that the studied problem is also NP-hard. Hence, we develop a metaheuristic in this section to solve the problem in a reasonable time. The VNS is a very effective metaheuristic proposed by Mladenović and Hansen (1997). It aims at achieving an overall optimization in the solution space through systematically exploring different neighborhood structures. There are three significant principles for the VNS: the first principle is that a local optimal solution in one neighborhood structure may not be the optimal solution for another one; the second principle is that an overall optimal solution is the optimal solution for any kinds of neighborhood structures; and the last principle states that the local optimal solutions are close to each other, which is summarized in many experiments. VNS has shown a good performance in many complex scheduling problems (Bouffard and Ferland 2007, Wen et al. 2011, Zhang et al. 2018). To increase the diversity of solutions, other metaheuristics such as genetic algorithms and simulated annealing are integrated into the framework of VNS. However, Mladenović et al. (2016) suggested that integrating too many operations into VNS may not lead to better results. On the contrary, "less can yield more" exists in many problems. Based on this idea, we have designed a LIMA-VNS for an assembly scheduling problem with deteriorating and learning effects. We will first introduce the encoding scheme for the scheduling problem. Then, the procedure for generating the initial solution is described. Next, we present a simple and effective neighborhood for the scheduling problem, and the nested version of the neighborhood consists of the neighborhoods of the LIMA-VNS. This neighborhood composition is different from much existing

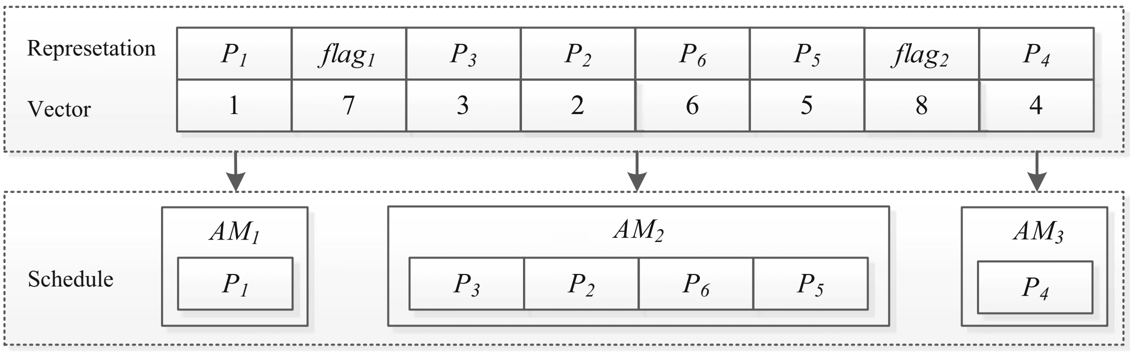

Fig. 3 The solution representation for a problem instance

literature which integrates several types of complex neighborhoods or heuristics into the VNS algorithm, such as in Roshanaei et al. (2009). Finally, we give the framework of the whole solution procedure. The neighborhood composition and framework of the algorithm both reflect the "less can yield more" mentioned above, obtaining better results with the most compact neighborhood and algorithm structure.

5.1 Solution representation

The metaheuristic LIMA-VNS is introduced to assign products to the assembly machines and to find an approximately optimal schedule for each machine. Thus, we use a $(N + G - 1)$-dimensional vector which is comprised of integers between 1 and $N + G - 1$ to express a solution in the assembly stage. The numbers between 1 and N denote products, while the numbers between $N + 1$ and $N + G - 1$ are flags which divide the products into G sets. The G sets will be assigned to the G assembly machines in turn. Hence, each number between 1 and $N+G-1$ must appear only once in each solution. A problem instance with 6 products and 3 machines is used to describe the solution representation in Fig. 3.

For the given solution vector $V = \{1, 7, 3, 2, 6, 5, 8, 4\}$, 7 and 8 are the two flags while 1–6 denote the 6 products. The product on the left of 7 is assigned to the assembly machine AM$_1$. The product P_4 is processed on the assembly machine AM$_3$. Products P_3, P_2, P_6, and P_5 are assigned to the assembly machine AM$_2$ and processed in the order $P_3 \rightarrow P_2 \rightarrow P_6 \rightarrow P_5$. The designed solution vector in this work specifies not only to which assembly machines the products are assigned ut also the processing order of the products on the assembly machines. Compared with continuous coding, this kind of discrete coding can better take advantage of VNS, which does not depend on complex update formulae, but relies on simple and diverse neighborhood structures. Compared with high-dimensional machine-assigned codes, this code has lower spatial complexity.

5.2 Initial solution generation

VNS is a single trajectory metaheuristic in the same way as simulated annealing and Tabu search (BoussaïD et al. 2013), which indicates that a single solution V is kept during the whole search process. Hence, the quality of the initial solution is significant for the performance of the metaheuristic. To obtain a good initial solution for LIMA-VNS, we first randomly generate a set which contains $x_{\max}$ solutions, and Algorithm 2 will then be used to improve each solution. Finally, the best solution among the improved solutions will be selected as the initial solution of the metaheuristic. The procedure for generating the initial solution is described in Algorithm 3 as follows:

Algorithm 3

```
Step 1. Set $x = 1$.
Step 2. Set $V_x = \{v_1, v_2, \ldots, v_{N+G-1}\}$, $X = \{1, 2, \ldots, N + G - 1\}$, and $y = 1$.
Step 3. Set $rand = random(1, N + G - 1)$, $v_y = X[rand]$.
Step 4. If $rand < N + G - 1$, then, set $X[rand] = X[rand + 1]$
    and go to step 6; else, go to step 5.
Step 5. If $rand > 1$, then, set $X[rand] = X[rand - 1]$ and go to
    step 6; else, output $V_x$ and end the algorithm.
Step 6. Set $y = y + 1$. If $y > N + G - 1$, then, go to step 7; else, go
    to step 3.
Step 7. Perform Algorithm-2 on $V_x$. Set $x = x + 1$. If $x > x_{max}$,
    then, output the best one among the $x_{max}$ solutions and end the
    algorithm; else, go to step 2.
```

In Algorithm 3, steps 3–6 can generate a feasible solution where each number between 1 and $N + G - 1$ appears once. Additionally, Algorithm 2 is able to improve the solution quality without generating infeasible solutions.

5.3 Neighborhood structures

The swap local search operator generates a new solution through an interchange of solution attributes (Mladenović and Hansen 1997), which is regarded as a very effective neighborhood structure in VNS. We modify this classical local search operator according to the nature of our investigated problem. Let x and y denote two integer numbers; for an original solution $V = \{v_1, v_2, \ldots, v_{N+G-1}\}$, the steps of the modified swap local search operator are shown as follows:

Algorithm 4

```
Step 1. Set $z = random(1, N+G-1)$ and $y = random(1, N+G-1)$.
Step 2. If $z = y$, then, go back to step 1; else, go to step 3.
Step 3. If $v_z > N \&\& v_y > N$, then, go back to step 1; else, go to step 4.
Step 4. Swap the two attributes such as $v_z$ and $v_y$ in $V$ to get the new solution $V'$. Output $V'$.
```

For the presented encoding scheme in Sect. 5.1, any interchange of two attributes which are both larger than N will not change the final schedule. Hence, compared to the general swap operator, the modified swap operator in this work ensures that the output solution V' is different from the original solution. Then, the metaheuristic can avoid unnecessary decoding and waste of computational time. Using NEI$_\omega$ to denote ω execution of the modified swap operator, we build a set of the neighborhood structures for the LIMA-VNS.

5.4 The framework of the solution procedure

Based on Algorithms 1–4, we have designed a LIMA-VNS and three modifications are made according to the nature of the investigated problem and our encoding scheme. To improve diversification searching, the first modification is to realize the output of different schedules for the local search operator. To improve the quality of the neighborhood solutions, the second modification is the designing of Algorithm 2 based on structural properties. The last modification is that we give an initial solution generation algorithm based on the encoding scheme in this paper, which can output a feasible solution of good quality. Integrating too many other heuristics into VNS usually increases the number of decodes. The decoding process takes a lot of time, especially for complex scheduling problems such as the integrated scheduling problem in this work. Hence, these complex algorithms often require a lot of computing time and violate the needs of the enterprises. In our proposed algorithm, the evaluation of solution quality is only performed once in each iteration and the designed neighborhood structure is concise and efficient, which means that we use fewer procedures to obtain better results. That is the so-called "less is more algorithm." The pseudocode of the whole solution procedure for the investigated problem is given as follows:

The pseudocode of the LIMA-VNS

```
Step 1. Perform Algorithm 1 to obtain the available time of each product
Step 2. Perform Algorithm 3 to obtain the initial solution. Set $\omega = 1$ and $it = 0$
Step 3. Select a neighborhood solution $V'$ from the NEI$_\omega$ of $V$
Step 4. Apply Algorithm 2 for $V'$, obtaining $V''$
Step 5. If $A_{\max}(V'') < A_{\max}(V)$, then set $\omega = 1$ and $V = V''$, else set $\omega = \omega + 1$
Step 6. If $\omega > \omega_{max}$, then set $\omega = 1$
Step 7. Set $it = runningtimeofthealgorithm$
Step 8. If $it \leq itmax$, then go to step 3, else go to step 9
Step 9. Output $V$
```

Costa et al. (2017) stated that the VNS metaheuristic based on a unique neighborhood structure can be very effective and provided the less is more approach (LIMA) on metaheuristic design. In this paper, the modified swap local search operator is regarded as the so-called unique neighborhood structure and we apply it in our implementation of a LIMA-VNS metaheuristic.

6 Computational experiments

In this section, we give the computational results of the developed VNS approach on the proposed scheduling problem and a comparison with other different metaheuristics. The selected metaheuristics in the comparative experiments have shown good performances on various types of scheduling problems, including VNS_R (Roshanaei et al. 2009), SA_C (Chen et al. 2017), and TS_E (El-Yaakoubi et al. 2017). The algorithms were coded in C++ language on an Intel Core 7-6700 with 16 G RAM, running Windows 7.

Algorithms 1 and 2 are used in all the metaheuristics which are compared by measuring the relative percentage deviation (RPD) between the reported makespan and the best-known makespan. The RPD is calculated as:

$$\text{RPD}_{\text{alg}} = \frac{A_{\max}(\text{alg}) - A_{\max}(\text{best})}{A_{\max}(\text{best})} \tag{10}$$

where $A_{\max}(\text{alg})$ is the makespan obtained by the metaheuristic alg while $A_{\max}(\text{best})$ denotes the best-known solution. Since the objective is to minimize the makespan, a smaller RPD means a better result. For each problem instance, 20 replications are operated for all the metaheuristics. Table 3 gives a summary of the attributes and levels of the generated instances. Since our problem has not been considered before, we surveyed real-world equipment manufacturing companies to obtain the data for the complete experiments. In order to get reasonable experimental data, we

Table 3 Attributes and levels

Attributes	Levels
Number of products	$N = 50, 100, 150$
Number of semi-product manufacturers	$g = 3, 6, 9$
Number of assembly manufacturers	$G = 2, 4, 6$
The deteriorating rates for setup operations	$\theta, \delta = U[0.01, 0.1]$
The normal processing time of the product components	$t_{i,m} = U[1, 10]$
The normal processing time of the products	$a_I = U[5, 10]$
The delivery time	$D_I = U[1, 5]$

also refer to the literature related to this article in recent years. For example, based on existing works such as Roshanaei et al. (2009) and Ahmadizar and Farhadi (2015), the deteriorating rates are in $U[0.01, 0.1]$ while the normal processing times are in $U[1, 10]$. Based on the relationship between processing time and transportation time, we set the delivery time in U $[1, 5]$. In real-world manufacturing, the assembly processing time is more fixed and longer than the component production time. Hence, we set the normal processing time of the products as $U[5, 10]$. Based on the levels of the attributes N, g, and G, 27 problem instances are generated. We use the PMA(x, y, z) to denote the instance with x products, y semi-product manufacturers, and z assembly machines.

In the experiment, we find that the metaheuristics converge to a good result in 50 s, even for the problem instance of the largest scale such as PMA$(150, 9, 6)$, which is shown in Fig. 4. Hence, we determine that each metaheuristic stops iterating after 50 s for each problem instance to achieve a fair comparison. In addition, in Fig. 4, LIMA-VNS also shows an advantage in convergence speed over the other three metaheuristics. Table 4 reports the summary of the results for the metaheuristics in 27 problem instances. The average objective value (Ave), the relative percentage deviation (RPD), and the variance (VAR) measured over the derived were calculated for each problem instance. For some small-scale experiments, such as PMA$(50, 3, 2)$ and PMA $(50, 3, 4)$, the other three metaheuristics may perform as well as LIMA-VNS. However, for most of the problem instances, average objective values and standard deviations obtained by LIMA-VNS are much smaller than those obtained by others, especially for the large-scale problem instances, e.g., PMA$(150, 3, 4)$ and PMA$(150, 9, 4)$. From Table 4, we can conclude that LIMA-VNS results in the best solution of Ave, VAR, and RPD among the four metaheuristics. From Fig. 5, we can see that solutions obtained by LIMA-VNS are gathered in the lower left corner of the coordinate system. However, for VNS_R, SA_C, and TS_E, the distribution of points on the graph is more dispersed compared to LIMA-VNS. Hence, it is shown that the designed LIMA-VNS can solve the problem stably. Based on the above computational experiment, we can infer that the LIMA-VNS has the best performance in converge speed, solution quality, and robustness. From the results of the experiment, we can summarize the contribution of the proposed method to real-world production. First, the proposed intelligent scheduling algorithm is easy to implement and can effectively improve the scientific nature of decision-making. Second, our algorithm can produce better results in a given time, that is, the application of the algorithm can reduce production time, thereby reducing the production and operating costs of the manufacturing companies. Moreover, timely delivery will increase user satisfaction and loyalty, thus providing long-term benefits for the enterprise. Finally, the LIMA-VNS is versatile and can solve other optimization decision problems for enterprises.

7 Conclusion

Intelligent decision-making in complex product manufacturing systems is important for the realization of smart manufacturing. This paper investigates a coordinated production, delivery, and assembly scheduling problem with a deteriorating effect and a learning effect. Through the discussion of the problem characteristics, we propose the structural properties for the serial-batching scheduling problem and the assembly scheduling problem under the constraints of component finishing time and transportation, respectively. For the serial-batching scheduling problem, an optimization scheduling rule has been developed to solve the problem in polynomial time. For the assembly scheduling problem, an improvement strategy has been proposed based on the structural properties. Since the problem is NP-hard, we have designed a LIMA-VNS to find an approximately optimal solution in a reasonable time. To validate the performance of the designed LIMA-VNS, we compare it with other three metaheuristics from the latest literature which includes VNS_R, SA_C, and TS_E. Via computational experiments, we show that our approach finds the best solutions within a given time, especially for large-scale problems. In this paper, the designed coding scheme, neighborhood structures, and the metaheuristic are effective and easy to implement Those methods can be used in real-life manufacturing to increase productivity, which in turn will improve the competitiveness of all participants in the global manufacturing system.

There are some topics worthy of consideration in our future work. On a theoretical level, one important future research direction would be to model the multistage problems for the coordinating manufacturing. Another potential direction would be to extend the LIMA-VNS to further different optimization problems with effective neighborhood structures. On a practical level, we would consider the different requests of the manufacturers, such as cost minimization,

Table 4 Summary of results for all instances

No.	PMA	LIMA-VNS			SA_C			VNS_R			TS_E		
		Ave	RPD (%)	VAR (%)	Ave	RPD (%)	VAR (%)	Ave	RPD (%)	VAR (%)	Ave	RPD (%)	VAR (%)
1	(50,3,2)	769.47	0.00	0.00	769.61	0.02	0.00	769.52	0.01	0.00	769.69	0.03	0.00
2	(50,3,4)	710.20	0.01	0.00	710.23	0.01	0.00	710.21	0.01	0.00	710.22	0.01	0.00
3	(50,3,6)	756.42	0.01	0.00	756.44	0.01	0.00	756.43	0.01	0.00	756.45	0.02	0.00
4	(50,6,2)	2264.62	0.42	0.02	2308.78	2.37	0.27	2358.03	4.56	1.01	2348.06	4.12	0.57
5	(50,6,4)	680.48	0.00	0.00	680.60	0.02	0.00	680.56	0.02	0.00	680.63	0.03	0.00
6	(50,6,6)	702.08	0.01	0.00	702.16	0.02	0.00	702.17	0.02	0.00	702.17	0.02	0.00
7	(50,9,2)	1238.23	0.01	0.00	1238.50	0.01	0.00	1238.74	0.02	0.00	1238.84	0.02	0.00
8	(50,9,4)	713.38	1.79	0.09	729.05	4.02	0.37	723.60	3.24	0.31	730.53	4.23	0.46
9	(50,9,6)	464.37	0.02	0.00	464.37	0.02	0.00	464.38	0.03	0.00	464.38	0.03	0.00
10	(100,3,2)	2536.05	6.74	0.05	2698.51	13.58	0.54	2992.63	25.96	0.71	2635.25	10.92	0.24
11	(100,3,4)	1223.24	1.23	0.02	1292.97	7.00	0.07	1396.06	15.53	1.79	1239.59	2.59	0.02
12	(100,3,6)	1096.63	0.64	0.00	1119.83	2.77	0.00	1146.22	5.19	0.01	1096.83	0.66	0.00
13	(100,6,2)	23,528.62	14.17	0.93	24,729.96	20.00	0.93	24,814.98	20.41	0.97	26,282.33	27.53	3.16
14	(100,6,4)	1478.27	0.70	0.01	1540.68	4.96	0.02	1623.60	10.60	0.05	1499.03	2.12	0.01
15	(100,6,6)	1222.93	0.31	0.00	1249.47	2.49	0.00	1294.10	6.15	0.02	1223.81	0.38	0.00
16	(100,9,2)	3805.77	1.24	0.04	4006.93	6.60	0.39	4072.06	8.33	0.05	3940.44	4.83	0.08
17	(100,9,4)	1472.96	0.73	0.01	1505.64	2.97	0.01	1556.29	6.43	0.01	1484.08	1.49	0.02
18	(100,9,6)	2041.78	2.15	0.01	2081.94	4.16	0.06	2179.48	9.04	0.04	2080.25	4.07	0.07
19	(150,3,2)	3568.61	5.17	0.08	4107.08	21.04	0.13	4348.86	28.17	0.92	4361.53	28.54	2.58
20	(150,3,4)	10,359.23	18.72	0.47	12,858.45	47.36	1.08	13,605.06	55.92	3.43	14,298.36	63.86	4.23
21	(150,3,6)	3058.50	1.35	0.02	3292.57	9.11	0.04	3435.58	13.85	0.03	3269.74	8.35	0.03
22	(150,6,2)	249,467.08	33.28	1.95	260,103.05	38.96	2.26	264,353.46	41.23	4.11	296,556.27	58.43	2.51
23	(150,6,4)	3241.07	2.33	0.02	3468.76	9.55	0.03	3566.61	12.61	0.03	3499.21	10.48	0.02
24	(150,6,6)	3610.01	1.87	0.02	3941.70	11.23	0.06	4101.31	15.74	0.04	3996.85	12.79	0.19
25	(150,9,2)	848,753.17	9.15	0.66	928,265.50	19.37	2.14	899,687.58	15.70	0.87	972,045.95	25.03	2.15
26	(150,9,4)	5099.84	8.08	0.09	5533.57	17.27	0.40	5673.44	20.23	0.06	5837.21	23.70	1.02
27	(150,9,6)	10,369.73	6.84	0.04	11,272.15	16.13	0.06	11,655.14	20.08	0.07	11,753.02	21.09	0.07
Average		43,860.47	4.33	0.17	47,460.31	9.67	0.33	46,663.19	12.56	0.54	50,574.1	11.68	0.65

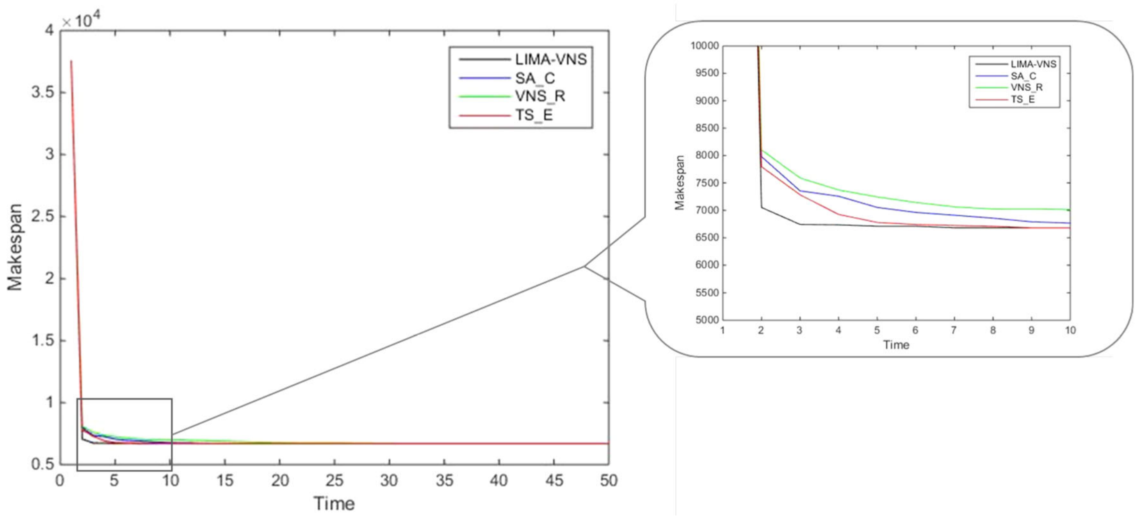

Fig. 4 The converge curves of the metaheuristics over time on problem PMA(150, 9, 6)

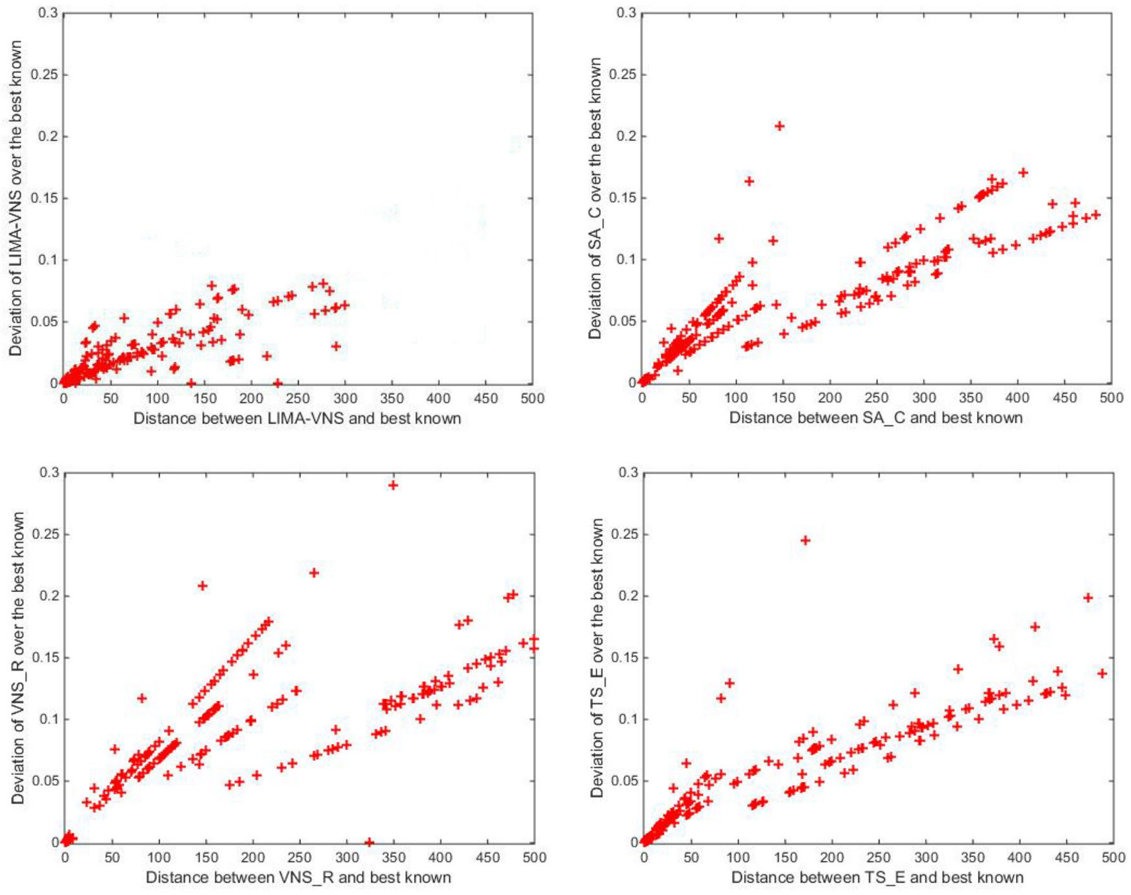

Fig. 5 Plots of distances between solutions in each replication and the best-known solution

lean production and design, more effective VNS, and heuristic algorithms to improve the manufacturing efficiency.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 71871080, 71601065, 71231004, 71690235, 71501058, 71601060, 71922009), and Innovative Research Groups of the National Natural Science Foundation of China (71521001), the Humanities and Social Sciences Foundation of the Chinese Ministry of Education (No. 15YJC630097), and Base of Introducing Talents of Discipline to Universities for Optimization and

Decision-making in the Manufacturing Process of Complex Product (111 project).

References

Ahmadizar, F., & Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers & Operations Research, 53,* 194–205.

Alam, T., & Raza, Z. (2018). A bacterial foraging-based batch scheduling model for distributed systems. *International Journal of Bio-Inspired Computation, 11*(1), 16–26.

Allahverdi, A., & Al-Anzi, F. S. (2006). Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times. *International Journal of Production Research, 44*(22), 4713–4735.

Azzouz, A., Ennigrou, M., & Ben Said, L. (2018). Scheduling problems under learning effects: Classification and cartography. *International Journal of Production Research, 56*(4), 1642–1661.

Bouffard, V., & Ferland, J. A. (2007). Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem. *Journal of Scheduling, 10*(6), 375–386.

BoussaïD, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences, 237,* 82–117.

Chen, J. C., Chen, Y. Y., Chen, T. L., & Lin, J. Z. (2017). Comparison of simulated annealing and tabu-search algorithms in advanced planning and scheduling systems for TFT-LCD colour filter fabs. *International Journal of Computer Integrated Manufacturing, 30*(6), 516–534.

Cheng, T. E., Hsu, C. J., Huang, Y. C., & Lee, W. C. (2011). Single-machine scheduling with deteriorating jobs and setup times to minimize the maximum tardiness. *Computers & Operations Research, 38*(12), 1760–1765.

Cheng, T. E., Lai, P. J., Wu, C. C., & Lee, W. C. (2009). Single-machine scheduling with sum-of-logarithm-processing-times-based learning considerations. *Information Sciences, 179*(18), 3127–3135.

Coffman, E. G., Jr., Garey, M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing, 7*(1), 1–17.

Costa, L. R., Aloise, D., & Mladenović, N. (2017). Less is more: Basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. *Information Sciences, 415,* 247–253.

El-Yaakoubi, A., El-Fallahi, A., Cherkaoui, M., & Hamzaoui, M. R. (2017). Tabu search and memetic algorithms for a real scheduling and routing problem. *Logistics Research, 10*(7), 1–18.

Fan, W., Pei, J., Liu, X., Pardalos, P. M., & Kong, M. (2018). Serial-batching group scheduling with release times and the combined effects of deterioration and truncated job-dependent learning. *Journal of Global Optimization, 71*(1), 147–163.

Framinan, J. M., Perez-Gonzalez, P., & Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research, 273*(2), 401–417.

Fu, Y., Ding, J., Wang, H., & Wang, J. (2018). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Applied Soft Computing, 68,* 847–855.

Gawiejnowicz, S. (2008). *Time-dependent scheduling*. Berlin: Springer.

Hariri, A. M. A., & Potts, C. N. (1997). A branch and bound algorithm for the two-stage assembly scheduling problem. *European Journal of Operational Research, 103*(3), 547–556.

Jia, J., & Mason, S. J. (2009). Semiconductor manufacturing scheduling of jobs containing multiple orders on identical parallel machines. *International Journal of Production Research, 47*(10), 2565–2585.

Kress, D., Barketau, M., & Pesch, E. (2018). Single-machine batch scheduling to minimize the total setup cost in the presence of deadlines. *Journal of Scheduling, 21*(6), 595–606.

Liao, C. J., Lee, C. H., & Lee, H. C. (2015). An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Computers & Industrial Engineering, 88,* 317–325.

Liu, P., Yi, N., Zhou, X., & Gong, H. (2013). Scheduling two agents with sum-of-processing-times-based deterioration on a single machine. *Applied Mathematics and Computation, 219*(17), 8848–8855.

Lu, S., Liu, X., Pei, J., Thai, M. T., & Pardalos, P. M. (2018). A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. *Applied Soft Computing, 66,* 168–182.

Luo, J., Liu, Z., & Xing, K. (2018). Hybrid branch and bound algorithms for the two-stage assembly scheduling problem with separated setup times. *International Journal of Production Research, 57,* 1–15.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100.

Mladenović, N., Todosijević, R., & Urošević, D. (2016). Less is more: Basic variable neighborhood search for minimum differential dispersion problem. *Information Sciences, 326,* 160–171.

Mor, B., & Mosheiov, G. (2011). Single machine batch scheduling with two competing agents to minimize total flowtime. *European Journal of Operational Research, 215*(3), 524–531.

Pei, J., Liu, X., Pardalos, P. M., Migdalas, A., & Yang, S. (2017). Serial-batching scheduling with time-dependent setup time and effects of deterioration and learning on a single-machine. *Journal of Global Optimization, 67*(1–2), 251–262.

Potts, C. N., Sevast'Janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., & Zwaneveld, C. M. (1995). The two-stage assembly scheduling problem: Complexity and approximation. *Operations Research, 43*(2), 346–355.

Renna, P., & Perrone, G. (2015). Order allocation in a multiple suppliers-manufacturers environment within a dynamic cluster. *The International Journal of Advanced Manufacturing Technology, 80*(1–4), 171–182.

Roh, J., Hong, P., & Min, H. (2014). Implementation of a responsive supply chain strategy in global complexity: The case of manufacturing firms. *International Journal of Production Economics, 147,* 198–210.

Roshanaei, V., Naderi, B., Jolai, F., & Khalili, M. (2009). A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems, 25*(6), 654–661.

Shahvari, O., & Logendran, R. (2018). A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics, 195,* 227–248.

Shokrollahpour, E., Zandieh, M., & Dorri, B. (2011). A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. *International Journal of Production Research, 49*(11), 3087–3103.

Tao, F., Cheng, Y., Zhang, L., & Nee, A. Y. (2017). Advanced manufacturing systems: Socialization characteristics and trends. *Journal of Intelligent Manufacturing, 28*(5), 1079–1094.

Wang, J. B. (2007). Single-machine scheduling problems with the effects of learning and deterioration. *Omega, 35*(4), 397–402.

Wang, X., & Cheng, T. E. (2007). Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan. *European Journal of Operational Research, 178*(1), 57–70.

Wang, J. B., Liu, M., Yin, N., & Ji, P. (2017). Scheduling jobs with controllable processing time, truncated job-dependent learning and deterioration effects. *Journal of Industrial & Management Optimization, 13*(2), 1025–1039.

Wen, Y., Xu, H., & Yang, J. (2011). A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system. *Information Sciences, 181*(3), 567–581.

Wu, C. C., Wang, D. J., Cheng, S. R., Chung, I. H., & Lin, W. C. (2018). A two-stage three-machine assembly scheduling problem with a position-based learning effect. *International Journal of Production Research, 56*(9), 3064–3079.

Yang, S., Wang, J., Shi, L., Tan, Y., & Qiao, F. (2018). Engineering management for high-end equipment intelligent manufacturing. *Frontiers of Engineering Management, 5*(4), 420–450.

Yin, Y., Cheng, T. C. E., Hsu, C. J., & Wu, C. C. (2013). Single-machine batch delivery scheduling with an assignable common due window. *Omega, 41*(2), 216–225.

Yin, Y., Cheng, T. C. E., Wang, D., & Wu, C. C. (2015). Improved algorithms for single-machine serial-batch scheduling with rejection to minimize total completion time and total rejection cost. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46*(11), 1578–1588.

Yin, Y., Wang, Y., Cheng, T. C. E., Liu, W., & Li, J. (2017). Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. *Omega, 69,* 17–28.

Yin, Y., Wang, Y., Cheng, T. C. E., Wang, D. J., & Wu, C. C. (2016). Two-agent single-machine scheduling to minimize the batch delivery cost. *Computers & Industrial Engineering, 92,* 16–30.

Yin, Y., Yang, Y., Wang, D., Cheng, T. C. E., & Wu, C. C. (2018). Integrated production, inventory, and batch delivery scheduling with due date assignment and two competing agents. *Naval Research Logistics (NRL), 65*(5), 393–409.

Zhang, B., Pan, Q. K., Gao, L., & Zhang, X. L. (2018). A hybrid variable neighborhood search algorithm for the hot rolling batch scheduling problem in compact strip production. *Computers & Industrial Engineering, 116,* 22–36.