Applied Soft Computing Journal 98 (2021) 106896

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Multi-task processing oriented production layout based on evolutionary programming mechanism

Zhao-Hui Sun [a,*], Di Liang [b,*], Zilong Zhuang [a], Liang Chen [a], Xinguo Ming [a]

[a] Department of Industrial Engineering and Management, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China
[b] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

ARTICLE INFO

Article history:
Received 6 November 2019
Received in revised form 23 September 2020
Accepted 4 November 2020
Available online 13 November 2020

Keywords:
Multi-task processing
Production layout evolution
Relevance between machines
Earliest finish time
Evolution programming

ABSTRACT

Traditional job scheduling is an independent optimization phase without considering material flow among machines. However, in actual production, the machine layout will affect material flow time, therefore affecting job completion time. In this paper, a novel multi-task-oriented production layout problem is proposed, with the goal to optimize material flow by adjusting machine placement. To solve this problem, a production layout evolution (PLEV) framework is established, which models it as a two-stage optimization problem. In the first stage, a concept of 'relevance between machines (RBM)' is proposed. The RBM measures the relationship between two machines according to the processing data. A machine placement scheme is designed based on RBM. In the second stage, a job scheduling rule based on the earliest finish time (EFT) is adopted to evaluate production layouts. Then, through an evolutionary programming mechanism, the production layout can evolve to optimize makespan. Experiments are carried out to verify the feasibility and performance of the proposed PLEV framework. The results demonstrate that promising product layouts for multi-task processing can be obtained by using this framework.

1. Introduction

The shop scheduling problem can be stated as arranging jobs to a set of machines in a manufacturing system [1]. Each job including one or more operations passes through the machines following a certain order [2]. Job shop scheduling (JSS) is a classic scheduling problem, where each job has a predefined and specific processing order [3,4]. If jobs visit the machines in the same order, the problem is called flow shop scheduling (FSS) [5]. If there is no precedence relation between operations, i.e., the processing order is arbitrary, it is called open shop scheduling (OSS) [6,7]. As an extension of JSS [8], OSS is an NP-hard combinatorial optimization problem [9]. Moreover, since operations can be processed in any order in OSS, the solution space is larger compared to JSS [10] and FSS [11]. The current OSS problems mainly focus on job scheduling problem based on fixed machine layout and ignore the effectiveness of machine layout optimization for job scheduling [12–14], which is exactly the primary focus of this paper.

The OSS problem is widely studied in workshop production and processing [12,13]. Different operation sequences will affect the *makespan* (i.e., the time when the last job is completed) [14]. In order to improve production efficiency, many algorithms are proposed to minimize the makespan [15–17], including exact algorithms (e.g., the branch and bound algorithm [17]) as well as heuristic and meta-heuristic algorithms [1,15,16]. Ahmadizar et al. [1] incorporated a local optimization heuristic into a genetic algorithm to solve the OSS problem with the goal to minimize the makespan. Pongchairerks et al. [15] developed a two-level particle swarm optimization (PSO) algorithm to solve the OSS problem. In this algorithm, the upper-level PSO algorithm fine-tunes the parameters of the lower-level PSO algorithm, which generates the operation schedules. Bai et al. [16] investigated a flexible OSS problem with static and dynamic versions. For large-scale problems, they designed a general dense scheduling algorithm based on the previous work [18] to accelerate convergence. For moderate-scale problems, they employed the differential evolution algorithm to obtain high-quality solutions. Sheikhalishahi et al. [19] proposed a multi-objective OSS problem to minimize the total makespan, minimize the total human error, and maximize the machine availability. Three meta-heuristic multi-objective optimization algorithms are developed to find near-optimal solutions. However, most OSS studies schedule operations to machines by considering the processing time and ignore the material flow time (i.e., material transportation time between machines) due to the physical distance between machines. That is, assuming that after one operation of a job is

* Corresponding authors.
E-mail addresses: zh.sun@sjtu.edu.cn (Z.-H. Sun), csdiliang@mail.scut.edu.cn (D. Liang).

https://doi.org/10.1016/j.asoc.2020.106896
1568-4946/© 2020 Elsevier B.V. All rights reserved.

completed, the next operation of this job can be performed immediately if the machine it needs is available. In some practical multi-task processing applications, the material flow time has a significant impact on the completion time of the job processing, which cannot be ignored. The material flow time between machines is mainly determined by the layout of machines in the production space and transportation speed of material. Mejía et al. [20] studied an OSS problem with travel times between machines and proposed a self-tuning variable neighborhood search algorithm to solve it. However, the travel times are generated randomly, which means that the machines have been placed and the machine layout is fixed before an OSS process. Different from that, in this study, it is assumed that the production space is empty and machines are not placed in the initial state. This paper considers both processing time and material flow time in OSS and focuses on performing material flow optimization by finding the optimal machine layout.

The current production facilities (machines) layout problems in the literature are mainly based on pre-defined and fixed material handling paths [21]. Taghavi et al. [22] studied an integrated facility layout design and flow assignment problem. They proposed a comprehensive heuristic method based on two heuristic rules and a perturbation algorithm to solve it. Zuo et al. [23] presented a multi-row parallel machine layout problem. In this problem, the machines are divided into several groups, each group contains the same kind of parallel machines. It is stipulated that the material flows only between different machine groups, and the total material flow cost is minimized by optimizing the position of all machines. In addition, some meta-heuristic algorithms were used to solve the production layout problem [24–30]. Sahin [31] applied a simulated annealing algorithm for the bi-objective production layout problem. Safarzadeh et al. [32] studied an extended multi-row facility layout problem with fuzzy clearances. A genetic algorithm approach was suggested to minimize the material handling and lost opportunity costs. Hosseini-Nasab et al. [33] introduced a PSO algorithm integrated with a simple and fast simulated annealing to solve a dynamic facility layout problem, so as to minimize the sum of handling and re-layout costs. Most of above papers aim to reduce the material handling cost based on the deterministic material flow information (i.e., known processing routes) [34]. That is, the quality of the facility layout can be easily evaluated based on known information. Unlike these machine layout problems, in this study, since the routing of each job between machines in OSS can be arbitrarily chosen, the material flow between machines is unknown in advance. The only information about material is the operation processing time. The uncertainty of material flow between these machines poses a challenge to the optimization of production layout. It means that the quality of facility layout can only be evaluated after an OSS process, which is much more difficult.

This paper considers an OSS problem with material flow time, termed OSSMFT. It aims to reduce the material flow time between machines by optimizing production layout, i.e., by determining the optimal placement of machines in the workshop. As a result, actual job completion time is also reduced. The production layout problem for OSSMFT can be regarded as a combination of facility layout and open shop scheduling, which is much more complex and challenging to solve. Firstly, in order to determine a relatively good initial production layout, a layout scheme is designed based on ‘relevance between machines’ (RBM). The ‘relevance’ is defined to measure relationships between machines according to the processing data. If two machines are used for the same job, the two machines are related. Then, the RBM is equal to the number of jobs that use them simultaneously. Secondly, since the evaluation of production layout can only be achieved through the stage of subsequent job processing, if existing evolutionary algorithms are adopted to evaluate the production layout, it will take too much time overhead due to a large number of iterations. For example, Hosseinabadi et al. [35] used an extended GA, a classic evolutionary algorithm, to solve open-shop scheduling problems. Since GA is a population-based and random algorithm, as the problem size increases, a larger-scale population and more generations are required to converge to good results. In [35], the population size and maximum generation number can reach 250 and 7000, respectively. When the number of jobs is 50 and the number of machines is 30, it will take more than 10 h of CPU time to obtain the optimal job schedule. If GA is applied to perform job scheduling to obtain an excellent makespan for a certain layout, then the entire problem solving will be very time-consuming. Therefore, in order to properly evaluate a production layout in a shorter time, a highly efficient job scheduling rule based on the Earliest Finish Time (EFT) [36,37] is adopted. It is a classical deterministic approach for solving job scheduling problems. Finally, as the initial machine placement scheme is not necessarily optimal, an evolutionary programming mechanism is employed to further optimize production layout.

The main contributions of this paper are as follows:

(1) A production layout problem for OSSMFT is proposed (PLP-OSSMFT), which aims to find an optimal production layout scheme to optimize material flow between machines, so as to minimize makespan of job processing.

(2) A framework termed Production Layout EVolution (PLEV) is established to solve the PLP-OSSMFT. It consists of three key components: initial production layout, production layout assessment, and an evolutionary programming mechanism.

(3) A novel initial production layout scheme is defined for initializing the population: By quantifying the degree of correlation between machines based on the information of job processing, closely related machines can be placed in the production units as close as possible.

(4) For the evaluation of production layouts, an EFT-based job scheduling rule is proposed, which takes operation processing time and material flow time into account. For multi-task processing, the EFT-based rule can evaluate the status of production layouts within an acceptable computational time while ensuring a good scheduling result.

(5) In order to evolve production layouts, an evolutionary programming mechanism is proposed that encodes production layouts and defines mutation and selection operators.

The rest of this paper is organized as follows. Section 2 elaborates OSSMFT and the production layout problem (i.e., PLP-OSSMFT). Section 3 presents the PLEV framework for solving PLP-OSSMFT and explains the design challenges for each component in the proposed framework. Section 4 details the design of PLEV and analyzes computational complexity of each component. Experiments demonstrate the feasibility and effectiveness of the PLEV framework in Section 5. Section 6 summarizes and points out future research directions.

2. Problem formulation

2.1. Introduction to OSSMFT

Consider an OSSMFT problem with size $n\times m$, in which n jobs need to be processed on m machines. The basic notations are listed in Table 1. J={$J_1, J_2,\ldots, J_n$} represents job set, and $M = \{M_1, M_2,\ldots, M_m\}$ represents machine set. Each job J_i ($1\leq i\leq n$) needs to be processed on $|Q_i|$ machine, where Q_i is a subset of M. The operation of job J_i on machine M_j ($1\leq j\leq m$) is defined as o_{ij}, and the operation set is denoted as $O = \{o_{ij}|\ 1\leq i\leq n,\ 1\leq j\leq m\}$, $|O|\leq n\times m$. All the operations in O can be processed in any order.

Table 1
Basic notations for the OSSMFT problem.

Symbol	Definition
n	Number of jobs.
m	Number of machines.
J_i	ith job.
M_j	jth machine.
o_{ij}	operation of J_i on M_j.
Q_i	Set of machines that are used by J_i.
p_{ij}	Processing time of o_{ij}.
f_{jk}	Material flow time between M_j and M_k.
ts	Speed of material transportation.
S_{ij}	Start execution time of o_{ij}.
C_{ij}	Completion time of o_{ij}.

Table 2
Example of processing time.

	M_1	M_2	M_3	M_4	M_5	M_6
J_1	0.5	2	4	0	2	7
J_2	0	0	2	1	7	6
J_3	0	1	4	5	2	3
J_4	2	3	0	7	1	4
J_5	2	0	0	7	1	6
J_6	3	3	3	3	3	3
J_7	8	1	2	4	5	2

For example, if a job is decomposed into u ($1 \leq u \leq m$) operations, the number of possible processing sequences of the job is $u! = 1 \times 2 \times \ldots \times u$. The processing time of each operation o_{ij} is defined as p_{ij} ($p_{ij}>0$), and the processing time of all operations are limited to $[P_l, P_u]$, where P_l and P_u denote the lower and upper bounds of processing time, respectively.

After an operation o_{ij} of job J_i is processed, material is transported to the machine required for the next operation of this job. f_{jk} indicates the time it takes for the material to be transported between the machines M_j and M_k. The material flow time in this paper is symmetrical, this is, $f_{kj} = f_{jk}$. Therefore, the makespan of OSSMFT is determined by the processing time of jobs and the flow time of the material between machines. Table 2 is an example of processing time. If one job does not need to be processed on one machine, the corresponding processing time is zero.

In a fixed production layout, the OSSMFT problem can be defined to find a sequence of operations with the given processing times on each machine to minimize the completion time of the last operation [14]:

$$makespan = \max(C_{ij}) = \max(S_{ij} + p_{ij}) \tag{1}$$

where S_{ij} is the start execution time of the operation o_{ij}. It is the decision variable of the OSSMFT problem. And C_{ij} is the completion time of o_{ij}.

The following constraints on operation and machine processing need to be met during job scheduling:

(1) *Operation processing constraint:* Once an operation is started, it cannot be interrupted during the processing. Also, two or more operations of a job cannot be processed on different machines simultaneously [35]. For example, if an operation o_{ij} has a precedent o_{ik}, then it can be started only after o_{ik} is completed and the corresponding material of J_i has been transported to M_j. Considering the possible waiting for M_j to be released, the following restriction needs to be met.

$$C_{ij} \geq C_{ik} + f_{kj} + p_{ij} \tag{2}$$

(2) *Machine processing constraint:* Each machine can only process one operation at any time. For example, if o_{tj} is being processed on M_j when the material of o_{ij} is transported to M_j, then o_{ij} cannot be started until o_{tj} is completed and the machine M_j is released.

$$C_{ij} \geq C_{tj} + p_{ij} \tag{3}$$

2.2. Production layout problem for OSSMFT

Consider a workshop that is physically divided into $l \times w$ production units, each unit can accommodate only one machine. Assuming that the speed of material flow is constant, the material flow time depends physical distance (in 2D or 3D space) between the machines, which is calculated according to the coordinates of these machines. In this paper, the transportation of materials is carried out on Manhattan paths, which can maintain flexible adjustment ability to prevent collision when transferring materials, since there can be multiple Manhattan paths between two machines. The material flow time is calculated by

$$f_{kj} = \frac{Manh(M_k, M_j)}{ts} \tag{4}$$

where f_{kj} represents material flow time between M_j and M_k, $Manh(M_k, M_j)$ represents the Manhattan distance between M_k and M_j. ts represents the transportation speed. It is a parameter about production capacity. A larger ts means shorter transportation time between two production units. The Manhattan distance is calculated based on the coordinates of the production units where they are placed. Fig. 1 illustrates the coordinates of production units in a workshop with size 3×3. Then the Manhattan distance between M_k and M_j is calculated by

$$Manh(M_k, M_j) = |x_k - x_j| + |y_k - y_j| \tag{5}$$

where (x_k, y_k) and (x_j, y_j) are the coordinates of units where M_k and M_j are placed, respectively. The placement of machines needs to meet the following constraints:

$$Z_{jxy} = \begin{cases} 1, & \text{if } M_j \text{ is placed in } (x, y) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$\sum_{j=1}^{m} Z_{jxy} \leq 1,\ 1 \leq x \leq l,\ 1 \leq y \leq w \tag{7}$$

$$\sum_{x=1}^{l} \sum_{y=1}^{w} Z_{jxy} = 1 \tag{8}$$

where Z_{jxy} is the binary decision variable of the machine layout problem, such that $Z_{jxy} = 1$ represents machine M_j is placed in (x, y). Eq. (7) represents that each production unit can place at most one machine. If the number of production units is greater than the number of machines, then some production units are empty. Eq. (8) represents that all machines are placed and each machine is only placed in one unit.

In the initial state, machines are unplaced and the production units are empty. The PLP-OSSMFT problem can be defined as to place machines in the production units to optimize material flow, thereby minimizing makespan by Eq. (1). In general, the objective of the PLP-OSSMFT problem is Eq. (1), and the constraints are Eq. (2) to Eq. (8).

3. Production layout evolution framework

3.1. Overview of PLEV

Fig. 2 shows the proposed framework for the PLP-OSSMFT problem in this paper. Firstly, the initial machine placement determines the initial production layout status. In a certain layout

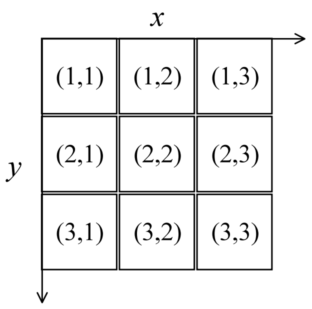


Fig. 1. Coordinates of production units in a workshop with size 3×3.

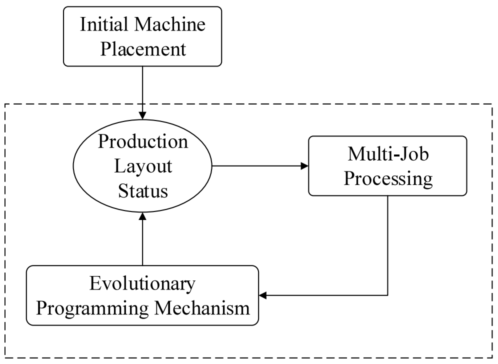


Fig. 2. The proposed PLEV framework.

status, the processing of jobs can be carried out. The goal of solving an OSSMFT is to minimize the makespan. The makespan is determined by the production layout and the job scheduling. If the job scheduling rule (e.g., earliest completion rule and evolutionary algorithms [6,35]) is determined, the difference of the completion time depends mainly on the different production layouts. That is, the actual job completion time can be used to evaluate the production layout status.

Since the machines required for the processing of each job and processing time of each operation are known in advance, this known processing information can be utilized to determine the initial production layout through a promising machine placement scheme. But for a certain rule, it is hard to obtain the optimal layout by just using it once for the complexity of the production layout problem. Therefore, the evolutionary programming mechanism is adopted to further optimize the initial production layout. The dashed box in Fig. 2 indicates that after repeated iterations, an improved layout scheme can be obtained.

3.2. Design challenges of PLEV

3.2.1. Challenge 1: initial machine placement scheme

A promising initial layout scheme will allow the production system to get a good production layout in the early stages of evolution. In the absence of any prior knowledge, the easiest way is to place each machine randomly. Since the machines used by each job are predetermined, and material flow between machines represents the 'relevance' between these machines, machine correlation can provide a reference for the layout. In addition, the different machines (Q_i) used by different jobs may cause some machines to be used intensively while others are idle most of the working time. Therefore, how to use the known processing information to determine the initial production layout is the first consideration.

An initial layout scheme based on machine relevance is proposed, which quantifies the correlation between machines and places machines according to the relevance between machines, so that relevant machines tend to be placed in nearby units. The initial layout scheme will be described in detail in Section 4.1.

3.2.2. Challenge 2: layout evaluation rule

Generally, if the production layout status is fixed, evolutionary algorithms can obtain a good result on the OSSMFT problem. However, the extensive iterations of evolutionary algorithms make the calculation process too time-consuming. It is not acceptable to evaluate the status of a certain production layout by high-complexity evolutionary algorithms, especially in some mass production applications. Therefore, an effective and efficient layout evaluation rule is in great need. Based on the above considerations, an EFT-based scheduling rule is adopted in this study, which select the earliest finished operation for a released machine, to evaluate production layouts. EFT comprehensively considers the operation processing time and material flow time. As the rule is a deterministic approach, it can quickly assess the status of the current production layout within an acceptable computational time. The EFT-based rule will be described in detail in Section 4.2.

4. Detailed design of production layout evolution

4.1. Initial machine placement scheme

Machines used by the same job are relevant and should be placed in the units with close positions. Furthermore, the relevance between machines should be quantified. Each machine is regarded as a node, then the existence of an edge between two machines depends on whether the two machines are used by the same job. The weight of the edge represents the number of jobs which use the two machines, i.e., RBM, and is calculated as

$$w_{jk} = \sum_{i=1}^{n} x_{ijk} \tag{9}$$

and

$$x_{ijk} = \begin{cases} 1, & \text{if } M_j \in Q_i \text{ and } M_k \in Q_i \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where i is the job index, j and k are machine indexes, w_{jk} is the weight of edge connecting machines M_j and M_k, x_{ijk} is an indicator variable, and Q_i is the set of machines used by job i. Eq. (9) represents that a higher RBM means more jobs using these two machines. Then they tend to be placed nearby to reduce the material flow time between them.

After calculating the relevance between each pair of machines, the relevant machines can be placed in nearby units. However, in actual production, a large number of jobs may cause a large number or even all of the machines to be related. Therefore, an evaluation mechanism is needed to measure the 'importance' of machines in order to determine which machines to place first. Inspired by PageRank [38,39], assessing the importance of a machine can be derived from the machines it is associated with. In this study, the sum of the weights of all the edges connected to each machine is calculated and regarded as an indicator of its importance. Machines with high importance are placed first.

The remaining machines, depending on the relevance with the machine that has been placed, tend to be placed near the machines with high relevance.

The specific steps are as follows:

(1) *Calculate the relevance between two machines*
For each machine pair, calculate the relevance according to Eqs. (9) and (10).

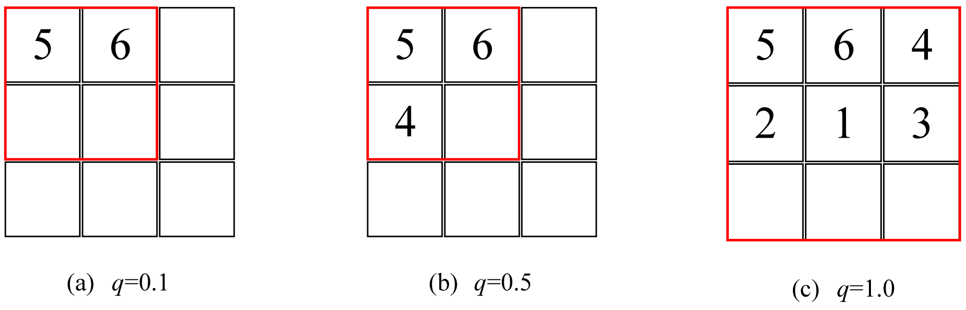


Fig. 3. Illustration of placing high-importance machines with different q values based on processing time in Table 2.

(2) *Assess the importance of each machine*
Calculate the sum of the weights of all edges for each machine node, and then normalize as the criterion of the machine importance by

$$I_j = \frac{\sum_{k=1}^{m} w_{jk}}{\sum_{j=1}^{m}\sum_{k=1}^{m} w_{jk}} \tag{11}$$

where I_j represents the importance of M_j, w_{jk} represents the weight of the edge between machines M_j and M_k. If $j = k$, w_{jk} is set to 0.

(3) *Place machines with high importance*
Rank the machines in order of importance from highest to lowest. Select top $\lfloor m\times q\rfloor$ as the 'high'-importance machines, where q is a predefined parameter (0<q<1). Then, the $\lfloor m\times q\rfloor$ machines are placed in turn into a unit block, with size $\left\lceil\sqrt{\lfloor m\times q\rfloor}\right\rceil \times \left\lceil\sqrt{\lfloor m\times q\rfloor}\right\rceil$, which is located in the center of the workshop. Note that when $\lfloor m\times q\rfloor$ is less than 2, $\lfloor m\times q\rfloor$ is set to 2 to ensure that at least two machines are used as reference for placement when placing subsequent machines. That is, $\lfloor m\times q\rfloor = \max\{\lfloor m\times q\rfloor, 2\}$. Taking the processing data in Table 2 as an example, the sequence of machines sorted by importance from highest to lowest is 5-6-4-2-1-3. Fig. 3 illustrates the placement of high-importance machines with different q values. The area enclosed by the red frame represents the central block. Figs. 3(a) and 3(b) show the central block with size 2×2 placed in the workshop with size 3×3. Fig. 3(c) shows the central block with size 3×3 placed in the workshop with size 3×3. It can be seen that q affects the number of high-importance and the placement of these machines. It should be noted that in a real production scenario, the topology of workshop can be arbitrary, such as a star or just one-dimensional row. And this method still works because it only requires that the topology of workshop has a relatively definite center area. The placement of important machines is mainly achieved by controlling the number of high-importance machines, and then matching the central units. Given that the number of high-importance machines is $\lfloor m\times q\rfloor$, the number of high-importance machines can be changed by the parameter q. For example, if the central block of workshop is a one-dimensional row, q can be set to $2/m$. It means that there are only two high-importance machines to place in the center in this step. Then the remaining machines are placed in empty units.

(4) *Place the remaining machines*
For each remaining unplaced machine, select R machines with the highest relevance among the machines already placed. In this paper, R is set to 2. That is, the machines with the highest and second highest relevance with each unplaced machine are recorded, denoted as R_1 and R_2. Then choose one of the unoccupied units to place so that it is closest to R_1 and R_2 in terms of Manhattan distance.

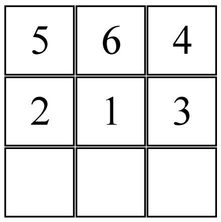


Fig. 4. An example of initial production layout status by Algorithm 1 based on processing time in Table 2. Assume that the workshop space size is 3×3 and q is set to 0.1.

The related process is shown in Algorithm 1, and an example is shown in Fig. 4. Note that this figure only illustrates the positional relationship among these machines and does not represent the determined position of them. For example, 5-6-4-2-1-3 in the first six units is equivalent to 2-1-3-5-6-4.

Algorithm 1: Initial machine placement based on RBM

```
Begin
1.  Calculate the relevance $w_{jk}$ between machines by Eq. (9) and Eq. (10);
2.  Assess the importance $I$ of each machine by Eq. (11);
3.  Sort the machines according to $I$, denoted as $MI$;
4.  for $j$=1 to $j$=$\lfloor m\times q\rfloor$
5.        Place $MI_j$ in the center $\lfloor m\times q\rfloor$ units of the workshop;
6.  end for
7.  for $j$=$\lfloor m\times q\rfloor$+1 to $j$=$m$
8.        Place $MI_j$ according to the relevance with machines that
      has been placed;
9.  end for
End
```

4.2. EFT-based layout evaluation rule

In order to make production layout evolve toward efficient production, an efficient method is needed to evaluate the quality of a production layout. This section details an evaluation rule based on EFT-based job scheduling.

Since the processing time of the operation is determined, if the start execution time of each operation is determined, its completion time can be calculated according to Eq. (1). For the convenience of illustration, assigning each operation to the corresponding machine is described as determining its start execution time in the following.

The specific steps of the EFT-based job scheduling rule are as follows:

(1) *Assign the first operation for machines*

Set the current time to 0. According to the EFT-based rule, each machine is assigned to one operation with the shortest processing time on it for the completion time of an operation started at time 0 is equal to its processing time. When selecting operations to be processed at time 0, it is necessary to satisfy the constraints in Section 2.1. That is, other operations belonging to the same job as the operation to be assigned are not allocated and the machine required for the operation is not occupied. The candidate operation set CO is defined as

$$CO = \{o_{ij} \mid ocp(J_i) = 0 \text{ and } ocp(M_j) = 0, \forall o_{ij} \in O\} \tag{12}$$

where $ocp(J_i)$ indicates whether the job corresponding to the operation o_{ij} has been assigned. If any operation of J_i has been assigned, $ocp(J_i) = 1$. $ocp(M_j)$ indicates whether the machine required for o_{ij} has been occupied. Note that in the initial state, all jobs are not assigned ($ocp(J_i) = 0$ for all jobs), and all machines are idle ($ocp(M_i) = 0$ for all machines). Then, select the operation with the shortest processing time in the candidate set CO, and assign it to the corresponding machine for processing. Set the start execution time S_{ij} of these already assigned operations to zero.

(2) *Assign remaining operations to be processed*

The idea of EFT-based scheduling rule is to assign an operation to a machine once it is idle, so that the operation is the earliest finished. If there are machines that are not occupied in (1) and need to perform production tasks, then assign operations to those machines first.

When all the machines that need to perform production tasks are occupied, scan all the operations that are being executed, set the current time ct as the completion time of the first completed operation, and release the corresponding machine, denoted as rm. If all the operations required to use rm have been assigned (i.e., the machine can be shut down after the last operation on it has been processed), continue to move along the timeline to find the next machine to be released. Otherwise, choose one operation from those jobs which have not been assigned to rm but need to be processed on rm, so that the completion time is the earliest. The start execution time of the operation is given by

$$S_{ij} = \begin{cases} C_{ij}^{ct} - p_{ij}, & \text{if } o_{ij} = \underset{o_{ij} \in O}{\operatorname{argmin}}(C_{ij}^{ct}) \\ NN, & \text{otherwise} \end{cases} \tag{13}$$

where C_{ij}^{ct} represents the completion time of o_{ij} if J_i is assigned to machine M_j at the current time ct (not necessarily the actual completion time). NN is a parameter which indicates the start execution time of o_{ij} has not been determined.

If the operation has a precedent o_{ik}, C_{ij}^{ct} is calculated by

$$C_{ij}^{ct} = \begin{cases} ct + p_{ij}, & \text{if } C_{ik} + f_{kj} \le ct \\ C_{ik} + f_{kj} + p_{ij}, & \text{otherwise} \end{cases} \tag{14}$$

where C_{ik} represents the actual completion time of o_{ik} and f_{kj} is the material flow time from M_k to M_j. If it has no precedent, then $C_{ij}^{ct} = ct$+p_{ij}. Eq. (14) indicates that if the operation o_{ik} has been processed and the corresponding material can be transported to the machine M_j before ct, the operation can be processed immediately without waiting; otherwise, the machine M_j needs to wait for the o_{ik} operation completion on M_k and transfer to M_j, the processing then can be started. For the calculation of C_{ij}^{ct}, (1) the o_{ik} operation is unfinished and (2) the operation is completed but material cannot be transported to M_j before ct, can be considered as one situation. That is, M_j needs to wait for the arrival of the material.

(3) *Continue with (2) until all the operations have been assigned. Then calculate makespan according to Eq. (1).*

The related process is shown in Algorithm 2, an example is shown in Fig. 5.

Algorithm 2: Production Layout Evaluation Rule based on EFT

```
Begin
1.  $ct$←0;            //$ct$ is the current time;
2.  Calculate the candidate set of operations $CO$ by Eq. (12);
3.  while $CO \neq \emptyset$ do
4.      Find $o_{ij}$ whose $p_{ij}$ is the smallest in $CO$;
5.      Assign $o_{ij}$ to $M_j$;
6.      $S_{ij}$←$ct$;            //$S_{ij}$ is the start execution time of $o_{ij}$
7.      $ocp(J_i)$←1;
8.      $ocp(M_j)$←1;
9.      Recalculate $CO$ by Eq. (12);
10. end while
11. while not all operation are assigned to machines do
12.     if there is a machine $M_j$ which has production task and $ocp(M_j)$=0
13.         $rm$←$M_j$;            //$rm$ is the released machine
14.         $ocp(M_j)$←1;
15.     else
16.         for each $o_{ij}$ that is being processed at current time $ct$
17.             If $C_{ij}$ is the smallest
18.                 $ct$←$C_{ij}$;            //$o_{ij}$ is completed
19.                 $rm$←$M_j$;
20.             end if
21.         end for
22.     end else
23.     if all operations that use $rm$ are completed
24.         Move to line 12 to find the next machine to be released;
25.     else
26.         for each $o_{ij} \in O$
27.             if $o_{ij}$ has not been assigned before and $M_j$=$rm$
28.                 Calculate $C_{ij}^{ct}$ by Eq. (14);
29.             end if
30.         end for
31.         Determine the next operation whose $C_{ij}^{ct}$ is the smallest,
            assign it to $rm$, and set its start execution time by Eq. (13);
32.     end else
33. end while
34. Calculate makespan according to Eq. (1);
End
```

4.3. PLEV based on evolutionary programming

As an important evolutionary algorithm, evolutionary programming [40,41] has been applied with success to many combinatorial optimization problems, such as robot path planning [42], flow shop scheduling [43], and placement of renewable distributed generators [44]. As PLP-OSSMFT can be regarded as a combinatorial problem, evolutionary programming is suitable to optimize productions layout based on the initial layout status, so that the production layout status can evolve in the direction of minimizing makespan.

(1) *Initialization Configurations*
After determining the initial machine layout, the machine numbers are sequentially mapped to codes of individuals with length $l \times w$. It should be noted that all N_p individuals in the population have the same initial genes (i.e., the initial codes), where N_p is the population size;

(2) *Mutation*
Randomly choose two units and exchange the machines placed in them;

(3) *Generate new population*
If a trial individual (after mutation) gets a shorter makespan compared with that of the original individual (before mutation), replace the individual with the trial individual, and then, select N_p individuals to form the next generation by a roulette wheel selection.

The related process is shown in Algorithm 3. Firstly, the initial production layout is determined by Algorithm 1 and is mapped

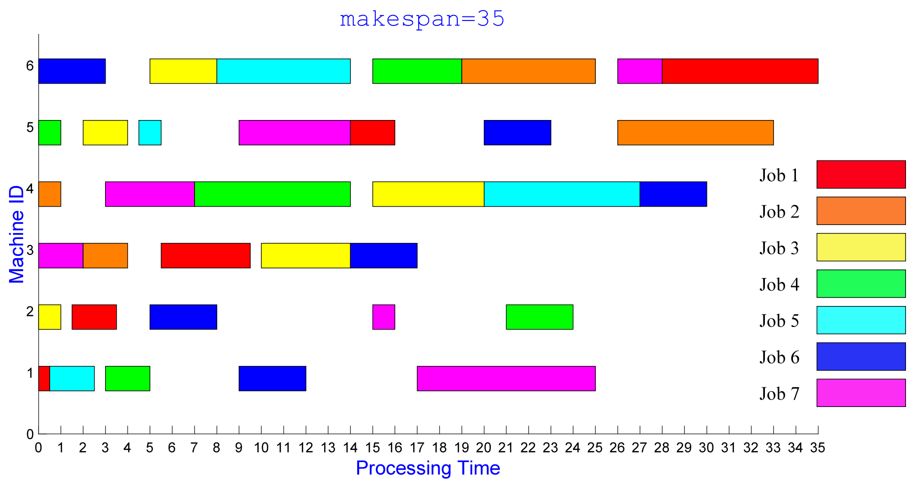


Fig. 5. Gantt chart of job scheduling by the EFT-based rule according to the processing time in Table 2 and the production layout in Fig. 4. *ts* is set to 1 here.

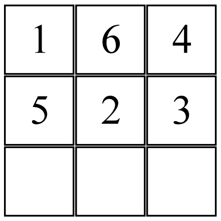


Fig. 6. An example of production layout after evolution according to the processing time in Table 2.

to the code of each individual. In an evolution process, mutation individuals are generated by the mutation operator. Then, map the code of each mutation individual to the production layout and calculate the corresponding material flow time. The makespan can be obtained by the 'earliest finish time' rule, which represents the quality of this layout. If the makespan of the mutation individual is shorter than the original individual, it will replace the original one. The best individual is updated if a shorter makespan is obtained. Finally, the new population is generated by a roulette wheel selection. The evolution process repeated until the termination condition is criterion. Note that these evolution operators used are not fixed but can be improved in the future, such as designing better population initialization methods and reasonably increasing the degree of mutation. The influence of different evolution operators on the algorithm performance will be explained in Section 5.4.

According to the data in Table 2, through the evolutionary programming mechanism, the final machine placement scheme can be obtained, shown in Fig. 6. The corresponding job scheduling is shown in Fig. 7. It can be seen that this machine layout scheme is globally optimal because machine 6 is always busy and has no idle time. For the sake of comparison, sequential placement following the order of machine index are taken as a reference, and the corresponding scheduling is shown in Fig. 8. It can be seen that for this example, the final completion time obtained by the sequential placement is shorter than that by the RBM-based initial machine placement scheme. However, through the evolutionary programming mechanism, the optimal machine placement scheme can be obtained.

4.4. Computational complexity analysis

In this section, the computational complexity of the initial machine layout and the evolution of production layouts for one generation are analyzed, respectively.

The complexity for initial machine placement is $O(m^2n)$ $+O(m^2)$+$O(m\log m)$+$O(m)$+$O(m^2+mlw)$, where $O(m^2n)$ represents the complexity of the relevance calculation. $O(m^2)$ indicates the complexity of computing the degree of importance of machines, $O(m\log m)$ is the complexity of sorting machines according to the degree of importance, $O(m)$ indicates the complexity of placing these machines with high degree of importance in the central area (block), and $O(m^2+mlw)$ indicates the complexity of placing the remaining machines, including the two steps of finding the R most relevant machines and selecting the rational unit. Generally, since $l\times w<n\times m$, the complexity of initial machine placement is $O(m^2n)$.

For the evolution of production layouts, the computational complexity of one evaluating production layout (i.e., the EFT-based scheduling rule) is analyzed first. Assume that each job has m operations to be processed, that is, a total of $n\times m$ operations need to be assigned to the corresponding machines for processing. The computational complexity of the scheduling rule is $O(m^2n)$+ $O(m^2n+n^2m)$, where $O(m^2n)$ is the complexity of assigning the first operation to each machine at the initial time, and $O(m^2n+n^2m)$ is the complexity of assigning all the remaining operations, including two steps of releasing each machine and assigning the job to be processed next for it. According to the operational rules of $O(\cdot)$, the complexity of the EFT-based scheduling rule is $O(m^2n+n^2m)$.

Therefore, the complexity of one generation is $O(N_p)$+ $O(N_pm^2n+N_pn^2m)$+$O(N_p^2)$, where $O(N_p)$ represents the complexity of initializing the population, mutation or updating the best individual. $O(N_pm^2n+N_pn^2m)$ is the complexity of fitness evaluation of the population, and $O(N_p^2)$ is the complexity of selection. According to the operational rules of $O(\cdot)$, the complexity of one generation is $O(N_pm^2n+N_pn^2m)$ $+O(N_p^2)$.

5. Experiments

In this section, the performance of PLEV on PLP-OSSMFT is investigated. The datasets used in the experiments are published on GitHub (https://github.com/liangdii/PLEV). All algorithms are implemented in C++ and run on a PC with a Core quad-core CPU i7 and 8 GB.

5.1. Multi-task production

Consider a production layout of a set of m machines in a workshop size $l\times w$. A set of jobs are randomly generated according to the parameter settings in Table 3, where ρ is a parameter

```
Algorithm 3: Production Layout Evolution
Begin
1.  $t \leftarrow 0$;
2.  Determine production layout $L(t)$ according to initial machine placement scheme;    //call Algorithm 1
3.  for $i$=1 to $i$=$N_p$                        //$N_p$ is the population size
4.      Map $L(t)$ to code of $X_i^t$;    //$X_i^t$ denotes the $i$-th individuals of the population in the $t$-th generation
5.      $mx_i(t) \leftarrow$makespan of $L(t)$;    //call Algorithm 2, $mx_i(t)$ denotes the corresponding makespan of $X_i^t$
6.  end for
7.  Keep the best individual as any one;
8.  while (termination criterion not met) do
9.      for $i$=1 to $i$=$N_p$
10.         Mutation: generate a mutation individual $Y_i^t$ for $X_i^t$;
11.         Map code of $Y_i^t$ to $L(t)$;
12.         $my_i(t) \leftarrow$makespan of $L(t)$;    //call Algorithm 2
13.         if ($my_i(t) < mx_i(t)$)
14.             $X_i^t \leftarrow Y_i^t$;
15.             $mx_i(t) \leftarrow my_i(t)$;
16.         end if
17.     end for
18.     Update the best individual to the one with the shortest makespan;
19.     $t \leftarrow t+1$;
20.     $X^t \leftarrow$select $N_p$ individuals from $X^{t-1}$ by a roulette wheel selection;
21. end while
End
```

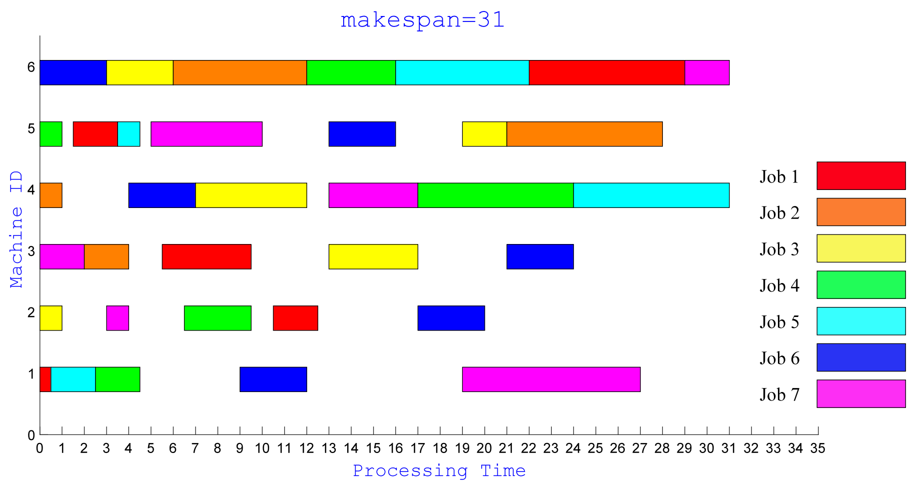


Fig. 7. Gantt chart of job scheduling after the evolution of production layout according to the processing time in Table 2.

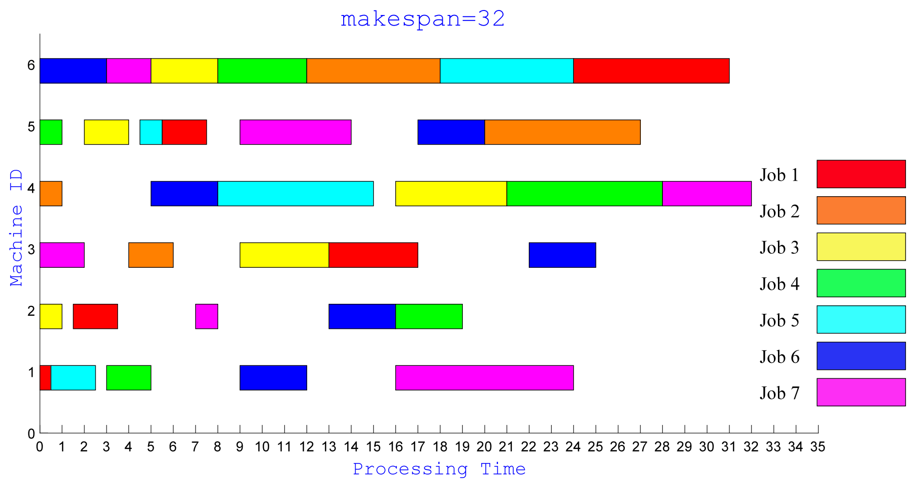


Fig. 8. Gantt chart of job scheduling by placing machines sequentially following the order of machine ID according to the processing time in Table 2.

Table 3
Parameter settings in PLEV.

ρ	P_l	P_u	ts	q	N_p	G
0.8	1	50	0.01	[0.1,0.3]	40	50

that controls whether an operation needs to be processed. The process of generating test datasets is described as follows. It consists of two steps: determining whether an operation is to be processed and setting the processing time of an operation. Firstly, for each operation, a random number is generated uniformly distributed in [0, 1]. If the random number is less than ρ, the operation needs to be processed and processing time is generated randomly according to a discrete uniform distribution in $[P_l, P_u]$; otherwise, no processing is required for it, that is, the processing time of the operation is set to 0. P_l and P_u represent the lower and upper bounds of time required for the operation processing, respectively. ts is the transportation speed of material. N_p and G are the population size and the maximum generation number.

Since the subsequent machine placement is based on the top $\lfloor m\times q\rfloor$ machines, q has a significant influence on the initial machine placement. It is found that different q may cause the initial machine placement results to be quite different. Therefore, the value of q is set from 0.1 to 0.3 with a step length of 0.01, and then call the layout evaluation rule. Finally, the initial layout status is determined by taking the machine layout with the shortest makespan in these solutions.

In order to facilitate the comparison of PLEV performance, initial machine placement (in Section 4.1), sequential placement and random placement are used as references. The random placement means randomly placing machines in the workshop several times. For fair comparison, the number of iterations is set to 2000 in random placement, to ensure that the number of iterations is equal to the maximal function evaluations (FEs = $N_p \times G = 40 \times 50$) for PLEV, and then the shortest makespan is adopted to compare. In addition, 30 independent runs with PLEV is performed for each case, with the best result, mean, standard deviation, and running time (in seconds) recorded. The experimental results (i.e., makespan) of machine layout in workshops with different space sizes are shown in Table 4, where the "No." column represents the index of the test case in multi-task production, numbered from A1 to A13. The best results are highlighted in bold face. It can be seen that

(1) The results (i.e., makespan) of sequential placement are the worst in all test cases;

(2) In most cases, the results of the initial layout are better than those by sequential placement and competitive with those of random placement for 2000 times, indicating the effectiveness of the RBM-based initial placement scheme;

(3) After evolution, all completion times are shorter than those of initial layout, which verifies the effectiveness of the evolution programming mechanism in PLEV;

(4) When the number of machines is fixed, as the number of jobs increases, the makespan without considering the material flow time (makespan is determined only by the processing time of operations) is gradually increased, while the makespan obtained in consideration of the material flow time has an insignificant difference. This is mainly reflected in the fact that the Gantt chart of job scheduling with material flow time (e.g., Fig. 5) will be denser. Since machines can process operations concurrently and makespan is only determined by the completion time of the machine which processes the last operation, when the number of machines is constant, the increase of jobs within a certain range has no significant effect on the completion time of the last operation.

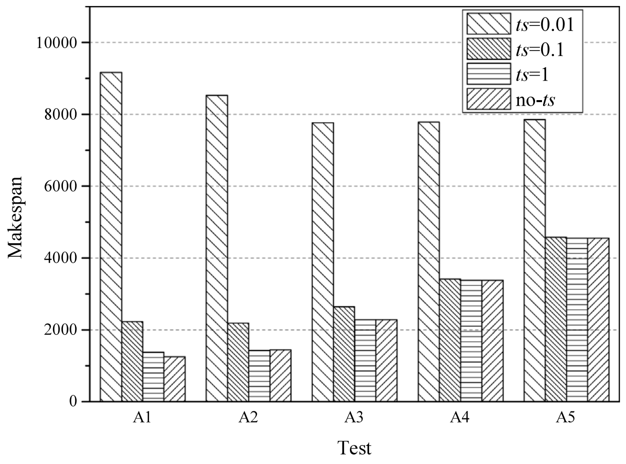


Fig. 9. The effect of different ts on completion time, no-ts means that the time of material flow is not considered.

5.2. Effect of material flow speed on makespan

In this section, the effect of different material flow speeds ts on completion time is investigated. The results are shown in Fig. 9. Note that the processing time of the operation is distributed in [1,50] as in Section 5.1. It can be seen from Fig. 9 that when the flow speed is 0.01, the material flow time has a significant impact on completion time of the operations. As the flow speed increases, the influence of the material flow time on the completion time decreases. When the flow speed is 1, the result of the final completion time is similar to the result of no-ts.

Therefore, when the flow time of the material is relatively large compared to the processing time of jobs, it is of great significance to reduce makespan by optimizing production layout.

5.3. Batch production

In Section 5.1, whether an operation needs to be processed is determined by ρ, so that each job may use different machines with others. In this section, some batch production tests are used to verify the performance of PLEV. Some homogeneous jobs with the same operations that use the same machines are designed for our experiment. While the processing time is affected by multiple factors, it is still randomly generated between [1,50] as in Table 3. Consider the production of 100 machines placed in a workshop size of 10×10. Table 5 lists some test templates, which are used to generate test cases. Each test template including the groups of homogeneous jobs. B1 to B4 represent different test templates for batch production. The operations to be processed in each group are the same (100 at most) and randomly determined by ρ. Take B1 as an example, ρ is used to determine the operations for 400 times (100 times for each group). The other parameters remain the same as in Table 3.

Since the machine numbers used are randomly determined, two cases are generated for each test template, shown in Table 5. The experimental results are shown in Table 6, where the "No." column represents the index of the test case. B1_1 to B4_2 represent the test cases randomly generated based on B1 to B4 in Table 5. For example, B1_1 and B1_2 are generated according to B1. It can be seen that

(1) In all cases, the results obtained by the initial layout and the evolution production layout are better than the results of sequential placement and random placement;

(2) In some cases, the results obtained by the initial layout status are significantly different from the random results, such

Table 4
Experimental results of multi-task production.

No.	Workshop space size ($l \times w$)	m	n	Number of operations	Without material flow	Sequential placement	Random placement (2000)	Initial placement	PLEV ($N_p \times G = 40 \times 50$)			
									Best	Mean	Standard deviation	Running time (s)
A1		20	16	260	565	3426	3343	3629	**3226**	3311.07	56.29	0.832
A2		20	20	321	634	3778	3342	3293	**3081**	3082.57	8.58	1.230
A3	5 × 5	20	40	637	987	3321	2993	3055	**2818**	2890.93	32.66	4.609
A4		20	60	964	1402	3393	3041	3262	**2827**	2883.90	27.66	10.866
A5		20	80	1279	1871	3413	3155	3278	**3044**	3081.47	24.88	20.428
A6		50	40	1609	1253	10585	9893	9449	**8892**	9166.50	89.73	22.024
A7		50	50	1987	1444	9814	9247	8940	**8321**	8526.00	94.91	32.219
A8	8 × 8	50	100	4003	2285	8740	8129	7944	**7665**	7766.67	47.23	136.692
A9		50	150	5966	3381	8271	8042	8114	**7694**	7781.93	44.54	336.481
A10		50	200	7965	4554	9584	8364	8226	**7743**	7860.53	60.69	660.559
A11		100	80	6414	2402	21829	19059	20113	**18413**	18765.50	169.21	283.883
A12	10 × 10	100	100	8032	2627	19507	18073	18784	**17472**	17785.50	121.71	454.825
A13		100	200	16009	4701	19870	15800	16736	**15586**	15720.00	77.31	2159.040

Table 5
Group of homogeneous jobs.

B1	[1, 60]	[61, 80]	[81, 90]	[91, 100]
B2	[1, 70]	[70, 80]	[81, 90]	[91, 100]
B3	[1, 80]	[81, 90]	[91, 100]	
B4	[1, 90]	[90, 100]		

as B1_2, B2_2, B3_1, B3_2, and B4_2. This may be due to the fact that batch production leads to greater relevance and importance of some machines. Placing these machines in close positions can effectively reduce the material flow time, making the initial layout scheme more effective.

5.4. Comparison of different evolution operators

In order to investigate the influence of different evolution operators on the performance of the proposed algorithm, some experiments based on some combinations of evolution operators are carried out in this section. The operators are defined as follows.

A*: Put only one copy of the initial solution into the population.

B1*: Fill the N_p-1 remaining with either random individuals.

B2*: with copies of the initial solution modified by randomly removing half of the machines and then inserting them again at random locations.

C*: Instead of swapping only two machines, apply the operator as follows: C(1): Swap two jobs. C(2): Draw a random number R from [0,1). If R<0.5, go back to C(1). Otherwise, mutation is finished.

These operators are integrated into the PLEV algorithm. The mean values of makespan in some test cases are listed in Table 7, where 'PLEV+A*' means using A* instead of the corresponding operator in the original method.

It can seen that using other steps to replace the original step performs worse. This is because a relatively effective layout has been obtained through the proposed initial production layout scheme. If only one copy of the initial solution is kept and the other N_p-1 individuals are initialized in a random way, the evolution of the N_p-1 individuals will be very slow, causing the algorithm to fail to converge quickly. This can be confirmed from the fact that the algorithms with B2* perform better than the algorithms with B1* because the degree of randomness in B2* is less. However, the algorithms with B2* still perform worse than the original PLEV because they also do not make full use of the good initial layout. Therefore, although B1* and B2* can expand the population diversity, they have a loss in convergence.

In addition, it can be seen that PLEV+B2*+C* tends to perform better than PLEV+B2* in some cases. This means that only performing step C* may improve the performance of the original algorithm, because it not only makes full use of the initial layout, but also expands the diversity reasonably. Therefore, another experiment of PLEV+C* is also conducted. The results are shown in Table 8. The best result, mean, and standard deviation are compared.

It can be seen that PLEV+C* performs better on the 'Best' index, but performs worse on the 'Mean' index. This is because it expands the degree of mutation and is more likely to explore better solutions, but the stability will be reduced. Therefore, these two approaches have their own advantages and disadvantages. Considering the mean value is more important than the best value, the original evolution operators are adopted in this paper.

5.5. Parameters analysis of evolutionary programming

The parameters of evolutionary programming include the population size N_p and the maximum generation number G. In this section, A11, A12, and B3_2 are taken as examples to investigate the impact of N_p and G on the PLEV performance.

N_p is set from 10 to 50 with a step length of 10 and G is set from 10 to 90 with a step length of 20. The results are plotted in Fig. 10. The tendency of the curves indicates that the solution quality improves as N_p or G increases. However, large N_p or G value will increase running time consumption. Since the results are not significantly improved when $N_p \geq 40$ or $G \geq 50$, this study sets $N_p = 40$ and $G = 50$ to make a tradeoff between solution quality and running time.

6. Conclusion

In this paper, a production layout problem oriented to OSSMFT is modeled as a two-stage optimization problem. The first stage is to determine the initial production layout, and the second stage is the evolution of production layout. A novel framework is established to solve it. Based on the production layout evolution framework, an initial production layout scheme based on RBM is proposed in the first stage, which utilizes machine-usage information to provide a feasible and reasonable layout. In the stage of production layout evolution, an EFT-based scheduling rule is adopted to evaluate the status of production layout, and an evolutionary programming mechanism is developed to enable production layout to evolve toward a shorter completion time.

Numerical experiments are designed to verify the performance of the proposed PLEV framework, including multi-task production data generated completely randomly and batch production data.

Table 6
Experimental results of batch production.

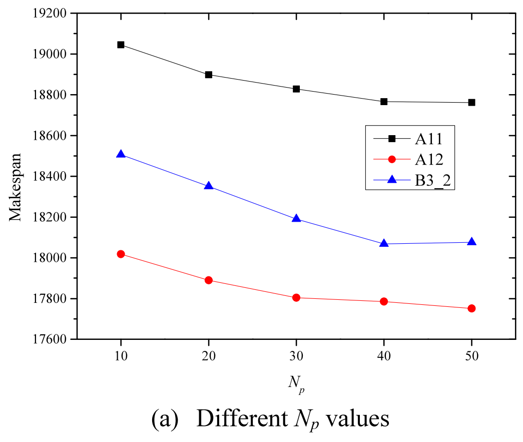
No.	Without material flow	Sequential placement	Random placement (2000)	Initial placement	PLEV ($N_p \times G = 40 \times 50$)			
					Best	Mean	Standard deviation	Running time (s)
B1_1	3005	27908	20882	20206	**17770**	18159.07	221.39	414.865
B1_2	3020	19794	17853	16860	**15935**	16093.17	102.73	424.542
B2_1	2913	21588	19676	19603	**16755**	17468.23	244.45	473.788
B2_2	2881	26225	22160	18952	**18593**	18701.70	125.58	395.001
B3_1	2832	29089	23526	21733	**18385**	19028.93	266.13	417.762
B3_2	2953	23529	21198	20350	**17482**	18068.43	257.12	455.589
B4_1	3054	21418	17272	16797	**15379**	15683.70	117.58	471.252
B4_2	2952	30313	24278	18242	**16135**	16362.40	209.97	392.107

Table 7
Comparison results between PLEV and other algorithms.

No.	PLEV	PLEV+A*+B1*	PLEV+A*+B2*	PLEV+A*+B1*+C*	PLEV+A*+B2*+C*
A1	**3311.07**	3428.47	3384.43	3437.43	3372.33
A2	**3082.57**	3205.77	3173.00	3236.33	3193.60
A3	**2890.93**	2972.53	2968.20	2988.27	2945.27
A4	**2883.90**	3007.10	2977.03	3020.63	2972.90
A5	**3081.47**	3130.00	3115.37	3128.77	3097.80
A8	**7766.67**	7887.53	7898.00	7920.90	7923.47
A12	**17785.50**	17867.60	17801.20	17882.80	17798.20

Table 8
Comparison results between PLEV and PLEV+C*.

No.	PLEV			PLEV+C*		
	Best	Mean	Standard deviation	Best	Mean	Standard deviation
A1	3226	3311.07	56.29	**3126**	**3306.00**	67.81
A2	3081	3082.57	8.58	**2974**	**3068.77**	35.00
A3	2818	**2890.93**	32.66	**2812**	2904.87	34.71
A4	2827	**2883.90**	27.66	**2793**	2905.70	41.79
A5	3044	**3081.47**	24.88	**2993**	3085.33	32.55
A6	8892	9166.50	89.73	**8592**	**9141.60**	139.77
A7	8321	**8526.00**	94.91	**8312**	8547.87	92.66
A8	**7665**	**7766.67**	47.23	7703	7777.03	48.35
A9	7694	**7781.93**	44.54	**7669**	7782.10	61.10
A10	**7743**	**7860.53**	60.69	7756	7867.17	67.23
A11	**18413**	**18765.50**	169.21	18492	18854.40	193.61
A12	17472	17785.50	121.71	**17154**	**17756.40**	187.17
A13	15586	15720.00	77.31	**15511**	**15696.30**	85.94
B1_1	17770	**18159.07**	221.39	**17488**	18248.00	275.92
B1_2	15935	**16093.17**	102.73	**15904**	16141.20	119.92
B2_1	**16755**	**17468.23**	244.45	16864	17505.80	250.04
B2_2	18593	**18701.70**	125.58	**18473**	18803.70	138.28
B3_1	18385	**19028.93**	266.13	**18281**	19191.70	326.99
B3_2	17482	**18068.43**	257.12	**17284**	18259.40	354.13
B4_1	**15379**	**15683.70**	117.58	15487	15763.20	105.45
B4_2	16135	**16362.40**	209.97	**16063**	16504.40	310.96



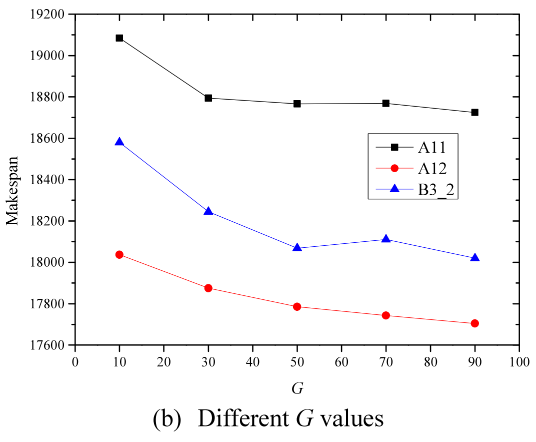


Fig. 10. Influence of the parameters N_p and G on PLEV. (a) Different N_p values. (b) Different G values.

Moreover, the applicable scenarios of PLEV are investigated by the study of transportation speed of material. The experimental results show that PLEV can obtain better layout schemes within acceptable time, especially for extensive homogeneous jobs in batch production compared to random machine layout.

Through the proposed PLEV framework, the multi-task processing oriented production layout problem studied in this paper is systematically analyzed and solved. The experiments further verify the feasibility and effectiveness of the proposed framework. As a combinatorial optimization problem with two-stage planning and scheduling, the multi-task processing oriented production layout problem is complex and difficult. However, through the PLEV framework, the complexity of solving the production layout problem can be significantly reduced, which is verified by the running time of experiments. It means that PLEV can be adopted in larger-scale workshop layout scenarios, and can alleviate the contradiction between running time and problem scale. Such characteristics can well fit the large-scale mixed-line production in the manufacturing industry, which requires factories to dynamically adjust the production layout for different order combinations to shorten the product delivery cycle.

The research framework is extensible. In the future, more methods can be proposed to further optimize production layouts based on this evolutionary framework. In addition, after the optimized production layout is obtained by the proposed PLEV framework, some meta-heuristic approaches can be used to further optimization the open shop scheduling stage, so as to obtain a better makespan. Furthermore, the design idea of PLEV is to correlate optimization problems in different stages so as to coordinate optimization through evolutionary mechanisms. It will also provide a reference for solving other multi-stage combinatorial optimization problems.

CRediT authorship contribution statement

Zhao-Hui Sun: Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision. **Di Liang:** Formal analysis, Validation, Writing - original draft, Writing - review & editing, Supervision. **Zilong Zhuang:** Visualization. **Liang Chen:** Project administration. **Xinguo Ming:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The author would like to thank SJTU, China Innovation Center of Producer Service Development, Shanghai Research Center for industrial Informatics, Shanghai Key Lab of Advanced Manufacturing Environment, National Natural Science Foundation of China (Grant No. 71632008) and Major Project for Aero engines and Gas turbines (Grant No. 2017-I-0007-0008, Grant No. 2017-I-0011-0012) for the funding support to this research.

References

[1] F. Ahmadizar, M.H. Farahani, A novel hybrid genetic algorithm for the open shop scheduling problem, Int. J. Adv. Manuf. Technol. 62 (2012) 775–787, http://dx.doi.org/10.1007/s00170-011-3825-1.
[2] Y. Wang, X. Li, R. Ruiz, S. Sui, An iterated greedy heuristic for mixed no-wait flowshop problems, IEEE Trans. Cybern. 48 (2018) 1553–1566, http://dx.doi.org/10.1109/TCYB.2017.2707067.
[3] Y. Yuan, H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms, IEEE Trans. Autom. Sci. Eng. 12 (2015) 336–353, http://dx.doi.org/10.1109/TASE.2013.2274517.
[4] S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming, IEEE Trans. Evol. Comput. 18 (2014) 193–208, http://dx.doi.org/10.1109/TEVC.2013.2248159.
[5] Y. Han, D. Gong, Y. Jin, Q. Pan, Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns, IEEE Trans. Cybern. 49 (2019) 184–197, http://dx.doi.org/10.1109/TCYB.2017.2771213.
[6] H. Panahi, R.T. Moghaddam, Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization, Expert Syst. Appl. 38 (2011) 2817–2822, http://dx.doi.org/10.1016/j.eswa.2010.08.073.
[7] B. Naderi, S.M.T. Fatemi Ghomi, M. Aminnayeri, M. Zandieh, A contribution and new heuristics for open shop scheduling, Comput. Oper. Res. 37 (2010) 213–221, http://dx.doi.org/10.1016/j.cor.2009.04.010.
[8] D. Lei, M. Li, L. Wang, A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold, IEEE Trans. Cybern. 49 (2019) 1097–1109, http://dx.doi.org/10.1109/TCYB.2018.2796119.
[9] K. Gao, F. Yang, M.C. Zhou, Q. Pan, P.N. Suganthan, Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm, IEEE Trans. Cybern. 49 (2019) 1944–1955, http://dx.doi.org/10.1109/TCYB.2018.2817240.
[10] S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Automatic programming via iterated local search for dynamic job shop scheduling, IEEE Trans. Cybern. 45 (2015) 1–14, http://dx.doi.org/10.1109/TCYB.2014.2317488.
[11] Q.-K. Pan, L. Wang, K. Mao, J.-H. Zhao, M. Zhang, An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process, IEEE Trans. Autom. Sci. Eng. 10 (2013) 307–322, http://dx.doi.org/10.1109/TASE.2012.2204874.
[12] S. Noori-Darvish, I. Mahdavi, N. Mahdavi-Amiri, A bi-objective possibilistic programming model for open shop scheduling problems with sequence-dependent setup times, fuzzy processing times, and fuzzy due-dates, Appl. Soft Comput. 12 (2012) 1399–1416, http://dx.doi.org/10.1016/j.asoc.2011.11.019.
[13] Y.-M. Huang, J.-C. Lin, A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems, Expert Syst. Appl. 38 (2011) 5438–5447, http://dx.doi.org/10.1016/j.eswa.2010.10.010.
[14] D. Bai, L. Tang, Open shop scheduling problem to minimize makespan with release dates, Appl. Math. Model. 37 (2013) 2008–2015, http://dx.doi.org/10.1016/j.apm.2012.04.037.
[15] P. Pongchairerks, V. Kachitvichyanukul, A two-level particle swarm optimisation algorithm for open-shop scheduling problem, Int. J. Comput. Sci. Math. 7 (2016) 575–585, http://dx.doi.org/10.1504/IJCSM.2016.081693.
[16] D. Bai, Z.-H. Zhang, Q. Zhang, Flexible open shop scheduling problem to minimize makespan, Comput. Oper. Res. 67 (2016) 207–215, http://dx.doi.org/10.1016/j.cor.2015.10.012.
[17] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, M. Sterna, J. Weglarz, Open shop scheduling, in: Handbook on Scheduling. International Handbooks on Information Systems, 2019, pp. 321–343, http://dx.doi.org/10.1007/978-3-319-99849-7_9.
[18] D. Bai, L. Tang, Open shop scheduling problem to minimize makespan with release dates, Appl. Math. Model. 37 (2013) 2008–2015, http://dx.doi.org/10.1016/j.apm.2012.04.037.
[19] M. Sheikhalishahi, N. Eskandari, A. Mashayekhi, A. Azadeh, Multi-objective open shop scheduling by considering human error and preventive maintenance, Appl. Math. Model. 67 (2019) 573–587, http://dx.doi.org/10.1016/j.apm.2018.11.015.
[20] G. Mejía, F. Yuraszeck, A self-tuning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel/setup times, European J. Oper. Res. (2020) http://dx.doi.org/10.1016/j.ejor.2020.02.010.
[21] A. Drira, H. Pierreval, S. Hajri-Gabouj, Facility layout problems: A survey, Annu. Rev. Control. 31 (2007) 255–267, http://dx.doi.org/10.1016/j.arcontrol.2007.04.001.
[22] A. Taghavi, A. Murat, A heuristic procedure for the integrated facility layout design and flow assignment problem, Comput. Ind. Eng. 61 (2011) 55–63, http://dx.doi.org/10.1016/j.cie.2011.02.011.
[23] X. Zuo, S. Gao, M.C. Zhou, X. Yang, X. Zhao, A three-stage approach to a multirow parallel machine layout problem, IEEE Trans. Autom. Sci. Eng. 16 (2019) 433–447, http://dx.doi.org/10.1109/TASE.2018.2866377.
[24] A. Kheirkhah, H. Navidi, M.M. Bidgoli, Dynamic facility layout problem: a new bilevel formulation and some metaheuristic solution methods, IEEE Trans. Eng. Manage. 62 (2015) 396–410, http://dx.doi.org/10.1109/TEM.2015.2437195.
[25] X.Q. Zuo, C.C. Murray, A.E. Smith, Solving an extended double row layout problem using multiobjective tabu search and linear programming, IEEE Trans. Autom. Sci. Eng. 11 (2014) 1122–1132, http://dx.doi.org/10.1109/TASE.2014.2304471.

[26] X. Zuo, B. Li, X. Huang, M.C. Zhou, C. Cheng, X. Zhao, Z. Liu, Optimizing hospital emergency department layout via multiobjective tabu search, IEEE Trans. Autom. Sci. Eng. 16 (2019) 1137–1147, http://dx.doi.org/10.1109/TASE.2018.2873098.

[27] A. Baykasoglu, T. Dereli, I. Sabuncu, An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems, Omega 34 (2006) 385–396, http://dx.doi.org/10.1016/j.omega.2004.12.001.

[28] F. Jolai, R. Tavakkoli-Moghaddam, M. Taghipour, A multiobjective particle swarm optimisation algorithm for unequal sized dynamic facility layout problem with pickup/drop-off locations, Int. J. Prod. Res. 50 (2012) 4279–4293, http://dx.doi.org/10.1080/00207543.2011.613863.

[29] M.A. Mohammed, R.A. Hasan, Particle swarm optimization for facility layout problems FLP–A comprehensive study, in: 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, Cluj-Napoca, Romania, 2017, http://dx.doi.org/10.1109/ICCP.2017.8116988.

[30] M.F. Anjos, M.V.C. Vieira, Mathematical optimization approaches for facility layout problems: The state-of-the-art and future research directions, European J. Oper. Res. 261 (2017) 1–16, http://dx.doi.org/10.1016/j.ejor.2017.01.049.

[31] R. Sahin, A simulated annealing algorithm for solving the bi-objective facility layout problem, Expert Syst. Appl. 4 (2011) 4460–4465, http://dx.doi.org/10.1016/j.eswa.2010.09.117.

[32] S. Safarzadeh, H. Koosha, Solving an extended multi-row facility layout problem with fuzzy clearances using GA, Appl. Soft Comput. 61 (2017) 819–831, http://dx.doi.org/10.1016/j.asoc.2017.09.003.

[33] H. Hosseini-Nasab, L.A. Emami, A hybrid particle swarm optimization for dynamic facility layout problem, Int. J. Prod. Res. 51 (2013) 4325–4335, http://dx.doi.org/10.1080/00207543.2013.774486.

[34] A. Sadrzadeh, A genetic algorithm with the heuristic procedure to solve the multi-line layout problem, Comput. Ind. Eng. 62 (2012) 1055–1064, http://dx.doi.org/10.1016/j.cie.2011.12.033.

[35] A.R. Hosseinabadi, J. Vahidi, B. Saemi, A.K. Sangaiah, M. Elhoseny, Extended genetic algorithm for solving open-shop scheduling problem, Soft Comput. 23 (2019) 5099–5116, http://dx.doi.org/10.1007/s00500-018-3177-y.

[36] H. Loedding, A. Piontek, The surprising effectiveness of earliest operation due-date sequencing, Prod. Plan. Control 28 (2017) 459–471, http://dx.doi.org/10.1080/09537287.2017.1302616.

[37] O. Rose, The Shortest Processing Time First (SPTF) dispatch rule and some variants in semiconductor manufacturing, in: Winter Simulation Conference (WSC01), IEEE, Arlington, VA, 2001, pp. 1220–1224.

[38] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, in: Stanford Digital Library Technologies Project, 1998.

[39] H. Ishii, R. Tempo, Distributed randomized algorithms for the pagerank computation, IEEE Trans. Autom. Control. 55 (2010) 1987–2002, http://dx.doi.org/10.1109/TAC.2010.2042984.

[40] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102, http://dx.doi.org/10.1109/4235.771163.

[41] R.G. Regis, Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions, IEEE Trans. Evol. Comput. 18 (2014) 326–347, http://dx.doi.org/10.1109/TEVC.2013.2262111.

[42] M.A. Contreras-Cruz, V. Ayala-Ramirez, U.H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Appl. Soft Comput. 30 (2015) 319–328, http://dx.doi.org/10.1016/j.asoc.2015.01.067.

[43] L. Wang, D.Z. Zheng, A modified evolutionary programming for flow shop scheduling, Int. J. Adv. Manuf. Technol. 22 (2003) 522–527, http://dx.doi.org/10.1007/s00170-002-1477-x.

[44] D.K. Khatod, V. Pant, J. Sharma, Evolutionary programming based optimal placement of renewable distributed generators, IEEE Trans. Power Syst. 28 (2013) 683–695, http://dx.doi.org/10.1109/TPWRS.2012.2211044.