Computers & Operations Research 161 (2024) 106432

Contents lists available at ScienceDirect

Computers and Operations Research

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

# Automatic design of constructive heuristics for a reconfigurable distributed flowshop group scheduling problem

Biao Zhang [a], Lei-lei Meng [a], Chao Lu [b,*], Yu-yan Han [a], Hong-yan Sang [a]

[a] School of Computer Science, Liaocheng University, Liaocheng 25200, PR China
[b] School of Computer Science, China University of Geosciences, Wuhan 430074, PR China

ARTICLE INFO

*Keywords:*
Constructive heuristic
Automatic algorithm design
Distributed flowshop scheduling
Reconfigurability
Group scheduling

ABSTRACT

This study addresses a reconfigurable distributed flowshop group scheduling problem (RDFGSP), the characteristics of which lie in the reconfigurability of the flowlines, and the families with grouped jobs. Given its NP-hard property, we are committed to developing constructive heuristics to meet real-time requirements. By combining different algorithm components, a large number of constructive heuristics can be obtained, rendering the identification of the best heuristic through artificial experimental design quite difficult. To take full advantage of the problem domain knowledge, the iterated F-Race (I/F-Race), an automatic algorithm design (AAD) methodology, is employed to automatically conceive constructive heuristics with minimum human intervention. Initially, a general and configurable *meta*-algorithm is developed by considering the problem-specific characteristics, which integrates the configurable routing rule, sequencing rule, dispatching rule, and non-delay factor. Subsequently, by using the AAD, the *meta*-algorithm can be instantiated to a complete constructive heuristic, which can generate active, non-delay, or hybrid schedules. In the experimental study, compared with the full factorial design, the AAD can conceive a much more effective automated heuristic by tuning a much smaller number of configurations. Furthermore, the solution accuracy and efficiency of the generated heuristic are validated in solving small-scale problems by comparison with the commercial solver and other heuristics. The generated heuristic substantiates an advancement of approximately 28-fold in contrast to the best compared heuristic at a very small cost when solving large-scale problems.

## 1. Introduction

In the face of fierce market competition and the production requirements of multiple users, multitasking and many varieties, manufacturing enterprises face intense pressure to improve their competitiveness and customer services. Therefore, they have begun to explore advanced production methods to achieve their goals. The cellular manufacturing system (CMS) is one such advanced production organization that is gaining acceptance among manufacturing enterprises (Pan et al., 2022; Neufeld et al., 2020; Huang and Zhou, 2022). In the case of a surface mount technology (SMT) workshop, a classical CMS for processing the printed circuit boards (PCBs), two essential features stand out: mounting components on both sides and families with grouped jobs. It is known that production scheduling plays a crucial role in enhancing the core competitiveness of manufacturing enterprises (Khare and Agrawal, 2021; Lu et al., 2022; Neufeld et al., 2016). However, the two features are not well studied in the current scheduling literature. Thus, motivated by the two features in the SMT workshop, this paper studies a reconfigurable distributed flowshop scheduling problem (RDFGSP) that takes the flowlines as independent workshops and has the objective of minimizing the makespan (Zhang et al., 2023). The addressed problem involves determining how to allocate families to the cells at the flows, and how to sequence the families and the jobs accommodated in them. Thus, it is much more complex than the distributed flowshop scheduling problem (DFSP) and is NP-hard. Developing powerful optimization approaches has significant academic and practical value.

To address the RDFGSP, a mixed integer linear programming (MILP) model is introduced. This MILP can be run on the mathematical programming solver, and solve small-scale problems to optimality. Nevertheless, owing to the NP-hard nature of the problem, the MILP model faces significant challenges as the problem scale increases. It struggles to yield an optimal solution within a reasonable time and may even fail to discover the optimal solution altogether. Consequently, we advocate for

* Corresponding author.
*E-mail address:* luchao@cug.edu.cn (C. Lu).

https://doi.org/10.1016/j.cor.2023.106432
Received 14 August 2022; Received in revised form 13 August 2023; Accepted 14 September 2023
Available online 18 September 2023

the adoption of a heuristic approach, which can offer expedited problem-solving capabilities to fulfill real-time requirements. Within the confines of this study, our focus rests upon constructive heuristics, which diligently construct a comprehensive schedule in an incremental manner, commencing from a blank canvas.

On the one hand, the process of designing constructive heuristics often relies on expert knowledge and consumes a considerable amount of time. On the other hand, constructive heuristics are typically tailored to specific situations, which can pose a challenge since real-world scenarios are subject to constant change. This dynamic nature may render existing heuristics ineffective or even infeasible. To overcome these challenges, the automated algorithm design (AAD) methodology (López-Ibáñez et al., 2016; Bezerra et al., 2020) can be introduced, which enables the automatic design of heuristics with minimal human intervention. In this paper, the iterated F-Race (I/F-Race), a machine learning method that was first proposed for model selection, is implemented as the AAD. I/F-Race has garnered substantial attention and undergone extensive study in recent years (Zhao et al., 2023; Huang et al., 2019; Pagnozzi and Stützle, 2019; Zhang et al., 2022a). In this paper, I/F-Race is employed to conceive a constructive heuristic automatically for solving the RDFGSP in the framework of a developed *meta*-algorithm. In the experimental study, compared with the full factorial design (FFD), the AAD is capable of devising a significantly more effective automated heuristic while tuning a substantially smaller number of configurations. Furthermore, the solution accuracy and efficiency of the automated heuristic are verified in solving small-scale problems by comparison with the commercial solver and other heuristics. Additionally, the effectiveness of the automated heuristic is demonstrated at a very small cost in time when solving large-scale problems. The main contributions of the paper are twofold: (1) A *meta*-algorithm for the constructive heuristics for solving the RDFGSP is proposed. (2) An AAD method is designed to conceive the constructive heuristic automatically.

The organization of the paper is as follows. In Section 2, we present a brief literature review. Section 3 describes the RDFGSP, and the MILP model is presented. The heuristic configuration is defined in Section 4. Section 5 describes the AAD, and Section 6 discusses the experimental study. Finally, the conclusions and future studies are presented in Section 7.

## 2. Literature review

To the best of our knowledge, the literature on the RDFGSP addressed in this paper is very limited. Nevertheless, by exploring related research on flowshop variants, valuable insights can be gleaned. Hence, we first undertake close studies for the flowshop variants, including the flowshop scheduling problem (FSP), distributed flowshop scheduling problem (DFSP), and distributed flowshop group scheduling problem (DFGSP). We then shift out focus to the study of constructive heuristics for solving FSP variants. Subsequently, the studies on I/F-Race are reviewed. Finally, we provide summaries of the reviews and pose research questions for further investigation.

### 2.1. Review of the studies on FSP variants

The classical FSP involves determining the scheduling sequence of jobs that are to be processed on a flowline composed of multiple and consecutive machines. Initially formulated as a two-machine problem, the FSP has been studied for almost 70 years and has been proven to be NP-hard in the strong sense (Nawaz et al., 1983). The DFSP was first formulated by Naderi and Ruiz (Naderi and Ruiz, 2010) as an extension of the FSP, introducing a multi-factory environment. It requires to determine the assignment of jobs to multiple factories and the scheduling sequence of jobs within each factory. By considering the families consisting of the jobs with underlying sameness, Pan et al. (Pan et al., 2022) further introduced a novel scheduling problem known as the DFGSP. The DFGSP involves determining how to assign families to factories, how to sequence the families in each factory, and how to sequence the jobs in each family.

Owing to the significance both in theory and applications of the FSP variants, various approaches of finding optimal or near-optimal solutions have been developed. These approaches can generally be classified into two classes: exact and approximate techniques. Exact techniques typically employ enumeration methods to obtain an optimal solution. For instance, Bai et al. (Bai et al., 2022) introduced a branch and bound algorithm for the FSP. Hamzadayı (Hamzadayı, 2020) proposed an effective benders decomposition algorithm for solving the DFSP. Pan et al. (Pan et al., 2022) formulated the DFGSP as an MILP model, enabling the utilization of mathematical programming optimizers to obtain optimal solutions. However, in most cases, these techniques require a prohibitive amount of computation even for small-scale problems. Approximate techniques offer a more efficient and cost-effective approach for obtaining near-optimal solutions. These techniques can broadly be categorized as either constructive heuristics or improvement heuristics. Constructive heuristics are capable of building promising solutions by leveraging problem-specific characteristics, operating within polynomial time. Furthermore, it is worth noting that the performance of the improvement heuristics, such as stochastic local search method for the FSP (Pagnozzi and Stützle, 2019), iterated greedy algorithm for the DFSP (Khare and Agrawal, 2021), cooperative co-evolutionary algorithm for the DFGSP (Pan et al., 2022), can be significantly enhanced by incorporating promising initial solutions generated by constructive heuristics. In the realm of FSP research, the design of well-crafted constructive heuristics assumes a pivotal role. Hence, this study specifically focuses on constructive heuristics, which are reviewed in the following subsection.

### 2.2. Review of the constructive heuristics for FSP variants

A popular and simple heuristic, namely NEH, for minimizing makespan in the general FSP was presented by Nawaz et al. in 1983 (Nawaz et al., 1983), which outperformed other algorithms developed earlier, such as the heuristics of Palmer (Palmer, 1965), Gupta (Gupta, 1971), and Campbell-Dudek-Smith (Campbell et al., 1970). The NEH heuristic consists of two phases: the sorting phase and the insertion phase. Since 1983, many researchers have tried to improve NEH. Dong et al. (Dong et al., 2008) developed an improved NEH-based heuristic based on the idea of balancing the utilization among all machines. Framiman et al. (Framinan et al., 2003) and Abedinnia et al. (Abedinnia et al., 2016) summarized the room to improve the NEH heuristic by modifying the sorting phase and insertion phase. Framinan et al. (Framinan et al., 2003) generalized the NEH heuristic to minimize makespan, idletime and flowtime, and proposed heuristics for these three objectives. Framiman et al. (Framinan et al., 2004) also established a common framework to categorize heuristic algorithms for FSP. The framework divided the existing heuristics into three phases: (a) index development, (b) solution construction, and (c) solution improvement. Framinan and Leisten (Framinan and Leisten, 2003) extended the NEH heuristic to minimize the flowtime by exploiting the partial schedules in more detail. Laha and Sarin (Laha and Sarin, 2009) improved the heuristic proposed in (Framinan et al., 2004) to minimize the flowtime, which allowed the iteration of the insertion step rather than only performing the pairwise interchanges. Abedinnia et al. (Abedinnia et al., 2016) extended the NEH heuristic and improved the heuristic proposed in (Laha and Sarin, 2009) by introducing new priority rules for weighting jobs and decision criteria for the insertion phase.

Regarding the DFSP, it involves solving the assignment of jobs to the factories and scheduling sequence of jobs in the families. Naderi and Ruiz (Naderi and Ruiz, 2010) proposed 14 constructive heuristics and variable neighborhood descent methods. From the comprehensive computational results, they found that two NEH-based heuristics, namely NEH1 and NEH2, integrating two family assignment rules perform best among the 14 constructive heuristics. Ruiz et al. (Ruiz

et al., 2019) extended the NEH2 proposed in (Naderi and Ruiz, 2010) by adjusting the insertion phase, resulting in a novel constructive heuristic, namely NEH2_en. Additionally, based on NEH2, Shao et al. (Shao et al., 2021) proposed three distributed NEH variants, namely DNEH1, DNEH2, and DNEH3. DNEH1 aims to obtain a lower makespan and balance the workload of factories. DNEH2 is an extension of DNEH1, which integrates a greedy insertion step. DNEH2 makes more efforts on the factories with the minimum makespan and the maximum makespan. Chen et al. (Chen et al., 2021) extended several NEH-based heuristics when dealing with the blocking flowshop scheduling problem in a distributed environment, and they showed that NEH2EE performs best among all the algorithms. Pan et al. (Pan et al., 2019) proposed a novel constructive heuristic that combined the NEH heuristic with the LR heuristic (Liu and Reeves, 2001). In the heuristic, a partial job sequence is first constructed using the LR, and then the remaining jobs are arranged by using the NEH.

As regards the research of constructive heuristics for solving DFSP with the consideration of the group feature, there is only one paper in the literature. Pan et al. (Pan et al., 2022) studied a distributed flowshop group scheduling problem (DFGSP) to arrange a variety of jobs subject to group constraints at a number of identical manufacturing cells. Each cell has a flowshop structure. To solve the problem, they proposed a heuristic, namely LPT-GA-NEH, to construct an initial solution. In this heuristic, the job sequence is generated according to the largest processing time (LPT) rule, then the family sequence is determined by using a greedy assignment (GA) rule, and finally the NEH is used to improve the family sequences.

### 2.3. Review of the studies on I/F-Race

Regarding the AAD, related to the terms automated algorithm configuration, automated parameter tuning, and hyper-heuristic (Zhao et al., 2023), Huang et al. (Huang et al., 2019) classified the methods into three categories from the perspective of tuner's structure and composition: 1) simple generate-evaluate methods; 2) iterative generate-evaluate methods; 3) high-level generate-evaluate methods. The second category is the most fruitful class of approaches for tuning problems. Various methods, including I/F-Race, ParamILS, CALIBRA, REVAC, and so forth, fall in this category (Huang et al., 2019). Among them, I/F-Race is one of state-of-the-art methods, which has showed its effectiveness in tuning various problems, and has the following advantages. First, I/F-Race can handle both the numerical and categorical parameters, thus exhibiting a comprehensive utility. Second, I/F-Race uses the statistical model to make it interpretable and easy to understand, which is important for researchers and practitioners to develop the automated design techniques. I/F-Race is reviewed as follows.

Birattari et al. (Birattari and Kacprzyk, 2009; Birattari et al., 2002) proposed an F-Race method, which can optimize both the categorical and numerical parameters. It is inspired by the Hoeffding Race used for model selection in machine learning. In the method, the initial candidate configurations are obtained by a full factorial design. However, when there are a large number of parameters or the parameters have a wide range of values, adopting the full factorial design may become impractical and computationally limited. To alleviate this problem, Balaprakash et al. (2017) proposed an iterated application of F-Race, i.e., I/F-Race, which follows the search framework based on a learning model. Since its proposal, I/F-Race has become a competitive AAD method. Liao et al. (Liao et al., 2013; Liao et al., 2015) successfully configured a continuous optimization solver with high performance for solving different test sets. López-Ibáñez et al. (López-Ibáñez et al., 2016) developed a software package, called Irace, in programming language R, where several variants of I/F-Race are included. Since then, Irace has been used in the design of automatic algorithms to solve various engineering optimization problems (Pagnozzi and Stützle, 2019); (Pan et al., 2021; Bistaffa et al., 2019; Nagano et al., 2021). The above studies all focused on single-objective optimization. Bezerra et al. (Bezerra et al., 2020) proposed an automatic multi-objective evolutionary algorithm (MOEA) framework with a multi-objective formula. Based on the I/F-Race, Zhang et al. (Zhang et al., 2022a; Zhang et al., 2022b) conceived the MOEAs to solve the multi-objective hybrid flowshop scheduling problems with consistent and variable sublots.

### 2.4. Summary and research questions

The work in this study aims to solve the RDFGSP with the objective minimizing the makespan using constructive heuristics. The RDFGSP requires dealing with three interdependent decisions: (1) assignment of families to the cells at each flow, (2) scheduling sequence of families at the cells, and (3) scheduling sequence of jobs in the families. Regarding the first decision, which bears resemblance to the assignment of jobs to factories in the DFSP, it is possible to learn factory assignment rules from the existing literature. For the third decision, the NEH heuristic, which has demonstrated success in determining job sequences for both the FSP and DFSP, serves as an inspiration for developing NEH rules tailored to the RDFGSP, while considering problem-specific characteristics. Addressing the second decision, the key characteristic, i.e., reconfigurability, must be addressed. Accordingly, this involves developing rules that dynamically connect the flowlines to complete the production tasks. It is important to note that, thus far, no research paper has specifically addressed reconfigurability using constructive heuristics in a distributed flowshop configuration. For LPT_GA_NEH, although it can address the family sequence and job sequence, it cannot address the decision of cell connection. The other heuristics proposed in the distributed flowshop environment can only deal with the decisions of job sequencing and cell assignment, but they cannot deal with the decisions of family sequencing and cell connection.

Thus, in light of the promising performance of the I/F-Race, it is worthwhile to conceive an effective constructive heuristic by leveraging its strengths for the RDFGSP. This paper endeavors to answer three key research questions.

- How to develop a configurable procedure to construct a schedule for the RDFGSP?
- How to design the schedule rules for the RDFGSP considering the specific characteristics and existing experience?
- How does the I/F-Race automatically conceive the constructive heuristic for the RDFGSP, and to what extent can this heuristic be advanced?

## 3. Problem description

In this section, we first provide the description of the RDFGSP, and illustrate it with an example. Subsequently, the MILP model is formulated.

The RDFGSP is motivated by a surface mount technology (SMT) workshop for processing PCBs. The processing of PCBs can be divided into two flows. As shown in Fig. 1 (Zhang et al., 2023), once a side mounting flow (known as side B) has been accomplished, then another side mounting flow (known as side A) can be implemented. In the SMT workshop of large-scale enterprises, there are usually dozens of flowlines. The SMT workshop, therefore, can be regarded as a CMS, in which the flowline with an advanced degree of automation acts as the cell and can independently undertake a one-sided mounting flow. That is, each flowline can mount the A and B sides. Thus, to complete the process of one PCB, it is necessary to flexibly select and connect two lines into a whole line. The two lines can either be the same or different. In addition, different PCBs have different processing requirements because of their different designs. To reduce the switching time of mounting tools, a family accommodates PCBs that have a similar design, which are processed consecutively with a very small switching time.

In light of the above context, the RDFGSP is characterized as follows. Without loss of generality, a PCB is called a job here. Suppose that in a

CMS, there are some cells formed by several machines. A number of jobs are to be processed through two flows consecutively, and each flow can be undertaken by a cell. The jobs can be grouped into several families according to their similarities. The jobs within the same family must be processed sequentially, and their scheduling sequence can be adjusted. The setups for different families are required, and they are anticipatory and executed before processing. The setups for the different jobs within the same family are ignored. When a job is finished on a machine of a cell, it can be started on the next machine. Moreover, when a job is completed for its flow1, it can be immediately forwarded to its flow 2. The transportation time between the two cells is different. The jobs for flow 2 can only be processed when flow 1 is finished and transported to the cell for flow 2. The other assumptions for the RDFGSP are summarized below.

- All jobs are available at time zero, and preemption and interruption are not considered.
- Machine idle time is allowed, and buffer capacity is infinite between machines.
- Each job goes through all flows one by one, and at a specific flow, it goes through all stages one by one.
- At any time, one machine can at most process one job, and jobs from the same family are continuously processed.

### 3.1. Problem illustration

With the objective of minimizing the makespan, the problem aims to determine the assignment of families to the cells at each flow, the scheduling sequence of families on the cells, and the sequence of jobs in the families. To illustrate the problem, a simple example with four families is introduced, and the four families contain 4, 2, 2 and 3 jobs respectively, and two cells that have three machines. The relevant processing data are given in Table 1, where the processing time of each job in each family, the setup time of each family on each machine at the two flows, and the transfer time between each two cells are given.

This instance is formulated as a MILP introduced in Section 3.2, and run on IBM ILOG CPLEX 12.7.1. An optimal schedule is obtained and drawn in Fig. 2. From Fig. 2, it can be seen that in the optimal schedule, the processes of different families have different routes by connecting different cells. For example, for processing family 1, Cell 1 is connected with Cell 2. Cell 1 is responsible for flow 1 of family 1, while Cell 2 is responsible for flow 2 of family 1. In addition, the scheduling sequence of families at each cell, and the scheduling sequence in each family are all involved. For example, the scheduling sequence of families on cell 1 at flow 1 is (Family 1, Family 4). The scheduling sequence in Family 1 is (Job 2, Job 3, Job 4, and Job 1).

### 3.2. MILP model

**Notations:**

$k$: Flow index.

$|F|$: Number of families.

$f$: Family index.

$F$: Set of families, and $F = \{1, 2, \ldots, f, \ldots, |F|\}$.

$n_f$: Number of jobs in family $f$.

$j$: Job index.

$J_f$: Set of jobs in family $f$, and $J_f = \{1, 2, \ldots, j, \ldots, n_f\}$.

$|C|$: Number of cells.

$C$: Set of cells, and $C = \{1, 2, \ldots, c, \ldots, |C|\}$.

$|M|$: Number of machines in each cell.

$m$: Machine index.

$M$: Set of machines, and $M = \{1, 2, \ldots, m, \ldots, |M|\}$.

$p_{j,f,m,k}$: Processing time on machine $m$ of job $j$ in family $f$ at flow $k$.

$s_{f,m,k}$: Setup time of family $f$ on machine $m$ at flow $k$.

$t_{c,c'}$: Transportation time from cell $c$ to cell $c'$.

$G$: A very large positive value.

**Decision variables:**

$B_{j,f,m,k}$: Beginning time of job $j$ in family $f$ on machine $m$ at flow $k$.

$E_{j,f,m,k}$: Ending time of job $j$ in family $f$ on machine $m$ at flow $k$.

$D_{f,c,k}$: Binary variable that takes the value of 1 when family $f$ is assigned to cell $c$ at flow $k$.

$X_{f,f',c,k,k'}$: Binary variable that takes the value of 1 when flow $k$ of family $f$ is scheduled before flow $k'$ of family $f'$ on cell $c$.

$Y_{j,j',f}$: Binary variable that takes the value of 1 when job $j$ is scheduled before job $j'$ in family $f$.

With the above notations listed in the **Notations**, the MILP model (Zhang et al., 2023) is presented as follows.

**Objective:**

$$Minimize C_{\max} \tag{1}$$

Eq. (1) is to minimize $C_{\max}$, which is defined as the maximum completion time of all the jobs at flow 2.

**Constraints:**

$$C_{\max} \geqslant E_{j,f,|M|,2}, \quad \forall j \in J_f, f \in F \tag{2}$$

Eq. (2) requires that $C_{\max}$ must be larger than the completion time of each job in each family at flow 2.

$$\sum_{c=1}^{|C|} D_{f,c,k} = 1, \quad \forall f \in F, k \in [1,2] \tag{3}$$

Eq. (3) defines that each family at each flow can only be assigned to one cell.

$$B_{j,f,m,1} \geqslant s_{f,m,1}, \quad \forall j \in J_f, f \in F \tag{4}$$

Eq. (4) indicates that the beginning time of each job at flow 1 should be larger than or equal to its setup time.

$$E_{j,f,m,k} - B_{j,f,m,k} = p_{l,g,m,k}, \quad \forall j \in J_f, f \in F, m \in M, k \in \{1,2\} \tag{5}$$

Eq. (5) defines the relationship between $B_{j,f,m,k}$ and $E_{j,f,m,k}$, and indicates that interruption is not allowed in the processing,

$$B_{j,f,m+1,k} - E_{j,f,m,k} \geqslant 0, \quad \forall j \in J_f, f \in F, m \in [1, \ldots, |M|-1], k \in \{1,2\} \tag{6}$$

Eq. (6) represents that the process of the job can be started only after its completion of the previous operation at the same flow.

$$B_{j,f,m,2} - E_{j,f,m,1} - s_{f,m,2} + G \times \left(3 - X_{f,f,c,1,2} - D_{f,c,1} - D_{f,c,2}\right) \geqslant 0, \quad \forall j \in J_f, f \in F, c \in C \tag{7}$$

Eq. (7) guarantees that if the two flows of the same family are assigned to the same cell, the operations on the machines at flow 2 can only be started after the operations on the machines at flow 1 and the setups are completed.

$$B_{j,f,m,k} - E_{j',f',m,k'} - s_{f,m,k} + G \times \left(3 - X_{f',f,c,k',k} - D_{f,c,k} - D_{f',c,k'}\right) \geqslant 0,$$

$$\forall j \in J_f, j' \in J_{f'}, f, f' \in F(f! = f'), c \in C, k, k' \in \{1,2\} \tag{8}$$

Eq. (8) expresses that the jobs in each family on the machines can be started only after the setups and the operations on the machines have been completed if there is any previous family assigned to the same cell.

$$X_{f,f',c,k,k'} + X_{f',f,c,k',k} \leqslant 1, \quad \forall f, f' \in F(f! = f'), c \in C, k', k \in \{1,2\} \tag{9}$$

Eq. (9) guarantees that $X_{f,f',c,k,k'}$ and $X_{f',f,c,k',k}$ cannot take the value of 1 at the same time.

$$X_{f,f',c,k,k'} + X_{f',f,c,k',k} \leqslant D_{f,c,k} + D_{f',c,k'}, \quad \forall f, f' \in F(f! = f'), c \in C, k', k \in \{1,2\} \tag{10}$$

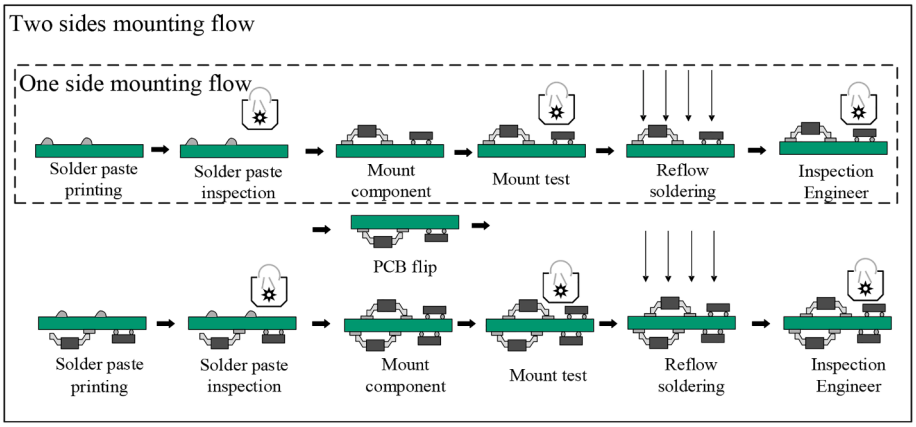



**Fig. 1.** The two flows in SMT processing.

$$D_{f,c,k}+D_{f',c,k'}-1\leqslant X_{f,f',c,k,k'}+X_{f',f,c,k',k},\qquad \forall f,f'\in F(f!=f'),c\in C,k',k\in\{1,2\} \tag{11}$$

Eq. (10) and Eq. (11) together establish the relationship between two variables $D_{f,c,k}$ and $X_{f,f',c,k,k'}$.

$$X_{f,f,c,2,1}==0,\forall f\in G,c\in C \tag{12}$$

Eq. (12) guarantees that if the two flows of a family are assigned to the same cell, flow 2 can only be started after flow 1 has been completed.

$$X_{f,f,c,1,2}\leqslant D_{f,c,1}+D_{f,c,2},\quad \forall f\in F,c\in C \tag{13}$$

$$D_{f,c,1}+D_{f,c,2}-1\leqslant X_{f,f,c,1,2},\quad \forall f\in F,c\in C \tag{14}$$

Eq. (13) and Eq. (14) together guarantee the relationship between two variables $D_{f,c,k}$ and $X_{f,f',c,k,k'}$ if the two flows of a family are assigned to the same cell.

$$B_{j,f,1,2}-E_{j,f,|M|,1}-t_{c,c'}+G\times\left(2-D_{f,c,1}-D_{f,c',2}\right)\geqslant 0,\quad \forall j\in J_f,f\in F,c,c'\in C \tag{15}$$

Eq. (15) guarantees that each job in each family can be started at flow 2 after it has been completed at flow 2 and transported to flow 2.

$$B_{j,f,m,k}-E_{j',f,m,k}+G\times\left(1-Y_{j,j',f}\right)\geqslant 0,\quad \forall j,j'\in J_f,f\in F \tag{16}$$

Eq. (16) denotes that each job in one family can be started only after the previous operations are completed if there is any previous job in the family.

$$Y_{j,j',f}+Y_{j',j,f}=1,\quad \forall j,j'\in J_f,f\in F \tag{17}$$

Eq. (17) guarantees that only one variable of the two variables, i.e., $Y_{j,j',f}$ and $Y_{j',j,f}$, can take the value of 1.

$$D_{f,c,k}\in\{0,1\} \tag{18}$$

$$Y_{j,j',f}\in\{0,1\} \tag{19}$$

$$X_{f,f',c,k,k'}\in\{0,1\} \tag{20}$$

Eqs. (18) to (20) define the possible values of the three decision variables.

## 4. Heuristic configuration for RDFGSP

In the last section, the model of the RDFGSP is discussed. To solve the problem, we focus on the constructive heuristics. Section 4.1 answers the first question raised in Section 2, while Sections 4.2–4.4 benefit from the literature and answer the second question. Primarily, given the absence of existing constructive heuristics tailored specifically for the RDFGSP, with the consideration of the problem-specific characteristics, the *meta*-algorithm including four categorical parameters and one numerical parameter is developed, which can be instantiated as a constructive heuristic. Subsequently, the potential values for the parameters are introduced. The specific value of each parameter will be determined by the I/F-Race detailed in the following section. Note that some notations used in this section can be found in Section 3.

### 4.1. Meta-algorithm of constructive heuristic

Recall that the RDFGSP requires dealing with three interdependent decisions mentioned in Section 3: (1) assignment of families to the cells at each flow, (2) scheduling sequence of families at the cells, and (3) scheduling sequence of jobs in the families. In order to tackle these problems, a procedure of constructive heuristic is designed. This procedure follows a sequential approach whereby the third decision is initially determined. Subsequently, the first and second decisions are ascertained through an iterative process.

**Algorithm 1**
Meta-algorithm of the constructive heuristic.

Initialize the set of job operations, i.e.,$O_{1,f}$, in each family $f$ at flow 1.
Initialize the set of job operations, i.e., $O_{2,f}$, in each family $f$ at flow 2.
Apply a sequencing rule SR on $O_{1,f}$ and $O_{2,f}$.
Initialize the set of family operations, i.e., $\Omega_1$, at flow 1.
Initialize the set of family operations, i.e., $\Omega_2$, at flow 2.
**Repeat**
  Step 1. Determine the cell $c^*$ by using a routing rule RR that is to process the family operation in $\Omega_1$.
  Step 2. Construct a set $\Omega'$ containing the available family operations to be scheduled in $\Omega_1$ in the following three sub-steps.
    Step 2.1. Calculate the completion time of the operations in $\Omega_1$ when they are scheduled on the cell $c^*$, and find the latest completion time $t(\Omega_1)$ among them.
    Step 2.2. Find the earliest beginning time $S(c^*)$ of $c^*$ according to Eq. (21).
    Step 2.3. Construct the set $\Omega'$ according to Eq. (22).
  Step 3. Apply the dispatching rule DR on the operations in $\Omega'$ to find the operation $O'$ to be scheduled on $c^*$.
  Step 4. Schedule $O'$ on $c^*$, and remove $O'$ from $\Omega_1$.
  Step 5. Add the following operation $next(O')$ of $O'$ into $\Omega_1$ if $next(O')$ exists in $\Omega_2$, and remove it from $\Omega_2$.
**Until** all operations in $\Omega_1$ and $\Omega_2$ have been scheduled.

Suppose that there are $|F|$ families to go through the two flows, and family $f$ contains $n_f$ jobs. **Algorithm 1** shows the procedure of the *meta*-algorithm to construct an active schedule, a non-delay schedule, or a

**Table 1**
The processing, setup, and transfer time of the example.

| Families | | | Family 1 | | | | Family 2 | | Family 3 | | Family 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Jobs | | | Job 1 | Job 2 | Job 3 | Job 4 | Job 1 | Job 2 | Job 1 | Job 2 | Job 1 | Job 2 | Job 3 |
| Processing time | Flow1 | Machine 1 | 66 | 89 | 81 | 53 | 91 | 68 | 56 | 95 | 88 | 75 | 98 |
| | | Machine 2 | 55 | 76 | 63 | 85 | 62 | 89 | 69 | 60 | 64 | 80 | 95 |
| | | Machine 3 | 81 | 50 | 57 | 68 | 69 | 95 | 87 | 77 | 53 | 74 | 99 |
| | Flow 2 | Machine 1 | 98 | 84 | 62 | 65 | 82 | 78 | 62 | 90 | 90 | 86 | 78 |
| | | Machine 2 | 81 | 74 | 62 | 69 | 93 | 96 | 52 | 88 | 61 | 79 | 93 |
| | | Machine 3 | 66 | 75 | 88 | 93 | 87 | 98 | 96 | 79 | 90 | 55 | 56 |
| Setup time | Flow1 | Machine 1 | 118 | | | | 146 | | 148 | | 127 | | |
| | | Machine 2 | 113 | | | | 125 | | 142 | | 139 | | |
| | | Machine 3 | 113 | | | | 102 | | 135 | | 105 | | |
| | Flow 2 | Machine 1 | 146 | | | | 118 | | 109 | | 126 | | |
| | | Machine 2 | 142 | | | | 149 | | 105 | | 112 | | |
| | | Machine 3 | 115 | | | | 146 | | 122 | | 146 | | |
| Transfer time | From Cell 1 | To Cell 1 | 69 | | | | 69 | | 69 | | 69 | | |
| | From Cell 1 | To Cell 2 | 84 | | | | 84 | | 84 | | 84 | | |
| | From Cell 2 | To Cell 1 | 52 | | | | 52 | | 52 | | 52 | | |
| | From Cell 2 | To Cell 2 | 56 | | | | 56 | | 56 | | 56 | | |

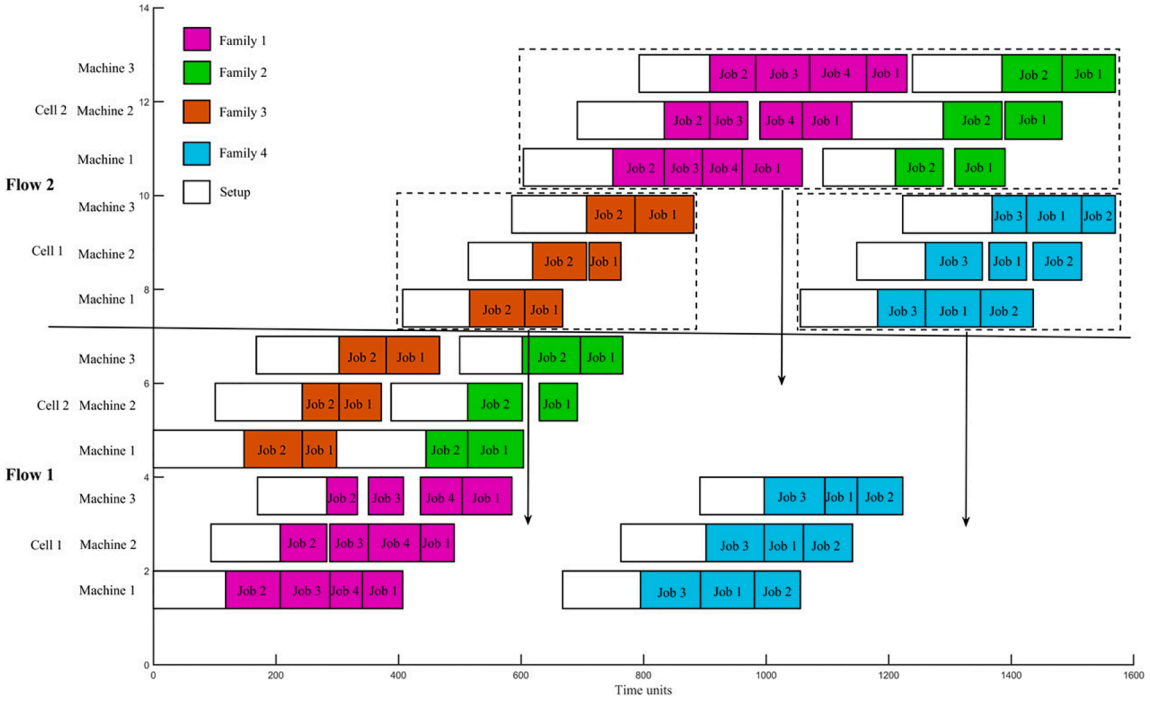



**Fig. 2.** Gantt chart of the schedule for example.

hybrid of both active and non-delay schedules. In the procedure, $O_{1,f}$ and $O_{2,f}$ represent the family operation of family $f$ at flow 1 and flow 2, respectively. They contain all the job operations $o_{1,j}$ and $o_{2,j}$ contained in family $f$. That is, $O_{1,f}$ and $O_{2,f}$ are initialized to $\{o_{1,1}, o_{1,2}, \ldots, o_{1,n_f}\}$ and $\{o_{2,1}, o_{2,2}, \ldots, o_{2,n_f}\}$, respectively. $\Omega_1$ and $\Omega_2$ are two sets of containing the family operations at flow 1 and flow 2, respectively. $\Omega_1$ is initialized to $\{O_{1,1}, O_{1,2}, \ldots, O_{1,i}, \ldots, O_{1,|F|}\}$, while $\Omega_2$ is initialized to $\{O_{2,1}, O_{2,2}, \ldots, O_{2,i}, \ldots, O_{2,|F|}\}$. $r(O^*)$ is the release/ready time of operation $O^*$. $U(c^*)$ is the ready/available time of cell $c^*$.

Once the job sequence in the family is determined, the processing of the family on the cell can also be confirmed. Thus, the procedure will first determine the scheduling sequence of the jobs in each family. After determining the sequences of jobs in all the families, an iterated procedure is conducted until all the operations in $\Omega_1$ and $\Omega_2$ have been scheduled. In the iterated procedure, the cell $c^*$ is determined by the routing rule RR to accommodate the next operation in Step 1. The routing rule makes real-time decisions to determine the cell with the highest priority. Subsequently, Step 2, which contains three sub-steps, aims to find available family operations in $\Omega_1$ to be scheduled on $c^*$. Step 2.1 is used to find the latest completion time of the operations in $\Omega_1$, while Step 2.2 is to find the earliest beginning time $S(c^*)$ of $c^*$. $S(c^*)$ can be calculated as follows:

$$S(c^*) = \max\{\min_{O^* \in \Omega_1}\{r(O^*)\}, U(c^*)\} \tag{21}$$

In Step 2.3, $\Omega'$ accommodates the operations that satisfy the time interval, which is defined by $S(c^*)$, $t(\Omega_1)$, and non-delay factor $\alpha$. $\Omega'$ can be defined as follows.

$$\Omega^{'} = \{\sigma \in \Omega_1 | r(\sigma) \leqslant S(c^*) + \alpha(t(\Omega_1) - S(c^*))\} \quad (22)$$

Step 3 uses the dispatching rule DR is used to select the operation in $\Omega^{'}$ to be scheduled on $c^*$. The most common form of DR is priority functions, which are used to assign priorities to operations in $\Omega^{'}$. The non-delay factor $\alpha \in [0,1]$ controls the look-ahead ability of the heuristic and determines the operations that should be considered to be processed next on $c^*$. If $\alpha = 0$, the procedure can only construct non-delay schedules, and only the operations that have already joined the queue are considered for scheduling. On the other hand, if $\alpha = 1$, the procedure will consider all the potential operations that are ready to join the queue of $c^*$ before the earliest completion time of $c^*$. After finding the operation $O^{'}$ in Step 3, the process of its scheduling on $c^*$ in Step 4 is shown below. Assume that $O^{'}$ contains the job operations in family $f^{'}$. Let $B_{j,f^{'},m,1}/E_{j,f^{'},m,1}$ and $B_{j,f^{'},m,2}/E_{j,f^{'},m,2}$ represent the beginning/ending time of job $j$ in operation $O^{'}$ on machine $m$ at flow 1 and flow 2, respectively. First, check whether its following operation $next(O^{'})$ at flow 2 exists in $\Omega_2$. If it exists, the scheduling is for its flow 1. Then, take job $j$ from $O^{'}$ one by one, and schedule them on the $|M|$ machines one by one according to Eq. (23) and Eq. (24). Otherwise, the scheduling is for its flow 2. Then, take job $j$ from $O^{'}$ one by one, and schedule them on the $|M|$ machines one by one according to Eq. (25) and Eq. (26), where $c^{'}$ represents the cell that is selected by $f^{'}$ at flow 1. The notations that are not mentioned in this subsection can be found in Section 3.

$$B_{j,f^{'},m,1} = \begin{cases} r(c^*) + s_{f^{'},m,1}, m = 1, j = 1 \\ \max\left(r(c^*) + s_{f^{'},m,1}, E_{i,f^{'},m-1,1}\right), m > 1, j = 1 \\ r(c^*), m = 1, j > 1 \\ \max\left(r(c^*), E_{j,f^{'},m-1,1}\right), m > 1, j > 1 \end{cases} \quad (23)$$

$$E_{j,f^{'},m,1} = B_{j,f^{'},m,1} + p_{j,f^{'},m,1} \quad (24)$$

$$B_{j,f^{'},m,2} = \begin{cases} \max\left(r(c^*) + s_{f^{'},m,2}, E_{j,f^{'},|M|,1} + t_{c^{'},c^*}\right), m = 1, j = 1 \\ \max\left(r(c^*) + s_{f^{'},m,2}, E_{j,f^{'},m-1,2}\right), m > 1, j = 1 \\ \max\left(r(c^*), E_{j,f^{'},|M|,1} + t_{c^{'},c^*}\right), m = 1, j > 1 \\ \max\left(r(c^*), E_{j,f^{'},m-1,2}\right), m > 1, j > 1 \end{cases} \quad (25)$$

$$E_{j,f^{'},m,2} = B_{j,f^{'},m,2} + p_{j,f^{'},m,2} \quad (26)$$

In summary, the meta algorithm can be seen as a configurable framework because of four configurable components: (1) the sequencing rule SR for sequencing the jobs in each family, (2) the routing rule RR to determine the cell that accommodates the operation that is to be scheduled, (3) the dispatching rule DR for assigning the operation on $c^*$, (4) and a non-delay factor $\alpha \in [0,1]$. Among them, the non-delay factor is a numerical parameter. The sequencing rule, routing rule and dispatching rule are categorical parameters and configurable, and thus their values are required to be designed specially.

### 4.2. Configurable sequencing rule

This subsection elucidates the configurable sequencing rule SR. The SR aims to sequence the jobs in each family at the flowlines, thus endowing this sequencing problem with certain FSP characteristics. Drawing upon the remarkable track record of the NEH in tackling the FSP with the aim of minimizing the makespan, we delve into the study of employing the NEH for sequencing the jobs in each family. The NEH heuristic consists of two phases: the sorting phase and the insertion phase. Framinan et al. (Framinan et al., 2003) and Abedinnia et al. (Abedinnia et al., 2016) summarized the room to improve the NEH by modifying the sorting phase and the insertion phase. Suppose that there are $n$ jobs in the given family. Based on the room for improvement, the general NEH framework can be summarized into three phases (i.e., index development, solution construction, and solution improvement) (Framinan et al., 2004), which is shown as follows.

**Step 1.** Calculate the indicator value $p_j$ of job $j$, where $j = 1, 2, \ldots, n$ and $n$ is the number of jobs.

**Step 2.** Sort the jobs in an order according to their indicator values based on the sorting criterion.

**Step 3.** Select the first two jobs, i.e., $j = 1$ and $j = 2$ in the order, and obtain the current partial sequence among the two partial sequences that results in a shorter makespan.

**Step 4.** For $j = 3, \ldots, n$, repeat the following substeps.

**Step 4.1.** Insert job $j$ in all $j$ possible slots in the current partial sequence, which consists of $j-1$ jobs.

**Step 4.2.** Select the best job partial sequence that results in the shortest makespan as the current partial sequence.

**Step 4.3.** Conduct a local search on the current partial sequence to obtain a better partial sequence.

The phase of index development involves Step 1 and Step 2, where the jobs are ranked according to a certain property based on their indicator values. The phase of solution construction involves Step 3 and Step 4 (Step 4.1-Step 4.2). In this phase, a solution is constructed in a recursive manner by trying one or more unscheduled jobs to be inserted in one or more positions of a partial sequence until the schedule is completed. In summary, the specific sequencing rule can be instantiated by selecting different options in the following three aspects: (1) in Step 1, different indicator calculation methods can be employed, (2) in Step 2, different sorting criteria can be selected, and (3) in Step 4.3, different local search methods can be used. Hence, the indicator calculation method, the sorting criterion method, and the local search method can be regarded as categorical parameters. These parameters are configurable, and their specific values are introduced below. That is, the sequencing rule SR can be instantiated by combining an indicator calculation method, a sorting criterion, and a local search method.

(a) Configurable indicator calculation criteria.

The indicator values are used in Step 2, and can reflect the processing properties of the jobs. To calculate the indicator values, several indicator calculation criteria are introduced, which focus on the process time of jobs from different aspects. The following criteria are widely used in the literature, and here they are tailored to suit the context of the RDFGSP by taking into account the two flows. Their time complexities can be reflected in the calculation formulas.

(1) *TPT*: Total processing time (*TPT*) (Nawaz et al., 1983), and $TPT_{j,f} = \sum_{k=1}^{2}\sum_{m=1}^{|M|} p_{j,f,m,k}$ for $j = 1, 2, \ldots, n_f$. This indicator is used in the original NEH.

(2) *DPT*: Deviation of processing time (*DPT*) (Dong et al., 2008), and $DPT_{j,f} = \frac{1}{2|M|}\left(\sum_{k=1}^{2}\sum_{m=1}^{|M|} p_{j,f,m,k}\right) + \sqrt{\frac{1}{2|M|-1}\sum_{k=1}^{2}\sum_{m=1}^{|M|}\left(p_{j,f,m,k} - \frac{1}{|M|}\sum_{m=1}^{|M|} p_{j,f,m,k}\right)^2}$ for $j = 1, 2, \ldots, n_f$.

This indicator considers the ideal of balancing the utilization among all machines.

(3) *SIP*: Slope Indices of Palmer (*SIP*) (Palmer, 1965), and $SIP_{j,f} = \sum_{k=1}^{2}\sum_{m=1}^{|M|}(2m - |M| - 1) \times p_{j,f,m,k}$ for $j = 1, 2, \ldots, n_f$. The Palmer heuristic aims to reduce of machines/job idle times.

(4) *ABS*(*SIP*): Absolute Slope Indices of Palmer (Abedinnia et al., 2016), and $ABS(SIP_{j,f}) = |SIP_{j,f}|$. The idea behind this is to better reflect the importance of jobs.

(5) *SIG*: Slope Indices of Gupta (Gupta, 1971), and $SIG_{j,f} = \frac{e_j}{\min_{1 \leqslant m \leqslant |M|-1, 1 \leqslant k \leqslant 2}\left(p_{j,f,m,k} + p_{j,f,m+1,k}\right)}$ for $j = 1, 2, \ldots, n_f$, where $e_j =$

$\begin{cases} 1 & \text{if } p_{j,f,1,1} < p_{j,f,|M|,2} \\ -1 & \text{otherwise} \end{cases}$. The indicator assigns a higher weight to jobs that are more likely to cause bottlenecks.

(6) *SIR*: Slope Indices of Rajendran (Rajendran, 1993), and $SIR_{j,f} = \sum_{k=1}^{2}\sum_{j=1}^{|M|}(|M| - m + 1) \times p_{j,f,m,k}$ for $j = 1, 2, \ldots, n_f$. The indicator assigns a higher weight to machines that are more likely to cause bottlenecks in production.

(7) *ABSDIF* (Framinan et al., 2003): Sum of the absolute differences of a job's processing times on every single machine with all other jobs, and $ABSDIF_{j,f} = \sum_{k=1}^{2}\sum_{m=1}^{|M|}\sum_{j=1}^{n_f}\left|p_{j,f,m,k} - p_{j',f,m,k}\right|$ for $j = 1, 2, \ldots, n_f$.

(8) *WABSDIF* (Framinan et al., 2003): Weighted sum of the absolute differences of a job's processing times on every single machine with all other jobs, and $WABSDIF_{j,f} = \sum_{k=1}^{2}\sum_{m=1}^{|M|}\sum_{j=1}^{n_f}(|M| - m + 1) \times \left|p_{j,f,m,k} - p_{j',f,m,k}\right|$ for $j = 1, 2, \ldots, n_f$.

(9) *SSSRA* (Framinan et al., 2003): sum of the absolute residuals, i.e., $SSSRA_{j,f} = \sum_{k=1}^{2}\sum_{j=1}^{n_f}\sum_{m=2}^{|M|}|r_{j,j',f,m,k}|$, where $r_{j,j',f,m,k} = p_{j,f,m,k} - p_{j',f,m-1,k}$ for $j = 1, 2, \ldots, n_f$.

(10) *SSWRSA* (Framinan et al., 2003): weighted sum of the absolute residuals, i.e., $SSSRA_{j,f} = \sum_{k=1}^{2}\sum_{j=1}^{n_f}\sum_{m=2}^{|M|}(|M| - m + 1) \times |r_{j,j',f,m,k}|$, where $r_{j,j',f,m,k} = p_{j,f,m,k} - p_{j',f,m-1,k}$ for $j = 1, 2, \ldots, n_f$.

Criteria (6) and (7) utilize the measures for the similarity of jobs by looking at the similarity of the processing times of different jobs on the same machine. Criteria (8) – (10) consider the similarity of jobs $j$ and $j'$ by defining the residual $r_{j,j',f,m,k}$.

(b) Configurable sorting criteria.

The sorting criteria are used in Step 2 to sort the jobs in a family based on their indicator values, thereby determining the job sequence in the families. Presented below are several commonly utilized criteria, and the reader can refer to (Framinan et al., 2003) for more information. They have the same time complexities, i.e., $O\left(n_f^2\right)$.

(1) Ascend: sort the jobs in ascending order of their indicator values.
(2) Descend: sort the jobs in descending order of their indicator values.
(3) Valley: sort the jobs as a "valley": low indicator values in the middle of the initial sequence and high values in the beginning and in the end.
(4) Hill: sort the jobs as a "hill": high indicator values in the middle of the initial sequence and low values in the beginning and in the end.
(5) HI/HILO: sort the jobs by alternately choosing one job with a high indicator value and one with a low value (HILO), starting with the extreme maximum and minimum values (HI/).
(6) HI/LOHI: sort the jobs by alternately choosing one job with a low indicator value and one with a high value (LOHI), starting with the extreme maximum and minimum values (HI/).
(7) LO/HILO: sort the jobs by alternately choosing one job with a high indicator value and one with a low value (HILO), starting with the smallest differences, extreme values in the end (LO/).
(8) LO/LOHI: sort the jobs by alternately choosing one job with a low indicator value and one with a high one (LOHI), starting with the smallest differences, extreme values in the end (LO/).

(c) Configurable local search methods.

The local search methods are used in Step 4.3, and aim to further improve the job sequence in the family. Since the local search methods extend the computational time and may not make up for the boost they bring, we also consider ignoring them in the heuristic. Regarding the first method, as it does not involve any operations, its time complexity is $O(0)$. As for the remaining three methods, they exhibit identical time complexities, i.e., $O(n_f^4 * |M|)$. There are also other methods in the literature, but they are mainly designed for special manufacturing environments that are not useful or applicable for the RDFGSP.

(1) Not adopted in the constructive heuristic (referred to as NA hereafter).
(2) Framinan and Leisten (Framinan and Leisten, 2003) performed pairwise exchanges at the end of each iteration to improve partial sequences (referred to as FL hereafter).
(3) Laha and Sarin (Laha and Sarin, 2009) extended FL by allowing all jobs assigned to a partial sequence to change their respective position by checking all other slots at the end of each iteration (referred to as LS hereafter).
(4) Abedinnia et al. (Abedinnia et al., 2016) modified the LS by using priority orders more effectively in the insertion phase (referred to as Hamid hereafter).

### 4.3. Configurable routing rule

For the routing rule RR, to determine the cell $c^*$ that accommodates the operation, the status of the cells should be considered. To this end, three different methods are introduced here, which have different computational complexities. For the first method, its time complexity is $O(|C|)$. For the second method, its time complexity is $O\left(\sum_{f=1}^{|F|} 3 \times n_f \times |C| \times |M|\right)$. For the third method, its time complexity is $O(2 \times |C| \times |F|)$. The notations not mentioned can be found in Section 3.

- Earliest available time (*EAT*). In Step 1 of the *meta*-algorithm, find the earliest available time of the cell among all the cells, and assign the cell as $c^*$.
- Earliest completion time (*ECT*). In Step 1 of the *meta*-algorithm, calculate the completion time of all the operations in $\Omega_1$ on all the cells. Find the earliest completion time of an operation on a cell, and assign the cell as $c^*$.
- Minimum working load (*MWL*). In Step 1 of the *meta*-algorithm, find the cell that has processed a minimum of operations, and assign the cell as $c^*$.

### 4.4. Configurable dispatching rule

For the dispatching rule DR, to find the operation $O'$ in $\Omega'$ to be scheduled on $c^*$, the processing properties of the operations in $\Omega'$ are required. The following seven methods are introduced. The time complexity of *FIFO* is $O(|\Omega'|)$, where $|\Omega'|$ is the number of operations in the set. The time complexities of *SPT* and *LPT* are $O\left(\sum_{f=1}^{|\Omega'|} n_f \times |M|\right)$. The time complexities of *SPTF* and *LPTF* are $O\left(\sum_{f=1}^{|\Omega'|} n_f\right)$. For *SPTB* and *LPTB*, their precise time complexities cannot be given since the bottleneck machine should be found. The worst time complexity is $O\left(|M| \times |\Omega'| + \sum_{f=1}^{|\Omega'|} n_f \times |M|\right)$.

- First in First out (*FIFO*). In Step 5 of the *meta*-algorithm, find the operation in $\Omega'$ with the earliest ready/release time, and set it as $O'$.
- Shortest/Longest processing time (*SPT*/*LPT*). In Step 5 of the *meta*-algorithm, find the operation in $\Omega'$ with the shortest/longest total processing time on all the machines, and set it as $O'$.
- Shortest/Longest processing time at the first stage (*SPTF*/*LPTF*). In Step 5 of the *meta*-algorithm, find the operation in $\Omega'$ with the shortest/longest processing time on the first machine, and set it as $O'$.

This rule is inspired by the fact that the processing time at the first stage has a larger influence.

- Shortest/Longest processing time till bottleneck stage (*SPTB/LPTB*). In Step 5 of the *meta*-algorithm, find the operation in $\Omega'$ with the shortest/longest processing time till the bottleneck machine, and set it as $O'$. The bottleneck stage is the machine that has the largest processing time for all the operations in $\Omega'$.

**Algorithm 2**
I/F-Race procedure.

**Input:** A set of tuning instances: $I$
Parameter space:$\Omega$
Total budget:$B$
**Output:**$\Theta^{\text{elite}}$
Let F-Race index $j$ to be 1.
$\Theta^j \sim SampleUniform(\Omega)$
$\Theta^{elite} \leftarrow Race(\Theta^j, B^j, I)$
$I \leftarrow I \backslash \{I_{used}\}$
$j \leftarrow j + 1$
**While** $B_{used} \leqslant B$ and $j \leqslant N^{race}$ **do**
  $\Theta^{\text{new}} \sim SampleNormal(\Omega, \Theta^{elite})$
  $\Theta^j = \Theta^{new} \cup \Theta^{elite}$
  $\Theta^{\text{elite}} \leftarrow Race(\Theta^j, B^j, I)$
  $I \leftarrow I \backslash \{I_{used}\}$
  $j \leftarrow j + 1$
**EndWhile**

### 4.5. Summary of the configurable parameters

Given the above, the configurable parameters and their value spaces are summarized in Table 2.

## 5. Automatic design process

In the last section, a configurable *meta*-algorithm framework and potential values of configurable parameters are given. In order to instantiate a particular configurable constructive, the I/F-Race is employed. This section begins by providing an introduce to the classical I/F-Race, followed by a detailed exposition of the core procedure, namely the race process, which has been designed to meet the specific requirements of the present study.

### 5.1. Classical I/F-Race

The outline of the I/F-Race procedure (López-Ibáñez et al., 2016) is given in **Algorithm 2.** There are primarily three steps in the procedure: (1) sampling new configurations according to a given distribution, (2) choosing the best configurations by racing from the newly sampled configurations, and (3) updating the sampling distribution to bias the sampling toward the best configurations. These three steps are repeated until the total budget $B$ or the number of races reaches the maximum value $N^{race}$. $SampleUniform(\Omega)$ means that a set of candidate configurations $\Theta_j$ is generated according to a uniform distribution from $\Omega$. At the first race, a set of candidate configurations $\Theta_1$ is generated through $SampleUniform(\Omega)$. $Race(\Theta_j, B_j, I)$ represents the procedure of the $j$th race to select a set $\Theta^{elite}$ of elite configurations from $\Theta_j$ by testing the instances from $I$ with the budget $B_j$. In one race, when the candidate configurations have been tested on an instance, the configurations that perform statistically worse than at least another one are discarded, and the race continues with the remaining surviving configurations to be tested on the next instance. The $j$th race terminates when the budget used reaches $B_j$ or a minimum number $MinSurvival$ of surviving configurations exist in $\Theta_j$. At the end of a race, the surviving configurations are sorted according to their performance by testing on the instances. Then, the first $\min(|\Theta_j|, MinSurvival)$ configurations are put into $\Theta^{elite}$. After a race, $I \leftarrow I \backslash \{I_{used}\}$ represents that the instances used in the above races are removed from the set. At the beginning of the following races, i.e., $j \neq 1$, a set $\Theta^{new}$ of new configurations is generated according to the normal distribution based on the elite configurations in $\Theta^{elite}$. Then, the configurations in $\Theta^{elite}$ and $\Theta^{new}$ are combined and act as the input to $Race(\Theta_j, B_j, I)$.

### 5.2. Race process

This subsection shows the procedure of the $j$th race, which aims to determine the configurations performing statistically worse that will be discarded, and the elite configurations that will enter into the $j+1$ th race. It is one key of the race, and its procedure is shown in **Algorithm 3**.

For solving one instance, the given heuristic outputs the fixed cost value. For this reason, each configuration is executed only once for one instance. Thus, to collect the statistical cost values for evaluating the performance of the candidate configurations, and to make statistical results more reasonable, credible and comprehensive, all candidate configurations are tested on a number of *NOI* (noted as $T^{each}$ in (López-Ibáñez et al., 2016) instances. Thus, for all the candidate configurations, a set of *NOI* cost values is collected to be compared by using the statistical test. $CCH(\Theta_c^j, I_{ins}^j)$ represents the running of the configurable constructive heuristic with the $c$th configuration in $\Theta^j$ for solving the test instance $I_{ins}^j$, which returns the objective value as the cost value. Since different tuning instances may have different dimensions of cost values, the Max-Min method is used to normalize the values to the range [0,1], resulting in $C_{c,ins}^j$. $C_{c,ins}^j$ represents the cost value of configuration $\Theta_c^j$ on instance $I_i^j$, which is stored in a value set $\varsigma_c$. The values in $\varsigma_c$ will accept the statistical test to determine whether the specific configuration performs statistically worse than at least one other one. The nonparametric Wilcoxon rank-sum test is used as the statistical test instead of Friedman's test recommended in (López-Ibáñez et al., 2016). The reason behind is that the Wilcoxon rank-sum test is widely used in the scheduling literature to conduct the statistical test (Zhang et al., 2020; Rivera et al., 2022; Kai et al., 2020). In the procedure, $Wilcoxon(\varsigma_c, \varsigma_{c'})$ represents conducting the nonparametric Wilcoxon rank-sum test on two value sets $\varsigma_c$ and $\varsigma_{c'}$. It returns "false" if $\varsigma_c$ is not statistically worse than $\varsigma_{c'}$; otherwise, it returns "true". After determining the statistical performance of all the configurations, the configurations are first sorted in ascending order of their average cost values over *NOI* instances. Then, the candidate configurations that perform statistically worse than at least one other configuration are in turn removed from the set reversely.

**Algorithm 3**
Procedure for the $j$th race.

**Input:** A set of candidate configurations $\Theta^j = \{\Theta_1^j, \ldots, \Theta_c^j, \ldots, \Theta_n^j\}$
  A set of test instances $I^j = \left\{I_1^j, \ldots, I_i^j, \ldots\right\}$
  Tuning budget: $B^j$
**Output:** The $\Theta^{elite}$ for the next race
Let $i = 1$,$UsingB = 0$
**While** $UsingB + |\Theta^j| \langle B^j$ **do**
  $UsingB + = |\Theta^j|$
  **For** $ins = i$ to $i + NOI$ **do**
    **For** $c = 1$ to $|\Theta^j|$ **do**
      $C_{c,ins}^j \leftarrow CCH(\Theta_c^j, I_{ins}^j)$
      Update $LB$ and $UB$ for $ins$th instance with $C_{c,ins}^j$
    **EndFor**
    **For** $c = 1$ to $|\Theta^j|$ **do**
      $C_{c,ins}^j \leftarrow MaxMin(C_{c,ins}^j, LB, UB)$
      Put $C_{c,ins}^j$ into set $\varsigma_c$
      $AvgCost_c + = C_{c,ins}^j$

(continued on next page)

**Algorithm 3** (*continued*)

```
13:     EndFor
14:   EndFor
15:   AvgCost_c / = NOI
16:   RankForElite_c + = AvgCost_c
17:   For c = 1 to |Θ^j| do
18:     For c' = 1 to |Θ^j| do
19:       If c! = c' do
20:         IsWorse_c ← Wilcoxon(ς_c, ς_c')
21:       Endif
22:     Endfor
23:   Endfor
24:   Θ^j ← RankConfigurations(Θ^j, AvgCost)
25:   For c = |Θ^j| to 1 do
26:     If IsWorse_c == true then
27:       Remove Θ^j_c from Θ^j
28:       If |Θ^j| ⟨ MinSurvival then
29:         Break;
30:       Endif
31:     Endif
32:   Endfor
33:   Let i = i + NOI
34: Endwhile
35: For c = 1 to |Θ^j| do
36:   RankForElite_c / = i
37: Endfor
38: Θ^j ← RankConfigurations(Θ^j, RankForElite)
39: For c = 1 to min(|Θ^j|, MinSurvival) do
40:   Put the configuration c into the set Θ^elite
41: Endfor
```

**Algorithm 4**
Sampling process based on the learning model.

```
Input: The elite configurations in BestSet from j-1th race
Output: The novel configurations for the j th race
1: For i = 1 to N^j_new do
2:   Let c = {X_1, X_2, X_3, X_4, X_5, X_6}
2:   Select a parent configuration with a probability p_z from the BestSet
3:   For d = 1 to 5 do
4:     Suppose that X_d ∈ {x^d_1, ..., x^d_i, ..., x^d_n}
4:     X_d ← Select a value from the solution space with a probability P_j(X_d = x^d_i)
5:   EndFor
6:   X_6 ← N(x^{d,z}, σ^j_d)
7:   Put the configuration c into Θ^j
8: EndFor
```

After that, the surviving configurations in $\Theta^j$ will be tested on the next *NOI* instances. The above procedure continues until a minimum number *MinSurvival* of surviving configurations exist in $\Theta^j$ or a predefined computational budget $B^j$ is used. At the end of a race, the surviving configurations are sorted in ascending order of their *RankForElite* values, which is obtained by averaging the cost values through the number of tuning instances used. Then, the first $\min(|\Theta^k|, MinSurvival)$ configurations are put into $\Theta^{elite}$. Finally, output the elite configurations in $\Theta^{elite}$.

Given the above, the race requires $N^{race}$ races, which depends on the number of configurable parameters $N^{param}$ ($N^{param} = 6$ in this paper). The computational budget $B^j$ in the $j$th race is related to the total budget $B$ and the budget $B_{used}$ used in the previous $j-1$ races. The number of candidate configurations $|\Theta^j|$ in the $j$th race can be calculated based on $B^j$ and a control parameter $\mu$. The above parameter definitions and their relationships are shown in Eqs. (27) to (29). One can refer to (López-Ibáñez et al., 2016) for more information.

$$N^{race} = \lfloor 2 + \log_2 N^{param} \rfloor \tag{27}$$

$$B^j = (B - B_{used}) / (N^{race} - j + 1),\ j = 1, \ldots, N^{race} \tag{28}$$

$$|\Theta^j| = \lfloor B^j / (\mu + \min(5, j)) \rfloor, j = 1, \ldots, N^{race} \tag{29}$$

At the end of a race, a number of novel candidate configurations should be generated for the next race by sampling from the learning model based on a number of *NOE* elite configurations from $\Theta^{elite}$. Generally, *NOE* is smaller than *MinSurivival*. This can bias the sampling toward the best configurations. **Algorithm 4** summarizes the sampling process based on the learning model. In the algorithm, $N^j_{new}$ denotes the number of novel configurations that are generated for the $j$th race, and its value can be calculated as $N^j_{new} = |\Theta^j| - |\Theta^{elite}|, j = 2, \ldots N^{race}$. $p_z$ denotes the probability, with which one parent configuration $\Theta^p$ can be selected from the **BestSet**, which accommodates the best $\min(|\Theta^{elite}|, NOE)$ configurations from $\Theta^{elite}$. $p_z$ can be calculated as follows.

$$p_z = \frac{|\boldsymbol{BestSet}^{k-1}| - r_z + 1}{|\boldsymbol{BestSet}^{k-1}| \cdot (|\boldsymbol{BestSet}^{k-1}| + 1)/2} \tag{30}$$

where $r_z$ is the rank index of the configurations. This can ensure that a configuration with a better performance has a higher probability of being selected as the parent. Lines 2–5 show the process of generating values for the categorical parameters, i.e., $X_1$, $X_2$,$X_3$,$X_4$, and $X_5$. $P_j(X_d = x^d_i)$ denotes the probability of obtaining the value $x^d_i$ for the parameter $X_d$ at the $j$th race. $P_j(X_d = x^d_i)$ can be calculated as follows.

$$\begin{aligned} P_j(X_d = x^d_i) &= P^z_{j-1}(X_d = x^d_i) \cdot \left(1 - \frac{j-1}{N^{race}}\right) + \Delta P, \quad \text{where } \Delta P \\ &= \begin{cases} \dfrac{j-1}{N^{race}} & if \quad x^d_i = x^{d,z}_i \\ 0 & otherwise \end{cases} \end{aligned} \tag{31}$$

where $x^{d,z}_i$ is the relevant parameter value in the parent configuration $\Theta^z$. Line 6 shows the process of generating value for the numerical parameter, i.e., $X_6$. Suppose that a numerical parameter can be sampled from the range $[\underline{x_d}, \overline{x_d}]$. $N(x^{d,z}, \sigma^j_d)$ denotes the truncated normal distribution, where $x^{d,z}$ is the value of $X_d$ in the selected parent configuration $\Theta^z$, and $\sigma^j_d$ is initially set to $(\overline{x_d} - \underline{x_d})/2$ at the end of the first race to sample the parameter values for the second race, i.e., $j = 1$, and it is decreased with the number of races as follows. In this way, the sampled parameter values are increasingly closer to the value of the parent configuration, benefiting the search around the promising configurations.

$$\sigma^j_d = \sigma^j_d \cdot \left(\frac{1}{N^j_{new}}\right)^{1/N^{param}} \tag{32}$$

## 6. Computational and statistical experiments

This section endeavors to undertake computational and statistical experiments. Primarily, substantial quantities of tuning and testing instances are collected to serve as comprehensive data, and the experimental setup is given. Secondly, the tuning phase for conceiving the constructive heuristic is presented. Subsequently, the performance of the conceived constructive heuristic is evaluated in the testing phase. Finally, the effectiveness and efficiency of the designed AAD are demonstrated. The source codes pertaining to the AAD procedure and CPLEX model are publicly accessible via this link: at https://pan.baidu.com/s/1PnXZfzhwi66nJfXEdIjyqQ?pwd=Lcus.

### 6.1. Tuning and testing data

To make comprehensive data, two sets of instances, encompassing varying problem scales, are generated utilizing the widely used design (Ruiz et al., 2019; Shao et al., 2021) in the literature. One set accommodates the small-scale instances, while another set accommodates the large-scale instances. The instances are labeled with the number of

**Table 2**
Configurable parameters.

| Name | Rule | Label | Type | Distribution | Space |
|---|---|---|---|---|---|
| Indicator calculation criterion | SR | $X_1$ | Categorical | Discrete distribution | From the set {*TPT*, *DPT*, *SIP*, *ABS*(*SIP*), *SIG*, *SIR*, *ABSDIF*, *WABSDIF*, *SSSRA*, *SSWRSA*} |
| Sorting criterion | SR | $X_2$ | Categorical | Discrete distribution | From the set {*Ascend*, *Descend*, *Valley*, *Hill*, *HI/HILO*, *HI/LOHI*, *LO/HILO*, *LO/LOHI*} |
| Local search method | SR | $X_3$ | Categorical | Discrete distribution | From the set {*NA*, *FL*, *LS*, *Hamid*} |
| Operation selection method | DR | $X_4$ | Categorical | Discrete distribution | From the set {*FIFO*, *LPT*, *SPT*, *LPTF*, *LPTB*, *SPTF*, *SPTB*} |
| Cell assignment method | RR | $X_5$ | Categorical | Discrete distribution | From the set {*EAT*, *ECT*, *MWL*} |
| Non-delay factor | $\alpha$ | $X_6$ | Numerical | Continuous distribution | From the range [0,1] |

families denoted by $f$, the number of cells denoted by $c$, and the number of machines in the flowline denoted by $m$. For the small-scale instances, $f$ comes from $\{3,4,5,6,7\}$, $c$ comes from $\{2,3\}$, and $m$ comes from $\{3,4\}$. This results in 20 different combinations of $f \times c \times m$. Each family comprises a number of jobs, sampled uniformly from the range of (Huang and Zhou, 2022; López-Ibáñez et al., 2016). For large-scale instances, $f$ comes from $\{15,20,25,30\}$, $c$ comes from $\{3,5,8\}$, and $m$ comes from $\{3,5,8\}$. The number of jobs for each family is sampled uniformly from the distribution of $U$ (Zhao et al., 2023; Naderi and Ruiz, 2010). Consequently, this leads to a total of 36 combinations of $f \times c \times m$. As the problem scale increases, the solution space expands correspondingly, giving rise to increased complexity in finding a solution.

The tuning data are used in the tuning phase to determine the best configuration by using the AAD. A series of 500 tuning instances with different problem scales are randomly generated as the tuning data. These instances constitute the set $I = \{I_1, \ldots, I_i, \ldots I_G\}$ in **Algorithm 2**. The testing data are used in the testing phase to demonstrate the effectiveness of the configuration conceived by the AAD. In the testing data, for each combination of the small-scale instances, one instance is randomly generated. This results in 20 instances. For each combination of the large-scale instances, ten instances are randomly generated, resulting in 360 instances in total.

Concerning the generation of the processing time, we consider the following reasonable ranges in accordance with the real-world configuration outlined in Section 3. The processing time of each job on a specific machine is derived from a uniform distribution of $U[50, 100]$. Meanwhile, the family setup time and transportation time between cells are drawn from uniform distributions of $U[100, 150]$ and $U[50, 100]$, respectively.

### 6.2. Experimental setup

We set the computational budget as a number of experiments for the tuning phase, where an experiment is the application of a configuration to *NOI* tuning instances. The value of *NOI* should be set appropriately. The reason behind this is that, a small value cannot obtain sufficient data to evaluate the statistical results, while a large value may consume too many computing resources. According to Eq. (25), the number of races is related to the number of parameters that need to be configured. Regarding the total budget $B$, the larger the value is, the more candidate configurations can be evaluated in more detail. Regarding the control parameter $\mu$, as shown in Eq. (27), it is related to the number of candidate configurations at each race. It allows the user to influence the ratio between the total budget and number of candidate configurations. Thus, to ensure that enough configurations can be evaluated sufficiently with high efficiency, the two parameters should be set appropriately. Regarding the parameters *NOE* and *MinSurvival*, the appropriate values can help the sampling toward the best configurations. A large *NOE* value makes the convergence rate of the procedure very low, while a small *NOE* value cannot ensure global optimality. A large *MinSurvival* value cannot make the candidate configurations sufficiently evaluated, while a small *MinSurvival* value cannot ensure global optimality. Generally, *NOE* is smaller than *MinSurvival*. Through preliminary experiments, suitable values for the parameters are determined, i.e., $NOI = 10, B = 6000, \mu = 1.5$, $MinSurvival = 20$ and $NOE = 10$. Under these settings, it is sufficient for the AAD procedure to converge and have global optimality. That is, the selected elite configurations change slightly with the increase of the total budget.

To evaluate the performance of the heuristics in the testing phase, in this study, the relative percentage deviation (*RPD*) (Zhang et al., 2021) (Zhang et al., 2020) between solutions $S$ and $S^{best}$ is selected as the performance metric to evaluate the algorithm performance.

$$RPD = \frac{f(S) - f\left(S^{best}\right)}{f\left(S^{best}\right)} * 100 \tag{31}$$

where $f(S)$ is the objective value obtained by the given algorithm, and $f\left(S^{best}\right)$ is the best objective value obtained by all the compared algorithms. All of the algorithms used in this paper are coded in C++, and the experiments are run on a 3.60 GHz Intel Core i7 CPU processor.

**Table 4**
Best elite configurations output by the AAD.

| Rank | Configurable parameter | | | | | |
|---|---|---|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
| 1 | *SIP* | *Ascend* | *LS* | *LPT* | *ECT* | 0 |
| 2 | *SIR* | *Descend* | *LS* | *LPT* | *ECT* | 0.76 |
| 3 | *SIR* | *Descend* | *LS* | *LPT* | *ECT* | 0.59 |
| 4 | *ABSDIF* | *HI/LOHI* | *Hamid* | *LPT* | *ECT* | 0 |
| 5 | *SIR* | *Descend* | *LS* | *LPT* | *ECT* | 0.54 |
| 6 | *ABSDIF* | *HI/LOHI* | *Hamid* | *LPT* | *ECT* | 0 |
| 7 | *ABSDIF* | *HI/LOHI* | *LS* | *LPT* | *ECT* | 0 |
| 8 | *ABSDIF* | *HI/LOHI* | *Hamid* | *LPT* | *ECT* | 0.20 |

**Table 3**
Procedural parameters of the AAD procedure.

| Race number | Limited budget | Number of test instances | Number of Candidate configurat-ions | Number of discarded configurat-ions | Number of remaining configurat-ions | Number of selected elite configurat-ions | Number of generated configurat-ions |
|---|---|---|---|---|---|---|---|
| 1 | 1500 | 70 | 600 | 586 | 14 | 10 | 466 |
| 2 | 1680 | 40 | 480 | 468 | 12 | 10 | 462 |
| 3 | 2135 | 100 | 474 | 456 | 18 | 10 | 524 |
| 4 | 2984 | 130 | 542 | 532 | 10 | 8 | – |

**Table 5**
Objective (time) values obtained by the CPLEX and constructive heuristics on small instances.

| Problem | Constructive heuristics | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPLEX | AAD | LPT_GA_NEH | NEH_NR | NEH2_en | DNEH1 | DNEH2 | DNEH3 | NEH2L | NEH2D | NEH2E | NEH2E_ex | NEH2EE |
| 3 × 2 × 3 | **1824**(60 s) | 1847(0) | 2254(0) | 2254(0) | 2254(0) | 2017 (0) | 2017(0) | 2017 (0) | 2266 (0) | 2266 (0) | 2258(0) | 2258(0) | 2258(0) |
| 3 × 2 × 4 | **1529**(45 s) | 1708(0) | 2055(0) | 2055(0) | 2055(0) | 1874 (0) | 1874(0) | 1874 (0) | 2057 (0) | 2055 (0) | 2055(0) | 2055(0) | 2020(0) |
| 3 × 3 × 3 | **1396**(60 s) | 1445(0) | 1648(0) | 1648(0) | 1648(0) | 1648 (0) | 1648(0) | 1648 (0) | 1647 (0) | 1638 (0) | 1638(0) | 1638(0) | 1629(0) |
| 3 × 3 × 4 | **1503**(60 s) | 1592(0) | 1588(0) | 1588(0) | 1588(0) | 1588 (0) | 1588(0) | 1588 (0) | 1628 (0) | 1609 (0) | 1596(0) | 1596(0) | 1605(0) |
| 4 × 2 × 3 | **2729**(60 s) | 2811(0) | 3072(0) | 3072(0) | 3072(0) | 2908 (0) | 2908(0) | 2908 (0) | 3039 (0) | 3039 (0) | 3039(0) | 3039(0) | 3036(0) |
| 4 × 2 × 4 | **2328**(60 s) | 2387(0) | 2907(0) | 2907(0) | 2907(0) | 2495 (0) | 2495(0) | 2495 (0) | 2914 (0) | 2913 (0) | 2913(0) | 2913(0) | 2931(0) |
| 4 × 3 × 3 | **1823**(60 s) | 1897(0) | 2577(0) | 2196(0) | 2577(0) | 2148 (0) | 2148(0) | 2148 (0) | 2577 (0) | 2558 (0) | 2567(0) | 2567(0) | 2550(0) |
| 4 × 3 × 4 | **1999**(60 s) | 2083(0) | 3218(0) | 2637(0) | 3218(0) | 2357 (0) | 2357(0) | 2357 (0) | 3243 (0) | 3243 (0) | 3243(0) | 3243(0) | 3221(0) |
| 5 × 2 × 3 | **2445**(60 s) | 2713(0) | 3414(0) | 2855(0) | 3414(0) | 2855 (0) | 2855(0) | 2855 (0) | 3414 (0) | 3414 (0) | 3414(0) | 3414(0) | 3444(0) |
| 5 × 2 × 4 | **3598**(60 s) | 3599(0) | 4112(0) | 4112(0) | 4112(0) | 4087 (0) | 4087 (16) | 4087 (0) | 4115 (0) | 4115 (0) | 4115(0) | 4115(0) | 4132(0) |
| 5 × 3 × 3 | 3105(60 s) | **2501**(0) | 2824(0) | 2517(0) | 2824(0) | 2569 (0) | 2569(0) | 2569 (0) | 2851 (0) | 2864 (0) | 2864(0) | 2864(0) | 2823(0) |
| 5 × 3 × 4 | 2197(60 s) | **2163**(0) | 2769(0) | 2739(0) | 2769(0) | 2528 (0) | 2528(0) | 2528 (0) | 2770 (0) | 2760 (0) | 2755(0) | 2755(0) | 2747(0) |
| 6 × 2 × 3 | 3271(60 s) | 3254(0) | 3367(0) | 3410(0) | 3367(0) | 3239 (0) | **3227(0)** | 3239 (0) | 3370 (0) | 3351 (0) | 3348(0) | 3348(0) | 3351(0) |
| 6 × 2 × 4 | 3670(60 s) | **3263 (0)** | 3706(0) | 3756(0) | 3706(0) | 3741 (0) | 3718(0) | 3718 (0) | 3658 (0) | 3653 (0) | 3647(0) | 3647(0) | 3686(0) |
| 6 × 3 × 3 | 3188(60 s) | **2555**(0) | 2863(0) | 2863(0) | 2863(0) | 3139 (0) | 3139(0) | 3139 (0) | 2820 (0) | 2806 (0) | 2837(0) | 2837(0) | 2652(0) |
| 6 × 3 × 4 | 3012(60 s) | **2386**(0) | 2635(0) | 2635(0) | 2635(0) | 2521 (0) | 2521(0) | 2521 (0) | 2642 (0) | 2642 (0) | 2641(0) | 2641(0) | 2641(0) |
| 7 × 2 × 3 | 4646(60 s) | 4334(0) | 4315(0) | 4315(0) | 4315(0) | 4920 (0) | 4832(0) | 4832 (0) | 4324 (0) | 4322 (0) | **4298 (0)** | **4298(0)** | 4337(0) |
| 7 × 2 × 4 | 4499(60 s) | **4175**(0) | 4644(0) | 4237(0) | 4644(0) | 4550 (0) | 4550(0) | 4550 (0) | 4636 (0) | 4650 (0) | 4624(0) | 4624(0) | 4645(0) |
| 7 × 3 × 3 | 3570(60 s) | **2550**(0) | 2981(0) | 2820(0) | 2965(0) | 2846 (0) | 2846(0) | 2846 (0) | 2992 (0) | 2992 (0) | 2992(0) | 2976(0) | 3457(0) |
| 7 × 3 × 4 | 3971(60 s) | **2722**(0) | 2844(0) | 2983(0) | 2844(0) | 3136 (0) | 3115(0) | 3115 (0) | 2849 (0) | 2849 (0) | 2862(0) | 2862(0) | 2870(0) |

### 6.3. Tuning phase

Table 3 summarizes some important procedural data in the tuning phase of the AAD. There exist four iterated races in the process. From Table 3, a total of 2052 (600 + 480 + 474 + 542 - 14 - 12 - 18) configurations have gone through the competitive process, and finally, 8 configurations are identified as elite configurations. In the process, a total of 340 (70 + 40 + 100 + 130) tuning instances are tuned to evaluate the performance of the configurations. We also find the trend that more evaluations per configuration (dividing the limited budget by the number of candidate configurations) will be performed as the race progresses. The reason behind this is that similar configurations are grouped in the later races, and thus more evaluations are requested to evaluate their performance.

Through four races and after testing 340 instances, the AAD outputs 8 elite configurations, which are shown in Table 4. From Table 4, we can see that the configurations take different parameter values. This indicates that promising heuristics can be constructed by different components. For the categorical parameter $X_1$, i.e., the indicator calculation criterion, three methods appear in the promising configurations: *SIP*, *SIR*, and *ABSDIF*. For the categorical parameter $X_2$, i.e., the sorting criterion, there are also three methods: *Ascend*, *Descend*, and *HI/LOHI*. There is a one-to-one correspondence between the three indicator calculation methods and the three sorting methods. More specifically, the *SIP* and *SIR* are associated with *Ascend* and *Descend*, respectively, while the *ABSDIF* is bound with *HI/LOHI*. For the categorical parameter $X_3$, only two methods, i.e., *LS* and *Hamid*, emerge in the promising solutions. These two methods are both extensions of the FL by allowing all jobs to adjust their positions. With regard to the categorical parameters $X_4$ and $X_5$, i.e., the operation selection method and cell assignment method, it can be seen that only one method is suitable for each parameter. This indicates that *LPT* performs much better than the other methods for the dispatching rule, as well as *ECT* for the routing rule. In other words, the configurations only with *LPT* and *ECT* can have promising performance. As regards the numerical parameter $X_6$, i.e., the non-delay factor, it can take values from different ranges. That is, for solving the RDFGSP, the complete schedule can be constructed by using delay or non-delay dispatching rules. In the following testing phase, the best configuration that ranks first in Table 4 will configure the automated heuristic to address the RDFGSP.

### 6.4. Testing phase

In this subsection, we evaluate the performance of the automated heuristic (referred to as AAD below), which is configured by the best configuration that ranks first in Table 4. Since there is no constructive heuristic in the literature to deal with the RDFGSP, several constructive heuristics that address the related problems, as well as the MILP model presented in Section 3, are selected as the compared algorithms. They are LPT_GA_NEH (Pan et al., 2022), NEH2 (Naderi and Ruiz, 2010), NEH_en (Ruiz et al., 2019), DNEH1 (Shao et al., 2021), DNEH2 (Shao et al., 2021), DNEH3 (Shao et al., 2021), NEH2L (Chen et al., 2021),

**Table 6**
ARPD (time) values obtained by the constructive heuristics on large instances.

| Problem | Constructive heuristics | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AAD | LPT_GA_NEH | NEH_NR | NEH2_en | DNEH1 | DNEH2 | DNEH3 | NEH2L | NEH2D | NEH2E | NEH2E_ex | NEH2EE |
| 15 × 3 × 3 | **0.418** (0.01) | 6.424(0) | 3.77(0) | 6.371(0) | 2.518(0) | 1.573(0) | 1.544(0) | 6.494(0) | 6.431(0) | 6.463(0) | 6.42(0) | 6.107(0) |
| 15 × 3 × 5 | **0.437** (0.02) | 8.544(0) | 7.055(0) | 8.27(0) | 3.059(0) | 1.834(0) | 2.574(0) | 8.471(0) | 8.547(0) | 8.6(0) | 8.44(0) | 9.269(0) |
| 15 × 3 × 8 | **0**(0.03) | 15.465 (0.01) | 16.077(0) | 15.225(0) | 14.848 (0) | 13.478 (0) | 14.179 (0) | 15.525 (0) | 15.24(0) | 15.24 (0.01) | 14.966(0) | 16.194 (0.01) |
| 15 × 5 × 3 | **0.946** (0.02) | 8.978(0.01) | 8.107(0) | 8.572(0) | 8.558(0) | 7.516(0) | 8.265(0) | 8.995(0) | 9.119(0) | 9.228(0) | 8.756(0) | 9.348(0) |
| 15 × 5 × 5 | **0.866** (0.03) | 11.848(0) | 9.139(0) | 11.717(0) | 1.927(0) | 1.404(0) | 1.757(0) | 12.277 (0) | 12.152 (0) | 12.283 (0) | 12.024(0) | 13.344(0) |
| 15 × 5 × 8 | **0.696** (0.05) | 6.514(0) | 5.778(0) | 6.345 (0.01) | 1.491(0) | 1.599(0) | 1.455 (0.01) | 6.746 (0.01) | 6.926 (0.01) | 6.926(0) | 6.88(0) | 6.925(0) |
| 15 × 8 × 3 | **1.257** (0.03) | 20.089(0) | 11.142(0) | 20.089 (0.01) | 14.329 (0) | 13.446 (0) | 13.446 (0) | 19.092 (0) | 19.227 (0) | 18.9(0) | 18.9(0) | 18.406(0) |
| 15 × 8 × 5 | **0.971** (0.01) | 16.863(0) | 13.523(0) | 16.863 (0.01) | 3.142(0) | 3.026 (0.01) | 3.026(0) | 17.114 (0.01) | 17.058 (0) | 17.099 (0.01) | 17.099 (0.01) | 15.374 (0.01) |
| 15 × 8 × 8 | **0**(0.08) | 21.192 (0.01) | 20.579(0) | 21.192 (0.01) | 11.17 (0.01) | 11.17(0) | 11.17(0) | 22.014 (0.01) | 21.674 (0.01) | 21.771 (0.01) | 21.771 (0.01) | 21.678 (0.01) |
| 20 × 3 × 3 | **0.067** (0.03) | 8.4(0) | 3.805(0) | 8.156 (0.01) | 3.391(0) | 2.372(0) | 3.067(0) | 8.223(0) | 8.283(0) | 8.198(0) | 8.085(0) | 9.078(0) |
| 20 × 3 × 5 | **0**(0.04) | 12.045 (0.01) | 4.623(0) | 11.925 (0.01) | 6.4(0) | 5.016(0) | 5.595(0) | 12.589 (0) | 12.093 (0) | 12.136 (0.01) | 11.968 (0.01) | 12.243 (0.01) |
| 20 × 3 × 8 | **0**(0.06) | 20.182 (0.01) | 18.787 (0.01) | 20.059 (0.01) | 20.12(0) | 18.759 (0.01) | 19.087 (0.01) | 21.436 (0.01) | 21.235 (0.01) | 21.231 (0.01) | 21.119 (0.01) | 19.733 (0.01) |
| 20 × 5 × 3 | **0.497** (0.03) | 8.716(0.01) | 7.217(0) | 8.595 (0.01) | 1.654(0) | 2.867(0) | 1.117 (0.01) | 9.537(0) | 8.595 (0.01) | 8.63(0) | 8.464(0) | 7.262 (0.01) |
| 20 × 5 × 5 | **0.147** (0.05) | 12.035(0) | 7.238(0) | 11.842 (0.01) | 3.011(0) | 1.798(0) | 2.969(0) | 11.731 (0.01) | 11.651 (0.01) | 11.734 (0.01) | 11.651 (0.01) | 10.763 (0.01) |
| 20 × 5 × 8 | **0**(0.09) | 16.592 (0.01) | 18.122(0) | 16.339 (0.01) | 11.751 (0.01) | 10.781 (0.01) | 11.525 (0.01) | 16.49 (0.01) | 16.21 (0.01) | 16.235 (0.01) | 16.274 (0.01) | 17.296 (0.01) |
| 20 × 8 × 3 | **0.135** (0.06) | 22.49(0.01) | 10.046(0) | 21.351 (0.01) | 7.341(0) | 6.965(0) | 7.25(0) | 22.594 (0) | 22.3 (0.01) | 22.289 (0.01) | 21.271 (0.01) | 22.153 (0.01) |
| 20 × 8 × 5 | **0.233** (0.09) | 18.882 (0.01) | 7.877(0) | 18.468 (0.01) | 3.75 (0.01) | 3.226 (0.01) | 3.488 (0.01) | 18.912 (0.01) | 18.721 (0) | 18.877 (0.01) | 18.236 (0.01) | 20.178 (0.01) |
| 20 × 8 × 8 | **0**(0.16) | 23.525 (0.02) | 16.013(0) | 23.32 (0.02) | 13.933 (0.01) | 13.517 (0.01) | 13.933 (0.01) | 23.045 (0.01) | 23.686 (0.01) | 23.673 (0.01) | 23.673 (0.01) | 22.576 (0.01) |
| 25 × 3 × 3 | **0.523** (0.05) | 8.316(0.01) | 1.391 (0.01) | 8.314 (0.01) | 6.218(0) | 4.446 (0.01) | 5.445 (0.01) | 8.597 (0.01) | 8.576 (0.01) | 8.473 (0.01) | 8.525 (0.01) | 6.057 (0.01) |
| 25 × 3 × 5 | **0.006** (0.06) | 6.115(0.01) | 3.05 (0.01) | 6.105 (0.01) | 8.041 (0.01) | 6.663 (0.01) | 7.532 (0.01) | 6.271 (0.01) | 6.028 (0.01) | 6.079 (0.01) | 6.03(0.01) | 7.513 (0.01) |
| 25 × 3 × 8 | **0**(0.09) | 24.931 (0.01) | 20.515 (0.01) | 24.772 (0.02) | 24.912 (0.01) | 23.253 (0.01) | 24.376 (0.01) | 25.878 (0.02) | 24.965 (0.02) | 25.023 (0.02) | 24.953 (0.02) | 24.201 (0.02) |
| 25 × 5 × 3 | **0.773** (0.06) | 9.887(0.01) | 5.81(0) | 9.783 (0.01) | 1.572(0) | 0.554(0) | 1.57(0) | 9.83 (0.01) | 9.893 (0.01) | 9.777 (0.01) | 9.733 (0.01) | 9.795 (0.01) |
| 25 × 5 × 5 | **0.509** (0.09) | 9.276(0.01) | 6.632 (0.01) | 9.036 (0.01) | 2.375 (0.01) | 1.231 (0.01) | 2.001 (0.01) | 7.626 (0.01) | 9.258 (0.01) | 9.194 (0.01) | 9.113 (0.01) | 5.295 (0.01) |
| 25 × 5 × 8 | **0**(0.15) | 18.995 (0.01) | 19.838 (0.01) | 19.002 (0.02) | 16.273 (0.01) | 14.86 (0.01) | 16.092 (0.01) | 18.802 (0.02) | 18.753 (0.02) | 18.753 (0.02) | 18.669 (0.02) | 18.265 (0.02) |
| 25 × 8 × 3 | **0.226** (0.11) | 18.381(0) | 6.6(0) | 18.063 (0.01) | 9.674 (0.01) | 9.014(0) | 9.487 (0.01) | 18.384 (0.01) | 18.441 (0.01) | 18.366 (0.01) | 18.035 (0.01) | 18.105 (0.01) |
| 25 × 8 × 5 | **0**(0.15) | 18.92(0.02) | 10.063(0) | 18.767 (0.02) | 12.93 (0.01) | 11.64 (0.01) | 12.296 (0.01) | 20.26 (0.01) | 20.489 (0.01) | 18.979 (0.01) | 18.835 (0.01) | 15.876 (0.01) |
| 25 × 8 × 8 | **0**(0.24) | 20.217 (0.02) | 13.797 (0.01) | 20.198 (0.02) | 19.089 (0.01) | 18.248 (0.02) | 18.46 (0.01) | 20.163 (0.02) | 20.05 (0.02) | 20.055 (0.02) | 20.055 (0.02) | 20.989 (0.02) |
| 30 × 3 × 3 | **0.465** (0.06) | 6.885(0.01) | 2.441 (0.01) | 6.769 (0.01) | 3.986 (0.01) | 2.692 (0.01) | 3.707 (0.01) | 7.389 (0.01) | 7.16 (0.01) | 7.176 (0.01) | 7.104 (0.02) | 8.283 (0.02) |
| 30 × 3 × 5 | **0**(0.1) | 6.774(0.02) | 4.607 (0.01) | 6.624 (0.02) | 4.776 (0.01) | 3.255 (0.02) | 4.609 (0.01) | 6.85 (0.02) | 6.785 (0.01) | 6.819 (0.01) | 6.743 (0.02) | 5.924 (0.02) |
| 30 × 3 × 8 | **0**(0.15) | 18.627 (0.03) | 19.075 (0.02) | 18.543 (0.04) | 17.909 (0.01) | 16.484 (0.02) | 17.591 (0.02) | 18.493 (0.02) | 18.453 (0.02) | 18.446 (0.02) | 18.369 (0.03) | 18.929 (0.03) |
| 30 × 5 × 3 | **0.549** (0.10) | 5.698(0.01) | 4.982 (0.01) | 5.565 (0.01) | 5.713 (0.01) | 3.793 (0.01) | 4.683 (0.01) | 6.326 (0.01) | 5.913 (0.01) | 5.821 (0.01) | 5.759 (0.01) | 6.865 (0.01) |
| 30 × 5 × 5 | **0**(0.16) | 10.78(0.01) | 7.92 (0.01) | 10.692 (0.02) | 3.881 (0.01) | 3.122 (0.01) | 3.842 (0.01) | 10.973 (0.02) | 10.655 (0.01) | 10.664 (0.02) | 10.557 (0.02) | 8.447 (0.02) |
| 30 × 5 × 8 | **0**(0.23) | 17.087 (0.02) | 16.799 (0.01) | 16.981 (0.03) | 13.074 (0.02) | 11.616 (0.02) | 13.051 (0.02) | 16.247 (0.02) | 16.899 (0.02) | 16.868 (0.02) | 16.741 (0.02) | 14.843 (0.03) |
| 30 × 8 × 3 | **0.127** (0.16) | 15.965 (0.01) | 7.32 (0.01) | 15.356 (0.01) | 4.015 (0.01) | 2.599 (0.01) | 3.452 (0.01) | 16.066 (0.01) | 15.982 (0.01) | 15.985 (0.01) | 15.554 (0.01) | 11.082 (0.01) |
| 30 × 8 × 5 | **0**(0.24) | 14.606 (0.02) | 10.306 (0.01) | 14.334 (0.02) | 6.126 (0.01) | 5.109 (0.01) | 5.92 (0.01) | 13.541 (0.01) | 13.79 (0.01) | 13.814 (0.02) | 13.486 (0.02) | 14.512 (0.02) |
| 30 × 8 × 8 | **0**(0.38) | 21.823 (0.03) | 18.169 (0.01) | 21.717 (0.03) | 15.614 (0.02) | 14.591 (0.02) | 15.515 (0.02) | 21.323 (0.02) | 21.394 (0.03) | 21.314 (0.03) | 21.269 (0.02) | 21.259 (0.03) |
| Mean | **0.273** (0.09) | 14.224 (0.01) | 10.228 (0.01) | 14.037 (0.01) | 8.571 (0.01) | 7.598 (0.01) | 8.197 (0.01) | 14.286 (0.01) | 14.240 (0.01) | 14.198 (0.01) | 14.042 (0.01) | 13.727 (0.01) |

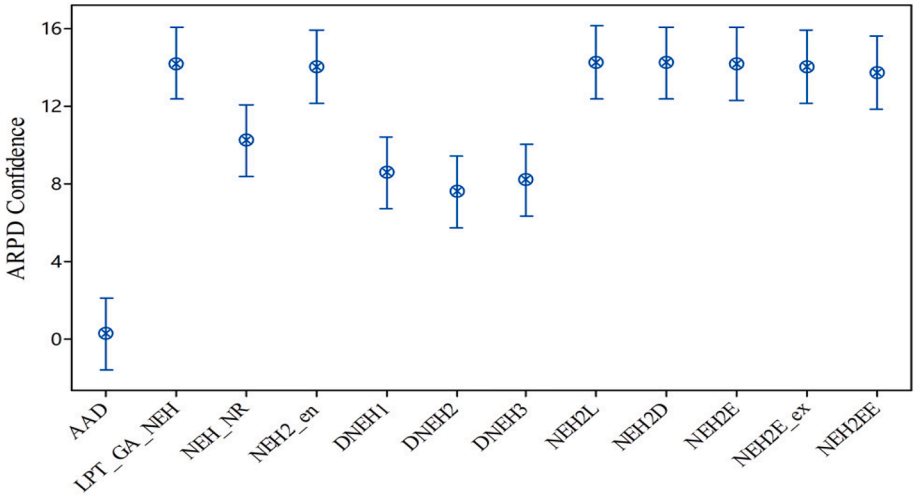



**Fig. 3.** Mean plot with confidence intervals at the 95% confidence level for the compared heuristics.

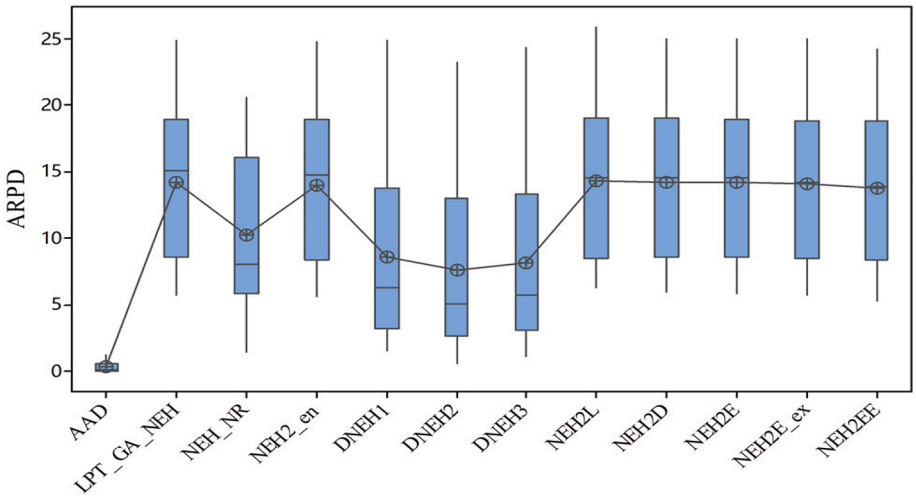



**Fig. 4.** Boxplot of the ARPD values.

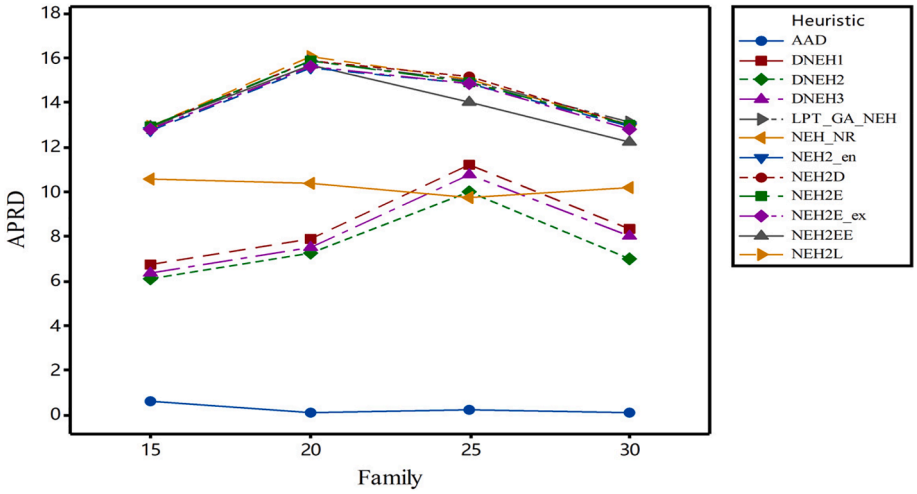



**Fig. 5.** Interaction plot between constructive heuristics and the number of families.

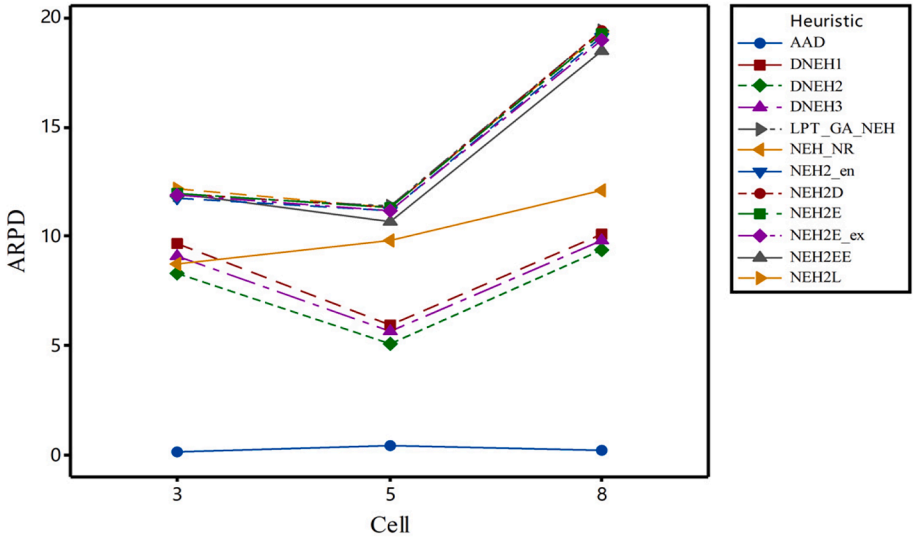



**Fig. 6.** Interaction plot between constructive heuristics and the number of cells.

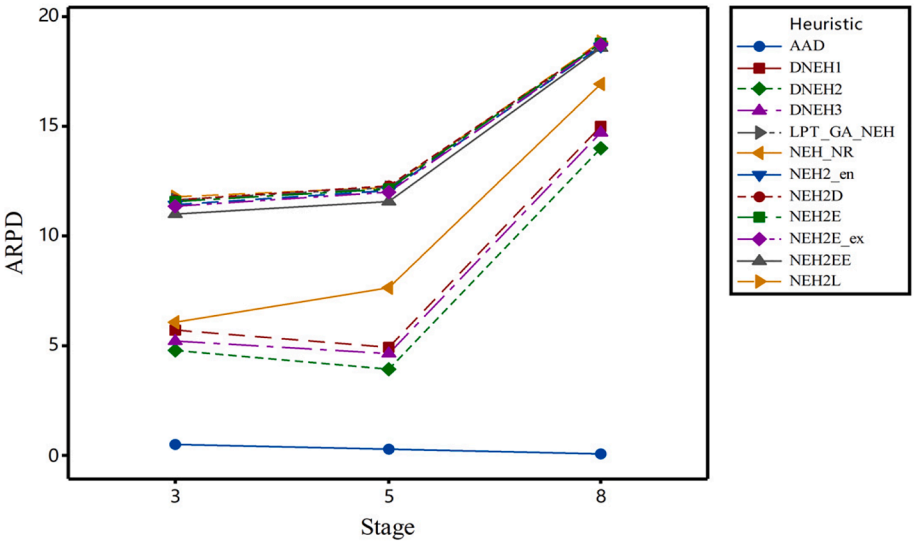
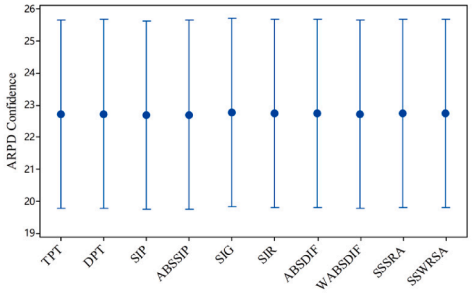


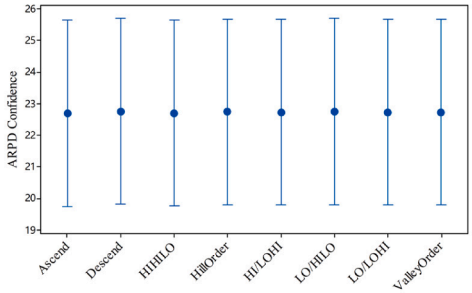

**Fig. 7.** Interaction plot between constructive heuristics and the number of stages.

NEH2D (Chen et al., 2021), NEH2E (Chen et al., 2021), NEH2E_ex (Chen et al., 2021) and NEH2EE (Chen et al., 2021). All of these heuristics are all proposed for solving the FSP variants, and have been introduced in Section 2. For LPT_GA_NEH, it cannot deal with the decision of cell connection. While for the other heuristics, they can only deal with the decisions of job sequencing and cell assignment, but cannot deal with the decisions of family sequencing and cell connection. These heuristics all contain the sequencing rule, and factory assignment rule. They are adjusted appropriately as follows. First, the sequencing rule is used to determine the job sequence in each family. Second, the factory assignment rule is used to determine the cell for each family. Finally, the sequencing rule is used to determine the family sequence on each cell. While for the cell connection, the families are assumed to be assigned to the same cell at the two flows. In addition to the constructive heuristics, the MILP model on the small instances is solved by IBM ILOG CPLEX 12.7.1 with the time limit set as 60 s.

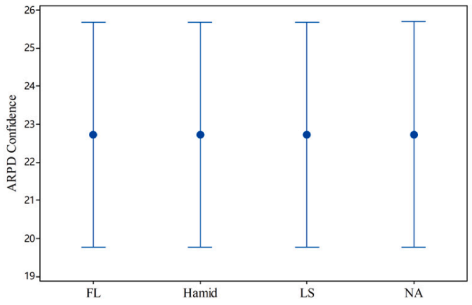
Table 5 shows the objective values and running time of the CPLEX and constructive heuristics when solving the 20 small-scale instances. It can be seen from Table 5 that CPLEX can solve only one instance, i.e., $3 \times 2 \times 4$, to optimality within 60 s. Regarding the objective values, when comparing CPLEX with the AAD, it can be found that CPLEX can get better values for the previous 10 instances. However, when the problem scales reach $5 \times 3 \times 3$ and larger, the AAD can get the best values for most instances. Regarding the running time, all of the constructive heuristics only take a very low time that is very close to 0 s. From the above observations, the effectiveness and efficiency of the AAD can be demonstrated. When comparing the AAD with the other heuristics, the AAD can get the best value for 19 out of 20 instances except $7 \times 2 \times 3$. Table 6 shows the average RPD (ARPD) values and the running time in seconds obtained by the constructive heuristics when solving the large-scale instances. The ARPD values of the given heuristic can be obtained by averaging the RPD values for solving the instances with the same problem scale. From Table 6, it can be observed that the AAD obtains much lower ARPD values for all of the problems. For 16 out of 36 problems, the AAD can even reach 0. The ARPD values and running time for different problems are averaged again to yield the mean values presented in the last row of Table 6. It becomes apparent that the AAD attains a value of 0.273, in stark contrast to the best of the other heuristics (DNEH2), which yields a value of 7.598. Grounded upon this observation, the AAD substantiates an advancement of approximately



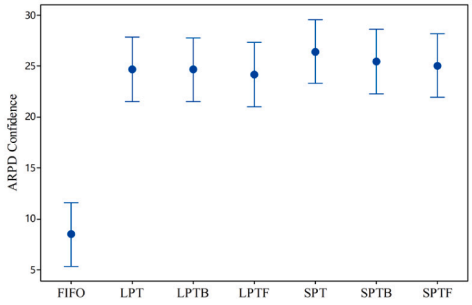

(a) Mean plot for the indicator calculation criteria.



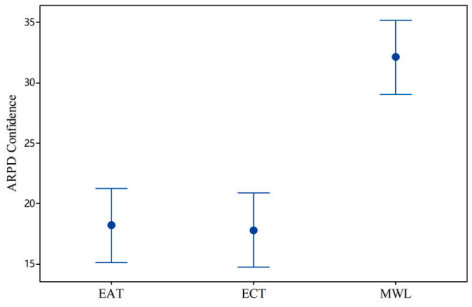

(b) Mean plot for the sorting criteria.



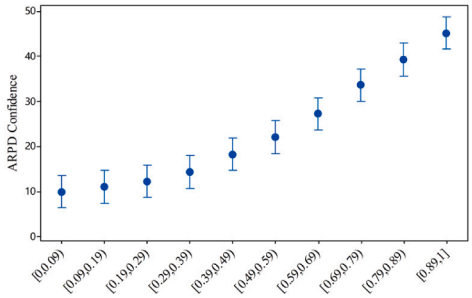

(c) Mean plot for the local search methods.




(d) Mean plot for the operation selection methods.




(e) Mean plot for the cell assignment methods.




(f) Mean plot for the non-delay factors.

**Fig. 8.** Mean plot with confidence intervals at the 95% confidence level for the six parameters.

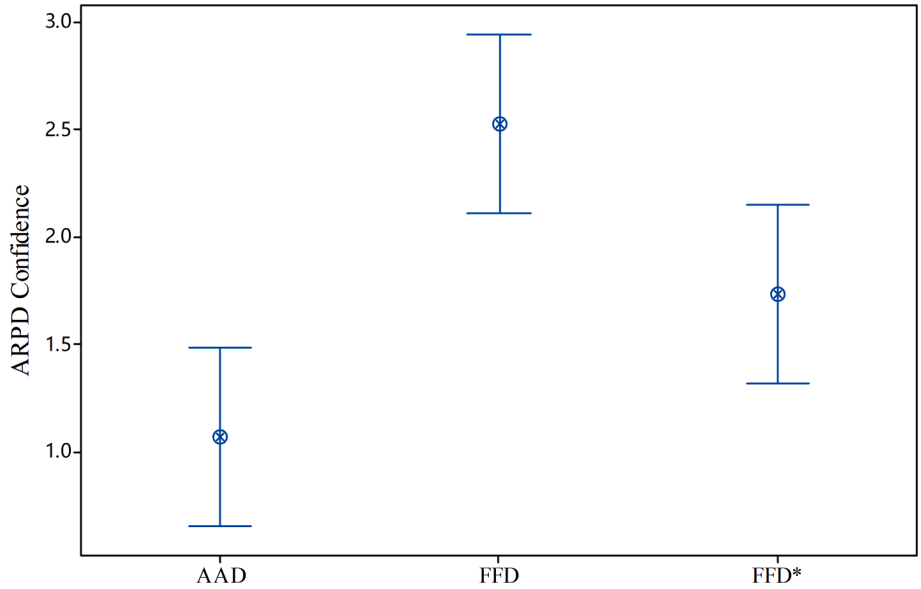



**Fig. 9.** Mean plot with confidence intervals at the 95% confidence level for the AAD, FFD and FFD*.

**Table 7**
ARPD values obtained by the AAD, FFD and FFD*.

| Problem | AAD | FFD | FFD* | Problem | AAD | FFD | FFD* |
|---|---|---|---|---|---|---|---|
| 15 × 3 × 3 | **0.989** | 2.032 | 2.205 | 25 × 3 × 3 | **0.427** | 1.411 | 0.557 |
| 15 × 3 × 5 | **0.929** | 2.273 | 1.673 | 25 × 3 × 5 | **0.597** | 0.775 | 0.877 |
| 15 × 3 × 8 | **1.346** | 3.307 | 2.319 | 25 × 3 × 8 | **0.634** | 1.252 | 1.191 |
| 15 × 5 × 3 | **0.935** | 4.553 | 1.660 | 25 × 5 × 3 | 0.853 | 3.913 | **0.773** |
| 15 × 5 × 5 | **0.606** | 4.564 | 2.240 | 25 × 5 × 5 | **0.962** | 2.097 | 1.168 |
| 15 × 5 × 8 | **0.357** | 2.297 | 2.258 | 25 × 5 × 8 | **0.216** | 2.456 | 0.593 |
| 15 × 8 × 3 | 2.022 | 5.550 | **1.843** | 25 × 8 × 3 | 1.873 | **1.062** | 2.055 |
| 15 × 8 × 5 | **3.783** | 5.593 | 5.847 | 25 × 8 × 5 | 1.726 | 2.790 | **1.345** |
| 15 × 8 × 8 | **0.577** | 5.562 | 0.821 | 25 × 8 × 8 | 2.314 | **0** | 3.377 |
| 20 × 3 × 3 | **0.550** | 1.215 | 1.548 | 30 × 3 × 3 | **0.278** | 1.535 | 0.599 |
| 20 × 3 × 5 | **0.396** | 1.592 | 1.107 | 30 × 3 × 5 | **0.637** | 1.325 | 0.814 |
| 20 × 3 × 8 | 1.829 | **0.717** | 2.063 | 30 × 3 × 8 | **0.184** | 0.552 | 0.311 |
| 20 × 5 × 3 | **0.864** | 2.699 | 2.299 | 30 × 5 × 3 | **0.666** | 2.368 | 1.414 |
| 20 × 5 × 5 | **0.466** | 3.011 | 0.914 | 30 × 5 × 5 | **0.402** | 3.493 | 0.931 |
| 20 × 5 × 8 | **0.824** | 2.478 | 2.038 | 30 × 5 × 8 | 1.516 | **0.342** | 1.755 |
| 20 × 8 × 3 | **0.421** | 4.863 | 2.819 | 30 × 8 × 3 | **0.555** | 3.127 | 1.199 |
| 20 × 8 × 5 | 3.292 | 1.810 | **1.706** | 30 × 8 × 5 | **0.538** | 5.437 | 1.147 |
| 20 × 8 × 8 | 3.227 | **1.960** | 5.486 | 30 × 8 × 8 | **0.700** | 0.876 | 1.345 |

28-fold in contrast to the best compared heuristic. These observations mean that for all the instances grouped under these problems, the AAD can get the best objective values. Notably, despite the AAD incurring a longer mean time of 0.09 s in comparison to DNEH2′s 0.01 s, the time investment remains inconspicuously modest. We contend that the huge improvement in precision of solutions can amply bridge the gap in time.

Additionally, to verify the statistical validity of the ARPD values, one-factor analysis of variance (ANOVA) (Zhang et al., 2021) where the heuristic type is considered as a single factor. Fig. 3 displays the mean plot with Tukey HSD intervals at the 95% confidence level. Note that if there is no overlap between the two intervals, it indicates that the data obtained by the given two heuristics are significantly different (Lu et al., 2023). We therefore can conclude that the AAD outperforms the other heuristics. Fig. 4 displays the box plot of the ARPD values to directly reflect the characteristics of the original data distribution. From Fig. 4, it can be easily observed that the ARPD values obtained by the AAD are very concentrated, and there is no exceptional value. Additionally, the upper bound of AAD is even smaller than the lower bounds of the others. All of these observations can verify the stability and effectiveness of the AAD. The running time tends to increase with increasing problem scale. Although the time consumed by the AAD is the longest, it is still much less than 1 s for all of the problems. Thus, considering its ARPD values, it can be concluded that the AAD shows outstanding performance at a very small computational penalty.

To show the performance of the constructive heuristics as the problem scale changes, Figs. 5-7 show the interaction plots between the constructive heuristics and the number of families, cells and stages, respectively. The increasing scale of families results in more operations being processed, while the increasing scale of cells increases the complexity of determining the cell to be assigned. And more stages mean more computational complexity. Thus, the problem becomes more complex when the number of families, cells and stages increases. It can be seen from Figs. 5-7 that the ARPD values obtained by the AAD change slightly with increasing problem scale, and they are all close to 0 and much smaller than those obtained by the other heuristics. Meanwhile, with the increase in the problem scale, the ARPD values obtained by the AAD have a downward trend. These observations indicate that as the problem scales increase, the AAD always performs excellently.

### 6.5. Evaluation of the AAD methodology

#### 6.5.1. Comparison with the FFD

Recall that as shown in Section 6.3, a total of 2052 configurations and 340 instances have been tuned in the tuning process of the AAD. To intuitively reflect the effectiveness and efficiency of the AAD methodology to conceive the automated heuristic, it is compared with the full

**Table 9**
Best elite configurations output by the AAD-F.

| Rank | Configurable parameter | | | | | |
|---|---|---|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
| 1 | *SIR* | *LO/HILO* | *Hamid* | *LPT* | *ECT* | 0.23 |
| 2 | *SIR* | *LO/HILO* | *Hamid* | *LPT* | *ECT* | 0.26 |
| 3 | *SIR* | *LO/HILO* | *LS* | *LPT* | *ECT* | 0.41 |
| 4 | *SIR* | *LO/HILO* | *Hamid* | *LPT* | *ECT* | 0.21 |
| 5 | *SIR* | *LO/HILO* | *LS* | *LPT* | *ECT* | 0 |
| 6 | *SIR* | *Hill* | *LS* | *LPTB* | *ECT* | 0 |
| 7 | *SIR* | *LO/HILO* | *LS* | *LPTB* | *ECT* | 0.30 |
| 8 | *SIR* | *LO/HILO* | *LS* | *LPTB* | *ECT* | 0 |
| 9 | *SIR* | *LO/HILO* | *LS* | *LPTB* | *ECT* | 0.62 |
| 10 | *SIR* | *LO/HILO* | *LS* | *LPTB* | *ECT* | 0.28 |

**Table 10**
ARPD values obtained by the AAD and AAD-F.

| Problem | AAD | AAD-F | Problem | AAD | AAD-F |
|---|---|---|---|---|---|
| 15 × 3 × 3 | **0.703** | 3.004 | 25 × 3 × 3 | 0.952 | **0.611** |
| 15 × 3 × 5 | **0.101** | 1.242 | 25 × 3 × 5 | **0.550** | 1.201 |
| 15 × 3 × 8 | **1.438** | 3.248 | 25 × 3 × 8 | **0.471** | 2.945 |
| 15 × 5 × 3 | **1.558** | 3.017 | 25 × 5 × 3 | **0.878** | 1.458 |
| 15 × 5 × 5 | **1.930** | 3.139 | 25 × 5 × 5 | **1.279** | 1.491 |
| 15 × 5 × 8 | **0.775** | 3.609 | 25 × 5 × 8 | **0.043** | 4.895 |
| 15 × 8 × 3 | 1.996 | **0.966** | 25 × 8 × 3 | **0.258** | 3.061 |
| 15 × 8 × 5 | **2.408** | 3.257 | 25 × 8 × 5 | 1.286 | **0.791** |
| 15 × 8 × 8 | **0.433** | 3.420 | 25 × 8 × 8 | **1.035** | 4.204 |
| 20 × 3 × 3 | **0.515** | 1.341 | 30 × 3 × 3 | **0.509** | 1.255 |
| 20 × 3 × 5 | **0.673** | 1.083 | 30 × 3 × 5 | **0.696** | 0.822 |
| 20 × 3 × 8 | **1.803** | 3.181 | 30 × 3 × 8 | **0.102** | 1.964 |
| 20 × 5 × 3 | **2.867** | 2.917 | 30 × 5 × 3 | **0.859** | 1.601 |
| 20 × 5 × 5 | **0.554** | 4.236 | 30 × 5 × 5 | **0.041** | 2.806 |
| 20 × 5 × 8 | **1.299** | 3.034 | 30 × 5 × 8 | **1.509** | 2.516 |
| 20 × 8 × 3 | **0.632** | 2.175 | 30 × 8 × 3 | 1.254 | **0.552** |
| 20 × 8 × 5 | 2.356 | **1.101** | 30 × 8 × 5 | **0.221** | 2.091 |
| 20 × 8 × 8 | **2.525** | 3.502 | 30 × 8 × 8 | **0.780** | 3.337 |

**Table 8**
Procedural parameters of the AAD and AAD-F.

| AAD variants | Race number | Limited budget | Number of test instances | Number of Candidate configurat-ions | Number of discarded configurat-ions | Number of remaining configurat-ions | Number of selected elite configurat-ions | Number of generated configurat-ions |
|---|---|---|---|---|---|---|---|---|
| AAD-F | 1 | 1500 | 20 | 600 | 128 | 472 | 10 | 456 |
| | 2 | 1631 | 40 | 466 | 81 | 385 | 10 | 378 |
| | 3 | 1748 | 40 | 388 | 39 | 349 | 10 | 364 |
| | 4 | 2058 | 50 | 374 | 1 | 373 | 10 | – |
| AAD | 1 | 1500 | 70 | 600 | 586 | 14 | 10 | 466 |
| | 2 | 1680 | 40 | 480 | 468 | 12 | 10 | 462 |
| | 3 | 2135 | 100 | 474 | 456 | 18 | 10 | 524 |
| | 4 | 2984 | 130 | 542 | 532 | 10 | 8 | – |

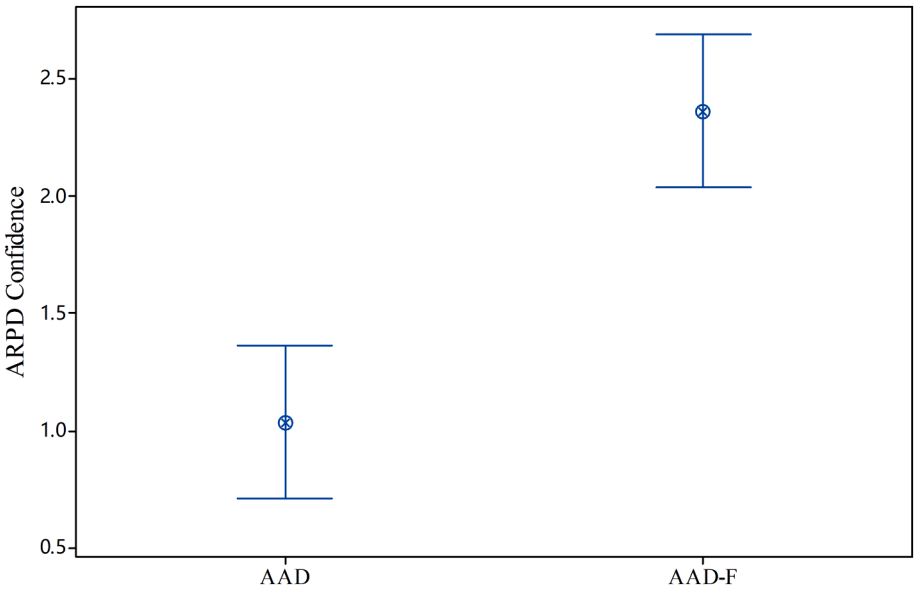



**Fig. 10.** Mean plot with confidence intervals at the 95% confidence level for the AAD and AAD-F.

factorial design (FFD) (Engin et al., 2011); (Yu et al., 2018) methodology. The FFD can help calibrate the algorithm and analyze the impact of each parameter value on the overall performance. A total of 67,200 $(10 \times 8 \times 4 \times 7 \times 3 \times 10)$ configurations are collected by combining each value of the six parameters. Note that for $X_6$, since it is the numerical parameter and its value space is from 0 to 1, we divide the whole space into 10 subspaces (i.e., [0,0.09), [0.09,0.19), etc.). In the design of the experiment, all the configurations are tested on all the large-scale problems, each of which selects one instance. The specific value of $X_6$ for a given configuration is randomly selected in the subspace. To show the impact of the parameter value, Tables A1-A5 in the Appendix A summarize the ARPD results for the six parameters, which are grouped under the same value for each parameter. Fig. 8 figures the mean plots with Tukey HSD intervals at the 95% confidence level to show the statistical validity.

From Tables A1-A3 and Fig. 8**(a – c)**, it is difficult to significantly identify the best indicator calculation criterion, sorting criterion, and local search method. We can see that the ARPD values for these parameters are relatively large. These results indicate that the performance of the configurations with the specific value of these parameters varies greatly. Regarding the cell assignment method, from Table A5 and Fig. 8 **(e)**, the *EAT* and *ECT* outperform the *MWL*. With regard to the non-delay factors, Table A6 and Fig. 8**(f)** show that with the increase of the non-delay factor value, the ARPD value shows a gradually increasing trend.

For the five aforementioned parameters, from Tables A1-A3, A5 and A6, in terms of the mean values, the lowest ones for these parameters are those obtained by *SIP*, *Ascend*, *LS*, *ECT*, and [0,0.09), respectively. These values are consistent with those of the best configuration obtained by AAD. While for the operation selection method, from Table A4 and Fig. 8 **(d)**, the *FIFO* is selected. However, the *FIFO* has not appeared in the elite configurations obtained by AAD as shown in Table 3. Thus, to show their performance directly, the AAD is compared to two configurations, i.e., FFD and FFD*. The FFD is composed of the parameter values that have the minimum mean ARPD values, while the FFD* is the configuration that has the minimum mean ARPD value over all the problems among the 67,200 configurations. Since it is difficult to list the computational results of all the configurations, the FFD* is directly given as $X_1 = SIG$, $X_2 = ValleyOrder$, $X_3 = FL$, $X_4 = LPT$, $X_5 = ECT$, and $X_6 \in [0, 0.09)$. Note that throughout the study for determining the FFD and FFD*, the specific value of $X_6$ is randomly selected in the subspace. While for the FFD and FFD* to be compared, $X_6$ is set to 0 as it occurs most frequently in Table 4.

Table 7 summarizes the ARPD values obtained by the AAD, FFD and FFD* on all the large-scale instances. It reveals that AAD outperforms FFD in 31 out of 36 problems, and surpasses FFD* in 32 out of 36 problems. Fig. 9 displays the mean plot for the AAD, FFD and FFD* using ANOVA, and shows that AAD performs significantly better than FFD. Through these observations, it can be concluded that the combination of the parameter values that have good comprehensive performance is not necessarily the best performer, and that the configuration performs well on some instances, but not necessarily in others. In addition, the number of configurations tuned in AAD is much smaller than that in FFD and FFD*. In summary, AAD has outstanding performance both in terms of effectiveness and efficiency.

#### 6.5.2. *Comparison with the classical I/F-Race*

In the AAD method, to determine the statistical results of the candidate configurations, the Wilcoxon rank-sum test is adopted instead of Friedman's test recommended in the classical I/F-Race. In order to verify the effectiveness of Wilcoxon rank-sum test, the method based on Friedman's test (noted as AAD-F) is implemented and compared with the employed method under the same total budget. Table 8 gives the procedural parameters in the process of the two AAD variants, where the data of AAD are also shown in Table 3. Table 9 displays the best elite configurations output by AAD-F. From Table 8, it can be seen that at each race, the AAD discards many more configurations than the AAD-F. This can verify that Wilcoxon rank-sum test has a better ability to identify the configurations that perform significantly worse. Benefiting from this, at each race of AAD, the number of surviving configurations decreases quickly. As shown in Table 8, the AAD can evaluate the surviving configurations on more instances. Consequently, the AAD can select the elite configurations that have better comprehensive performance.

To directly see the performance of the two best configurations output by the AAD and AAD-F, they are tested on the large-scale instances, and the ARPD values are summarized in Table 10. From the table, we can see that AAD performs best for 31 out of 36 problems. ANOVA is also employed to analyze their statistical results, and the mean plot for the AAD and AAD-F is displayed in Fig. 10. From the figure, it can be observed that the AAD performs significantly better than the AAD-F. These results verify that the AAD has a much better comprehensive performance, and thereby the effectiveness of the Wilcoxon rank-sum test is verified.

## 7. Conclusion

This paper studied an RDFGSP inspired by the production of PCBs in the SMT workshop. Its important features lie in the reconfigurability of the flowlines and the families with grouped jobs. To solve this problem, this paper first developed a configurable *meta*-algorithm considering the problem-specific characteristics. The *meta*-algorithm can be instantiated to a concrete constructive heuristic by assigning values to the parameters, i.e., sequencing rule, routing rule, dispatching rule, and non-delay factor. Each of these parameters has various values, and thus, it is difficult to determine the best combination. To this end, the I/F-Race, an AAD methodology, is employed to conceive the best constructive heuristic with minimum human intervention. Through extensive computational experiments, the effectiveness and efficiency of the AAD for conceiving the constructive heuristic by comparison with the full factorial design is demonstrated. Furthermore, the solution accuracy and efficiency of the generated heuristic are verified in solving small-scale problems by comparison with CPLEX and other heuristics. The effectiveness of the generated heuristic is demonstrated at a very small cost in time when solving large-scale problems.

An improvement heuristic starts with a complete initial solution, and improves the solution iteratively until a stopping criterion is met. It has been shown in the literature that the quality of construction heuristics can improve the performance of optimization algorithms by narrowing their search space or producing good initial solutions. Thus, our future study lies in extending the *meta*-algorithm by integrating the characteristics of the metaheuristics and conceiving an automated improvement heuristic.

## 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.

## Data availability

Data will be made available on request.

## Acknowledgments

This research is partially supported by National Natural Science Foundation of China (Grant No. 62303204, 52175490, 52205529), Natural Science Foundation of Shandong Province (Grant No. ZR2021QF036, ZR2021QE195), and Guangyue Young Scholar Innovation Team of Liaocheng University (Grant No. LCUGYTD2022-03).

## Appendix A

Tables A1-A6.

**Table A1**
ARPD values obtained by constructive heuristics with different indicator rules.

| Problem | Constructive heuristics with the fixed indicator rules | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | TPT | DPT | SIP | ABSSIP | SIG | SIR | ABSDIF | WABSDIF | SSSRA | SSWRSA |
| 15 × 3 × 3 | 11.769 | 11.672 | 11.537 | 11.52 | 11.636 | 11.517 | 11.76 | 11.744 | 11.712 | 11.718 |
| 15 × 3 × 5 | 14.566 | 14.456 | 14.46 | 14.414 | 14.551 | 14.385 | 14.314 | 14.528 | 14.48 | 14.414 |
| 15 × 3 × 8 | 25.631 | 25.73 | 25.481 | 25.568 | 25.718 | 25.619 | 25.644 | 25.529 | 25.88 | 25.604 |
| 15 × 5 × 3 | 18.92 | 18.964 | 19.06 | 19.102 | 19.126 | 19.025 | 19.061 | 19.049 | 19.093 | 19.092 |
| 15 × 5 × 5 | 27.616 | 27.511 | 27.571 | 27.562 | 27.764 | 27.398 | 27.6 | 27.621 | 27.565 | 27.464 |
| 15 × 5 × 8 | 32.131 | 32.188 | 32.266 | 32.376 | 32.552 | 32.538 | 32.039 | 31.896 | 32.192 | 32.459 |
| 15 × 8 × 3 | 26.614 | 26.5 | 26.598 | 27.017 | 26.568 | 26.412 | 26.425 | 26.654 | 26.867 | 26.749 |
| 15 × 8 × 5 | 30.408 | 30.772 | 30.938 | 30.861 | 30.935 | 31.173 | 30.877 | 30.907 | 31.03 | 30.739 |
| 15 × 8 × 8 | 45.314 | 45.186 | 44.979 | 45.327 | 44.798 | 45.28 | 44.872 | 44.591 | 45.079 | 45.455 |
| 20 × 3 × 3 | 10.983 | 10.92 | 11.007 | 11.156 | 11.142 | 10.945 | 11.16 | 11.009 | 11.119 | 11.064 |
| 20 × 3 × 5 | 16.429 | 16.41 | 16.38 | 16.169 | 16.163 | 16.457 | 16.426 | 16.299 | 16.448 | 16.429 |
| 20 × 3 × 8 | 20.617 | 20.543 | 20.697 | 20.714 | 20.687 | 20.68 | 20.841 | 20.717 | 20.576 | 20.637 |
| 20 × 5 × 3 | 15.354 | 15.058 | 15.402 | 15.328 | 15.393 | 15.305 | 15.411 | 15.349 | 15.386 | 15.263 |
| 20 × 5 × 5 | 22.238 | 21.984 | 22.149 | 22.112 | 22.197 | 22.157 | 22.399 | 22.037 | 22.304 | 22.074 |
| 20 × 5 × 8 | 27.008 | 27.226 | 26.921 | 26.954 | 27.166 | 26.98 | 27.106 | 27.388 | 27.088 | 27.134 |
| 20 × 8 × 3 | 23.193 | 23.051 | 22.901 | 23.007 | 23.356 | 23.223 | 22.996 | 22.993 | 23.061 | 23.141 |
| 20 × 8 × 5 | 29.03 | 29.203 | 29.058 | 29.188 | 29.072 | 29.003 | 28.774 | 28.867 | 28.858 | 28.918 |
| 20 × 8 × 8 | 43.023 | 43.176 | 43.012 | 42.931 | 43.083 | 42.936 | 43.297 | 43.184 | 43.066 | 42.883 |
| 25 × 3 × 3 | 9.508 | 9.653 | 9.533 | 9.585 | 9.507 | 9.556 | 9.6 | 9.606 | 9.592 | 9.592 |
| 25 × 3 × 5 | 12.476 | 12.502 | 12.553 | 12.44 | 12.566 | 12.37 | 12.54 | 12.527 | 12.623 | 12.505 |
| 25 × 3 × 8 | 20.821 | 20.972 | 20.843 | 20.944 | 20.872 | 20.941 | 20.976 | 20.916 | 20.934 | 20.904 |
| 25 × 5 × 3 | 15.655 | 15.482 | 15.697 | 15.704 | 15.686 | 15.868 | 15.792 | 15.734 | 15.766 | 15.822 |
| 25 × 5 × 5 | 19.784 | 19.735 | 19.694 | 19.606 | 19.671 | 19.628 | 19.814 | 19.846 | 19.825 | 19.648 |
| 25 × 5 × 8 | 31.921 | 31.959 | 31.764 | 31.96 | 31.927 | 31.759 | 32.18 | 31.999 | 31.992 | 32.097 |
| 25 × 8 × 3 | 19.62 | 19.429 | 19.592 | 19.365 | 19.803 | 19.604 | 19.865 | 19.69 | 19.556 | 19.646 |
| 25 × 8 × 5 | 26.551 | 26.531 | 26.299 | 26.281 | 26.578 | 26.48 | 26.721 | 26.416 | 26.456 | 26.36 |
| 25 × 8 × 8 | 34.971 | 35.036 | 35.115 | 34.9 | 35.122 | 35.076 | 34.869 | 35.225 | 34.918 | 34.921 |
| 30 × 3 × 3 | 9.002 | 9.042 | 8.958 | 8.871 | 8.968 | 8.997 | 8.891 | 8.802 | 8.943 | 8.949 |
| 30 × 3 × 5 | 13.373 | 13.27 | 13.381 | 13.27 | 13.455 | 13.278 | 13.314 | 13.229 | 13.432 | 13.32 |
| 30 × 3 × 8 | 20.746 | 20.91 | 20.984 | 20.93 | 20.934 | 20.993 | 20.886 | 20.914 | 20.875 | 20.904 |
| 30 × 5 × 3 | 13.867 | 13.893 | 13.929 | 13.84 | 13.941 | 13.98 | 13.898 | 13.703 | 13.825 | 13.84 |
| 30 × 5 × 5 | 19.029 | 19.148 | 19.067 | 19.005 | 19.001 | 19.155 | 18.928 | 19.021 | 18.857 | 18.857 |
| 30 × 5 × 8 | 30.346 | 30.442 | 30.137 | 30.346 | 30.391 | 30.293 | 30.614 | 30.72 | 30.323 | 30.305 |
| 30 × 8 × 3 | 19.842 | 20 | 19.968 | 19.841 | 19.805 | 20.002 | 19.801 | 19.894 | 19.763 | 20.134 |
| 30 × 8 × 5 | 23.782 | 23.739 | 23.568 | 23.519 | 23.74 | 23.935 | 23.769 | 23.602 | 23.499 | 23.734 |
| 30 × 8 × 8 | 35.645 | 35.86 | 35.523 | 35.479 | 35.675 | 35.99 | 35.462 | 35.523 | 35.624 | 36.029 |
| **Mean** | 22.716 | 22.726 | 22.695 | 22.700 | 22.765 | 22.748 | 22.748 | 22.715 | 22.739 | 22.745 |

**Table A2**
ARPD values obtained by constructive heuristics with different sorting rules.

| Problems | Constructive heuristics with the fixed sorting rules | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Ascend | Descend | HIHILO | HillOrder | HI/LOHI | LO/HILO | LO/LOHI | ValleyOrder |
| 15 × 3 × 3 | 11.808 | 11.586 | 11.626 | 11.565 | 11.761 | 11.662 | 11.517 | 11.565 |
| 15 × 3 × 5 | 14.433 | 14.426 | 14.362 | 14.558 | 14.43 | 14.398 | 14.451 | 14.493 |
| 15 × 3 × 8 | 25.831 | 25.478 | 25.602 | 25.762 | 25.81 | 25.698 | 25.498 | 25.561 |
| 15 × 5 × 3 | 18.882 | 19.141 | 18.852 | 19.037 | 19.037 | 19.364 | 19.119 | 19.056 |
| 15 × 5 × 5 | 27.603 | 27.602 | 27.336 | 27.794 | 27.627 | 27.512 | 27.57 | 27.737 |
| 15 × 5 × 8 | 32.308 | 32.13 | 32.359 | 32.462 | 32.247 | 32.174 | 32.231 | 32.385 |
| 15 × 8 × 3 | 26.487 | 26.62 | 26.326 | 26.941 | 26.599 | 26.707 | 26.787 | 26.631 |
| 15 × 8 × 5 | 30.891 | 30.631 | 31.125 | 30.692 | 31.344 | 30.489 | 30.366 | 30.999 |
| 15 × 8 × 8 | 44.487 | 45.327 | 45.653 | 45.56 | 45.21 | 45.221 | 44.899 | 44.285 |
| 20 × 3 × 3 | 11.055 | 10.977 | 11.024 | 11.02 | 11.154 | 11.005 | 11.129 | 11.112 |
| 20 × 3 × 5 | 16.291 | 16.404 | 16.355 | 16.245 | 16.363 | 16.383 | 16.557 | 16.39 |
| 20 × 3 × 8 | 20.785 | 20.522 | 20.673 | 20.683 | 20.688 | 20.552 | 20.546 | 20.74 |
| 20 × 5 × 3 | 15.313 | 15.165 | 15.264 | 15.344 | 15.481 | 15.427 | 15.403 | 15.334 |
| 20 × 5 × 5 | 22.034 | 22.298 | 21.989 | 22.098 | 22.059 | 22.363 | 22.141 | 22.378 |
| 20 × 5 × 8 | 26.783 | 27.315 | 27.09 | 27.201 | 27.27 | 27.246 | 26.981 | 26.873 |
| 20 × 8 × 3 | 23.029 | 22.92 | 23.123 | 22.991 | 23.095 | 23.122 | 23.142 | 23.296 |
| 20 × 8 × 5 | 29.273 | 29.095 | 28.853 | 28.823 | 28.689 | 29.182 | 29.202 | 28.891 |
| 20 × 8 × 8 | 42.787 | 43.285 | 42.939 | 42.967 | 43.326 | 42.919 | 43.058 | 43.238 |
| 25 × 3 × 3 | 9.574 | 9.618 | 9.649 | 9.54 | 9.559 | 9.638 | 9.487 | 9.568 |
| 25 × 3 × 5 | 12.515 | 12.565 | 12.472 | 12.606 | 12.559 | 12.547 | 12.484 | 12.534 |
| 25 × 3 × 8 | 20.904 | 20.929 | 20.811 | 20.847 | 20.876 | 20.968 | 20.919 | 20.934 |
| 25 × 5 × 3 | 15.769 | 15.604 | 15.674 | 15.722 | 15.601 | 15.863 | 15.75 | 15.853 |
| 25 × 5 × 5 | 19.677 | 19.61 | 19.792 | 19.819 | 19.922 | 19.722 | 19.719 | 19.528 |
| 25 × 5 × 8 | 31.771 | 32.104 | 31.914 | 31.972 | 31.913 | 31.996 | 32.077 | 31.909 |
| 25 × 8 × 3 | 19.329 | 19.609 | 19.828 | 19.834 | 19.702 | 19.7 | 19.579 | 19.449 |
| 25 × 8 × 5 | 26.352 | 26.516 | 26.451 | 26.562 | 26.384 | 26.565 | 26.438 | 26.322 |
| 25 × 8 × 8 | 35.049 | 35.41 | 34.82 | 34.963 | 34.773 | 35.274 | 35.024 | 34.945 |
| 30 × 3 × 3 | 9.105 | 8.931 | 8.858 | 8.921 | 8.87 | 8.944 | 8.994 | 8.961 |
| 30 × 3 × 5 | 13.441 | 13.432 | 13.269 | 13.324 | 13.213 | 13.207 | 13.386 | 13.392 |
| 30 × 3 × 8 | 20.968 | 21.028 | 20.792 | 20.93 | 20.72 | 20.909 | 20.986 | 20.867 |
| 30 × 5 × 3 | 13.803 | 13.861 | 13.944 | 14.036 | 13.805 | 13.855 | 13.866 | 13.776 |
| 30 × 5 × 5 | 18.937 | 19.241 | 19.162 | 18.944 | 18.914 | 18.954 | 18.834 | 19.067 |
| 30 × 5 × 8 | 30.322 | 30.47 | 30.356 | 30.35 | 30.448 | 30.384 | 30.296 | 30.5 |
| 30 × 8 × 3 | 19.799 | 20.05 | 19.81 | 19.637 | 19.891 | 19.879 | 20.219 | 20.077 |
| 30 × 8 × 5 | 23.843 | 23.734 | 23.708 | 23.557 | 23.488 | 23.62 | 23.78 | 23.822 |
| 30 × 8 × 8 | 35.902 | 35.654 | 35.554 | 35.271 | 35.455 | 35.64 | 35.825 | 35.985 |
| **Mean** | 22.698 | 22.758 | 22.706 | 22.738 | 22.730 | 22.752 | 22.729 | 22.735 |

**Table A3**
ARPD values obtained by constructive heuristics with different local search rules.

| Problem | Constructive heuristics with the fixed local search methods | | | |
|---|---|---|---|---|
| | FL | Hamid | LS | NA |
| 15 × 3 × 3 | 11.632 | 11.587 | 11.63 | 11.749 |
| 15 × 3 × 5 | 14.48 | 14.47 | 14.45 | 14.441 |
| 15 × 3 × 8 | 25.687 | 25.632 | 25.626 | 25.668 |
| 15 × 5 × 3 | 19.077 | 19.104 | 19 | 18.961 |
| 15 × 5 × 5 | 27.613 | 27.575 | 27.57 | 27.59 |
| 15 × 5 × 8 | 32.342 | 32.331 | 32.274 | 32.2 |
| 15 × 8 × 3 | 26.62 | 26.544 | 26.56 | 26.73 |
| 15 × 8 × 5 | 30.859 | 30.738 | 30.937 | 30.768 |
| 15 × 8 × 8 | 44.94 | 45.17 | 44.986 | 45.174 |
| 20 × 3 × 3 | 11.089 | 11.068 | 11.043 | 11.002 |
| 20 × 3 × 5 | 16.328 | 16.406 | 16.432 | 16.344 |
| 20 × 3 × 8 | 20.664 | 20.595 | 20.75 | 20.646 |
| 20 × 5 × 3 | 15.371 | 15.336 | 15.327 | 15.333 |
| 20 × 5 × 5 | 22.304 | 22.157 | 22.124 | 22.166 |
| 20 × 5 × 8 | 27.063 | 27.122 | 27.221 | 26.909 |
| 20 × 8 × 3 | 23.072 | 23.064 | 23.109 | 23.132 |
| 20 × 8 × 5 | 28.98 | 29.046 | 28.926 | 29.107 |
| 20 × 8 × 8 | 43.032 | 43.053 | 43.093 | 43.105 |
| 25 × 3 × 3 | 9.576 | 9.579 | 9.555 | 9.595 |
| 25 × 3 × 5 | 12.591 | 12.507 | 12.488 | 12.486 |
| 25 × 3 × 8 | 20.85 | 20.92 | 20.927 | 20.887 |
| 25 × 5 × 3 | 15.777 | 15.756 | 15.734 | 15.668 |
| 25 × 5 × 5 | 19.675 | 19.674 | 19.71 | 19.798 |
| 25 × 5 × 8 | 31.893 | 31.94 | 31.977 | 32.024 |
| 25 × 8 × 3 | 19.642 | 19.669 | 19.661 | 19.448 |

(*continued on next page*)

**Table A3** (*continued*)

| Problem | Constructive heuristics with the fixed local search methods | | | |
|---|---|---|---|---|
| | FL | Hamid | LS | NA |
| 25 × 8 × 5 | 26.483 | 26.538 | 26.531 | 26.478 |
| 25 × 8 × 8 | 34.998 | 35.058 | 34.942 | 35.068 |
| 30 × 3 × 3 | 8.901 | 8.968 | 8.923 | 8.982 |
| 30 × 3 × 5 | 13.325 | 13.335 | 13.29 | 13.366 |
| 30 × 3 × 8 | 20.924 | 20.868 | 20.879 | 20.949 |
| 30 × 5 × 3 | 13.912 | 13.868 | 13.868 | 13.85 |
| 30 × 5 × 5 | 19.06 | 18.97 | 18.958 | 19.065 |
| 30 × 5 × 8 | 30.383 | 30.376 | 30.468 | 30.401 |
| 30 × 8 × 3 | 19.899 | 19.986 | 19.897 | 19.892 |
| 30 × 8 × 5 | 23.715 | 23.632 | 23.697 | 23.826 |
| 30 × 8 × 8 | 35.61 | 35.678 | 35.631 | 35.681 |
| **Mean** | 22.732 | 22.731 | **21.673** | 21.674 |

**Table A4**
ARPD values obtained by constructive heuristics with different operation selection methods.

| Problems | Constructive heuristics with the fixed operation selection methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | FIFO | LPT | LPTB | LPTF | SPT | SPTB | SPTF |
| 15 × 3 × 3 | 6.877 | 11.386 | 11.445 | 13.457 | 13.104 | 13.056 | 11.996 |
| 15 × 3 × 5 | 6.472 | 15.493 | 15.139 | 14.802 | 19.568 | 16.712 | 13.084 |
| 15 × 3 × 8 | 6.547 | 28.172 | 29.154 | 25.963 | 30.43 | 29.981 | 29.4 |
| 15 × 5 × 3 | 11.782 | 21.093 | 21.318 | 20.406 | 21.132 | 18.36 | 19.26 |
| 15 × 5 × 5 | 11.289 | 28.011 | 28.117 | 28.123 | 33.221 | 32.739 | 31.406 |
| 15 × 5 × 8 | 10.286 | 35.257 | 34.852 | 34.736 | 37.484 | 36.199 | 37.165 |
| 15 × 8 × 3 | 15.439 | 26.535 | 24.912 | 26.386 | 31.394 | 30.207 | 31.12 |
| 15 × 8 × 5 | 12.849 | 31.09 | 31.3 | 31.473 | 37.28 | 35.906 | 35.754 |
| 15 × 8 × 8 | 14.873 | 47.776 | 50.858 | 47.175 | 52.814 | 51.333 | 50.361 |
| 20 × 3 × 3 | 5.482 | 14.022 | 12.731 | 13.8 | 11.135 | 10.672 | 9.397 |
| 20 × 3 × 5 | 5.332 | 17.753 | 18.801 | 18.626 | 18.899 | 16.551 | 18.536 |
| 20 × 3 × 8 | 5.252 | 20.928 | 20.231 | 20.21 | 29.288 | 26.537 | 22.1 |
| 20 × 5 × 3 | 8.513 | 15.476 | 15.698 | 15.94 | 16.463 | 17.638 | 17.611 |
| 20 × 5 × 5 | 9.145 | 23.016 | 24.372 | 23.618 | 24.951 | 26.227 | 23.705 |
| 20 × 5 × 8 | 7.573 | 31.657 | 31.877 | 27.024 | 33.17 | 28.716 | 29.824 |
| 20 × 8 × 3 | 13.386 | 23.833 | 24.422 | 23.381 | 26.224 | 25.408 | 25.227 |
| 20 × 8 × 5 | 13.222 | 29.213 | 28.996 | 27.148 | 35.285 | 34.784 | 34.079 |
| 20 × 8 × 8 | 13.036 | 47.039 | 44.21 | 46.058 | 50.354 | 50.756 | 50.381 |
| 25 × 3 × 3 | 4.543 | 9.31 | 9.211 | 8.859 | 11.212 | 12.203 | 11.764 |
| 25 × 3 × 5 | 4.264 | 13.541 | 14.534 | 13.899 | 14.083 | 13.965 | 13.478 |
| 25 × 3 × 8 | 4.357 | 23.933 | 22.099 | 26.31 | 27.053 | 21.561 | 20.925 |
| 25 × 5 × 3 | 8.062 | 17.678 | 17.059 | 16.843 | 17.399 | 16.14 | 16.814 |
| 25 × 5 × 5 | 7.361 | 22.697 | 23.158 | 22.415 | 21.878 | 20.238 | 20.266 |
| 25 × 5 × 8 | 7.658 | 36.501 | 36.824 | 33.102 | 37.007 | 36.061 | 36.608 |
| 25 × 8 × 3 | 10.858 | 20.324 | 20.388 | 20.637 | 24.162 | 21.044 | 20.175 |
| 25 × 8 × 5 | 10.961 | 28.607 | 30.166 | 29.164 | 29.776 | 28.264 | 28.622 |
| 25 × 8 × 8 | 9.538 | 41.265 | 40.982 | 40.084 | 39.754 | 35.549 | 37.518 |
| 30 × 3 × 3 | 4.056 | 10.379 | 10.433 | 9.309 | 9.949 | 9.379 | 8.978 |
| 30 × 3 × 5 | 4.41 | 13.709 | 15.165 | 13.526 | 15.125 | 14.871 | 16.367 |
| 30 × 3 × 8 | 4.227 | 24.549 | 25.135 | 23.584 | 22.946 | 22.712 | 22.96 |
| 30 × 5 × 3 | 6.774 | 15.86 | 16.239 | 15.311 | 15.222 | 13.844 | 13.803 |
| 30 × 5 × 5 | 6.807 | 21.117 | 20.479 | 21.27 | 20.749 | 22.281 | 20.764 |
| 30 × 5 × 8 | 6.234 | 34.905 | 33.407 | 33.481 | 32.936 | 36.238 | 35.582 |
| 30 × 8 × 3 | 9.965 | 21.653 | 21.557 | 21.376 | 22.359 | 21.17 | 21.208 |
| 30 × 8 × 5 | 9.098 | 25.349 | 24.374 | 25.714 | 25.586 | 27.845 | 27.792 |
| 30 × 8 × 8 | 8.997 | 40.066 | 38.897 | 38.258 | 42.854 | 41.815 | 39.274 |
| **Mean** | **8.487** | 24.700 | 24.682 | 24.207 | 26.451 | 25.471 | 25.092 |

**Table A5**
ARPD values obtained by constructive heuristics with different route rules.

| Problems | Constructive heuristics with the fixed route rules | | |
|---|---|---|---|
| | EAT | ECT | MWL |
| 15 × 3 × 3 | 8.494 | 8.267 | 17.977 |
| 15 × 3 × 5 | 12.127 | 11.854 | 19.448 |
| 15 × 3 × 8 | 22.624 | 22.466 | 31.953 |
| 15 × 5 × 3 | 14.691 | 14.114 | 28.39 |
| 15 × 5 × 5 | 22.965 | 22.421 | 37.363 |
| 15 × 5 × 8 | 26.387 | 26.254 | 44.112 |
| 15 × 8 × 3 | 20.296 | 19.79 | 39.596 |
| 15 × 8 × 5 | 25.231 | 24.51 | 42.638 |
| 15 × 8 × 8 | 38.598 | 38.026 | 58.682 |
| 20 × 3 × 3 | 8.303 | 7.826 | 17.064 |
| 20 × 3 × 5 | 13.7 | 13.461 | 21.903 |
| 20 × 3 × 8 | 17.799 | 17.581 | 26.498 |
| 20 × 5 × 3 | 11.087 | 11 | 23.954 |
| 20 × 5 × 5 | 17.671 | 17.169 | 31.558 |
| 20 × 5 × 8 | 22.134 | 21.646 | 37.525 |
| 20 × 8 × 3 | 17.19 | 16.785 | 35.337 |
| 20 × 8 × 5 | 22.2 | 21.954 | 42.571 |
| 20 × 8 × 8 | 36.142 | 36.071 | 57.326 |
| 25 × 3 × 3 | 7.058 | 6.902 | 14.748 |
| 25 × 3 × 5 | 10.209 | 9.855 | 17.604 |
| 25 × 3 × 8 | 18.255 | 18.112 | 26.302 |
| 25 × 5 × 3 | 11.432 | 10.974 | 24.863 |
| 25 × 5 × 5 | 15.17 | 14.552 | 29.449 |
| 25 × 5 × 8 | 27.305 | 26.792 | 41.59 |
| 25 × 8 × 3 | 14.105 | 13.624 | 31.289 |
| 25 × 8 × 5 | 19.991 | 19.727 | 39.636 |
| 25 × 8 × 8 | 28.228 | 27.706 | 48.823 |
| 30 × 3 × 3 | 6.735 | 6.287 | 13.684 |
| 30 × 3 × 5 | 10.777 | 10.414 | 18.848 |
| 30 × 3 × 8 | 17.716 | 17.372 | 27.718 |
| 30 × 5 × 3 | 9.5 | 9.309 | 22.787 |
| 30 × 5 × 5 | 14.793 | 14.23 | 27.975 |
| 30 × 5 × 8 | 25.935 | 25.583 | 39.783 |
| 30 × 8 × 3 | 14.473 | 13.751 | 31.381 |
| 30 × 8 × 5 | 17.764 | 17.267 | 35.969 |
| 30 × 8 × 8 | 28.94 | 28.374 | 49.927 |
| **Mean** | 18.223 | **17.834** | 32.119 |

**Table A6**
ARPD values obtained by constructive heuristics with different non-delay factors.

| Problem | Constructive heuristics with the fixed indicator rules | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | [0,0.09) | [0.09,0.19) | [0.19,0.29) | [0.29,0.39) | [0.39,0.49) | [0.49,0.59) | [0.59,0.69) | [0.69,0.79) | [0.79,0.89) | [0.89,1] |
| 15 × 3 × 3 | 18.062 | 16.884 | 16.453 | 18.316 | 21.952 | 24.663 | 28.601 | 30.721 | 38.202 | 49.01 |
| 15 × 3 × 5 | 14.518 | 18.301 | 16.453 | 16.365 | 20.656 | 26.933 | 33.747 | 41.786 | 48.796 | 57.344 |
| 15 × 3 × 8 | 13.006 | 22.102 | 27.731 | 32.006 | 42.06 | 50.357 | 58.236 | 67.048 | 74.445 | 76.931 |
| 15 × 5 × 3 | 6.514 | 6.613 | 6.723 | 7.135 | 7.998 | 9.459 | 11.71 | 12.932 | 17.768 | 23.923 |
| 15 × 5 × 5 | 6.93 | 7.178 | 7.386 | 8.652 | 10.524 | 13.336 | 15.235 | 23.954 | 30.893 | 33.686 |
| 15 × 5 × 8 | 6.748 | 7.454 | 7.571 | 11.607 | 14.946 | 17.936 | 26.713 | 35.184 | 39.896 | 40.428 |
| 15 × 8 × 3 | 8.103 | 8.147 | 8.973 | 9.179 | 10.289 | 13.264 | 15.882 | 18.775 | 24.884 | 32.661 |
| 15 × 8 × 5 | 9.841 | 10.375 | 10.833 | 12.654 | 17.543 | 21.164 | 23.453 | 31.111 | 37.456 | 44.474 |
| 15 × 8 × 8 | 8.019 | 9.978 | 13.263 | 15.347 | 22.093 | 30.988 | 32.136 | 38.631 | 45.703 | 52.645 |
| 20 × 3 × 3 | 12.827 | 14.714 | 15.728 | 17.29 | 18.748 | 19.506 | 24.499 | 30.541 | 36.457 | 42.22 |
| 20 × 3 × 5 | 16.239 | 14.54 | 16.506 | 20.851 | 24.239 | 25.911 | 32.675 | 42.803 | 44.96 | 52.6 |
| 20 × 3 × 8 | 18.831 | 23.701 | 23.379 | 28.574 | 40.167 | 45.605 | 50.766 | 58.887 | 71.718 | 72.521 |
| 20 × 5 × 3 | 4.984 | 4.789 | 4.925 | 5.152 | 6.219 | 6.893 | 9.665 | 11.614 | 16.77 | 20.859 |
| 20 × 5 × 5 | 4.718 | 5.045 | 6.063 | 6.831 | 8.132 | 9.155 | 14.059 | 20.882 | 21.92 | 28.655 |
| 20 × 5 × 8 | 5.727 | 5.814 | 8.476 | 10.173 | 15.976 | 18.902 | 30.872 | 35.054 | 36.638 | 42.106 |
| 20 × 8 × 3 | 8.356 | 8.548 | 9.216 | 10.054 | 10.514 | 13.957 | 16.611 | 21.283 | 27.513 | 34.684 |
| 20 × 8 × 5 | 8.504 | 8.475 | 9.375 | 11.404 | 14.66 | 15.677 | 20.295 | 30.779 | 37.774 | 40.221 |
| 20 × 8 × 8 | 8.332 | 10.585 | 13.8 | 20.074 | 26.488 | 32.229 | 44.308 | 51.374 | 54.716 | 61.539 |
| 25 × 3 × 3 | 11.066 | 11.082 | 11.269 | 12.248 | 15.912 | 18.984 | 21.573 | 26.83 | 30.963 | 38.86 |
| 25 × 3 × 5 | 11.89 | 12.841 | 14.586 | 16.659 | 19.553 | 22.866 | 30.764 | 39.462 | 43.785 | 52.407 |
| 25 × 3 × 8 | 13.709 | 12.565 | 18.978 | 24.489 | 25.665 | 37.991 | 41.832 | 52.081 | 60.648 | 62.146 |
| 25 × 5 × 3 | 4.574 | 4.738 | 4.714 | 4.999 | 6.261 | 7.284 | 8.495 | 13.024 | 14.588 | 21.413 |
| 25 × 5 × 5 | 5.183 | 5.45 | 5.713 | 6.709 | 9.78 | 10.777 | 15.068 | 20.508 | 25.218 | 29.983 |
| 25 × 5 × 8 | 5.415 | 6.564 | 6.956 | 11.521 | 14.794 | 19.065 | 25.151 | 33.846 | 35.29 | 41.201 |
| 25 × 8 × 3 | 7.309 | 7.215 | 7.54 | 8.784 | 9.009 | 11.86 | 15.633 | 17.979 | 23.336 | 31.109 |

(*continued on next page*)

**Table A6** (*continued*)

| Problem | Constructive heuristics with the fixed indicator rules | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | [0,0.09) | [0.09,0.19) | [0.19,0.29) | [0.29,0.39) | [0.39,0.49) | [0.49,0.59) | [0.59,0.69) | [0.69,0.79) | [0.79,0.89) | [0.89,1] |
| 25 × 8 × 5 | 7.314 | 7.835 | 8.428 | 11.336 | 12.694 | 17.557 | 19.586 | 30.937 | 35.49 | 41.312 |
| 25 × 8 × 8 | 8.449 | 8.464 | 13.773 | 14.455 | 22.696 | 33.568 | 41.271 | 48.072 | 56.222 | 60.125 |
| 30 × 3 × 3 | 10.541 | 10.454 | 11.34 | 12.591 | 15.572 | 17.042 | 22.372 | 25.56 | 30.115 | 38.22 |
| 30 × 3 × 5 | 10.337 | 10.968 | 11.624 | 12.586 | 18.217 | 20.377 | 24.637 | 31.189 | 43.259 | 47.961 |
| 30 × 3 × 8 | 11.353 | 12.864 | 15.255 | 19.128 | 31.058 | 33.639 | 50.31 | 56.749 | 61.157 | 69.14 |
| 30 × 5 × 3 | 9.686 | 10.467 | 11.744 | 13.860 | 17.577 | 21.268 | 26.494 | 33.021 | 38.292 | 43.917 |
| 30 × 5 × 5 | 18.062 | 16.884 | 16.453 | 18.316 | 21.952 | 24.663 | 28.601 | 30.721 | 38.202 | 49.01 |
| 30 × 5 × 8 | 14.518 | 18.301 | 16.453 | 16.365 | 20.656 | 26.933 | 33.747 | 41.786 | 48.796 | 57.344 |
| 30 × 8 × 3 | 13.006 | 22.102 | 27.731 | 32.006 | 42.06 | 50.357 | 58.236 | 67.048 | 74.445 | 76.931 |
| 30 × 8 × 5 | 6.514 | 6.613 | 6.723 | 7.135 | 7.998 | 9.459 | 11.71 | 12.932 | 17.768 | 23.923 |
| 30 × 8 × 8 | 6.93 | 7.178 | 7.386 | 8.652 | 10.524 | 13.336 | 15.235 | 23.954 | 30.893 | 33.686 |
| **Mean** | **6.748** | 7.454 | 7.571 | 11.607 | 14.946 | 17.936 | 26.713 | 35.184 | 39.896 | 40.428 |

## References


Abedinnia, H., Glock, C.H., Brill, A., 2016. New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem. Computers & Operations Research 74, 165–174.

Bai, D., Bai, X., Li, H., Pan, Q.-k., Wu, C.-C., Gao, L., Guo, M., Lin, L., 2022. Blocking flowshop scheduling problems with release dates. Swarm and Evolutionary Computation 74, 101140.

Balaprakash P., Birattari M., & Stützle T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In Hybrid Metaheuristics: 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings 4 (pp. 108-122). Springer Berlin Heidelberg.

Bezerra, L.C.T., López-Ibáñez, M., Stützle, T., 2020. Automatically designing state-of-the-art multi-and many-objective evolutionary algorithms. Evolutionary computation 28 (2), 195–226.

Birattari M., Stützle T., Paquete L., & Varrentrapp K. (2002, July). A Racing Algorithm for Configuring Metaheuristics. In Gecco (Vol. 2, No. 2002).

Birattari, M., Kacprzyk, J., 2009. Tuning metaheuristics: a machine learning perspective, 197. Springer, Berlin.

Bistaffa, F., Blum, C., Cerquides, J., Farinelli, A., Rodríguez-Aguilar, J.A., 2019. A computational approach to quantify the benefits of ridesharing for policy makers and travellers. IEEE Transactions on Intelligent Transportation Systems 22 (1), 119–130.

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A heuristic algorithm for the n job, m machine sequencing problem. Management science 16 (10), B-630–B-637.

Chen, S., Pan, Q.-K., Gao, L., 2021. Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm. Robotics and Computer-Integrated Manufacturing 71, 102155.

Dong, X., Huang, H., Chen, P., 2008. An improved NEH-based heuristic for the permutation flowshop problem. Computers & Operations Research 35 (12), 3962–3968.

Engin, O., Ceran, G., Yilmaz, M.K., 2011. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. Applied Soft Computing 11 (3), 3056–3065.

Framinan, J.M., Leisten, R., 2003. An efficient constructive heuristic for flowtime minimisation in permutation flow shops. Omega 31 (4), 311–317.

Framinan, J.M., Leisten, R., Rajendran, C., 2003. Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. International Journal of Production Research 41 (1), 121–148.

Framinan, J.M., Gupta, J.N., Leisten, R., 2004. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. Journal of the Operational Research Society 55 (12), 1243–1255.

Gupta, J.N., 1971. A functional heuristic algorithm for the flowshop scheduling problem. Journal of the Operational Research Society 22 (1), 39–47.

Hamzadayı, A., 2020. An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem. Computers & Operations Research 123, 105006.

Huang, C., Li, Y., Yao, X., 2019. A survey of automatic parameter tuning methods for metaheuristics. IEEE transactions on evolutionary computation 24 (2), 201–216.

Huang, B.o., Zhou, MengChu, 2022. Symbolic scheduling of robotic cellular manufacturing systems with timed petri nets. IEEE Transactions on Control Systems Technology 30 (5), 1876–1887.

Kai, L., Jc, A., Hong, F.A., Zj, C., Jw, A., 2020. Parallel machine scheduling with position-based deterioration and learning effects in an uncertain manufacturing system. Computers & Industrial Engineering 149, 106858.

Khare, A., Agrawal, S., 2021. Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem. International Journal of Production Research 59 (23), 7266–7282.

Laha, D., Sarin, S.C., 2009. A heuristic to minimize total flow time in permutation flow shop. Omega 37 (3), 734–739.

Liao, T., de Oca, M.A.M., Stützle, T., 2013. Computational results for an automatically tuned CMA-ES with increasing population size on the CEC'05 benchmark set. Soft Computing 17 (6), 1031–1046.

Liao, T., Molina, D., Stützle, T., 2015. Performance evaluation of automatically tuned continuous optimizers on different benchmark sets. Applied Soft Computing 27, 490–503.

Liu, J., Reeves, C.R., 2001. Constructive and composite heuristic solutions to the P//$\sum$ Ci scheduling problem. European Journal of Operational Research 132 (2), 439–452.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58.

Lu, C., Huang, Y., Meng, L., Gao, L., Zhang, B., Zhou, J., 2022. A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers. Robotics and Computer-Integrated Manufacturing 74, 102277.

Lu, C., Gao, R., Yin, L., Zhang, B., 2023. Human-Robot collaborative scheduling in energy-efficient welding shop. IEEE Transactions on Industrial Informatics. https://doi.org/10.1109/TII.2023.3271749.

Naderi, B., Ruiz, R., 2010. The distributed permutation flowshop scheduling problem. Computers & operations research 37 (4), 754–768.

Nagano, M.S., de Almeida, F.S., Miyata, H.H., 2021. An iterated greedy algorithm for the no-wait flowshop scheduling problem to minimize makespan subject to total completion time. Engineering Optimization 53 (8), 1431–1449.

Nawaz, M., Enscore Jr, E.E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11 (1), 91–95.

Neufeld, J.S., Gupta, J.N., Buscher, U., 2016. A comprehensive review of flowshop group scheduling literature. Computers & Operations Research 70, 56–74.

Neufeld, J.S., Teucher, F.F., Buscher, U., 2020. Scheduling flowline manufacturing cells with inter-cellular moves: non-permutation schedules and material flows in the cell scheduling problem. International Journal of Production Research 58 (21), 6568–6584.

Pagnozzi, F., Stützle, T., 2019. Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. European journal of operational research 276 (2), 409–421.

Palmer, D.S., 1965. Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. Journal of the Operational Research Society 16 (1), 101–107.

Pan, Q.K., Gao, L., Wang, L., Liang, J., Li, X.Y., 2019. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. Expert Systems with Applications 124, 309–324.

Pan, Q.K., Gao, L., Wang, L., 2022. An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. IEEE Transactions on Cybernetics 52 (7), 5999–6012.

Pan, B., Zhang, Z., Lim, A., 2021. Multi-trip time-dependent vehicle routing problem with time windows. European Journal of Operational Research 291 (1), 218–231.

Rajendran, C., 1993. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. International Journal of Production Economics 29 (1), 65–73.

Rivera, G., Porras, R., Sanchez-Solis, J.P., Florencia, R., García, V., 2022. Outranking-based multi-objective PSO for scheduling unrelated parallel machines with a freight industry-oriented application. Engineering Applications of Artificial Intelligence 108, 104556.

Ruiz, R., Pan, Q.K., Naderi, B., 2019. Iterated Greedy methods for the distributed permutation flowshop scheduling problem. Omega 83, 213–222.

Shao, W., Shao, Z., Pi, D., 2021. Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem. Computers & Operations Research 136, 105482.

Yu, C., Semeraro, Q., Matta, A., 2018. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. Computers & Operations Research 100, 211–229.

Zhang, B., Pan, Q.K., Gao, L., Meng, L.L., Li, X.Y., Peng, K.K., 2020. A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow

shop scheduling problem. IEEE Transactions on Systems, Man, and Cybernetics: Systems 50 (12), 4984–4999.
Zhang, B., Pan, Q.-K., Meng, L.-L., Zhang, X.-L., Ren, Y.-P., Li, J.-Q., Jiang, X.-C., 2021. A collaborative variable neighborhood descent algorithm for the hybrid flowshop scheduling problem with consistent sublots. Applied Soft Computing 106, 107305.
Zhang, B., Pan, Q.-k., Meng, L.-L., Lu, C., Mou, J.-H., Li, J.-Q., 2022a. An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots. Knowledge-Based Systems 238, 107819.
Zhang, B., Meng, L.L., Sang, H.Y., Lu, C., 2022b. Automatic algorithm design for multi-objective hybrid flowshop scheduling problem with variable sublots. Computer Integrated Manufacturing System 28 (11), 3403.
Zhang, B., Lu, C., Meng, L.-L., Han, Y.-Y., Sang, H.-Y., Jiang, X.-C., 2023. Reconfigurable Distributed Flowshop Group Scheduling with a Nested Variable Neighborhood Descent Algorithm. Expert Systems with Applications 217, 119548.
Zhao Q., Duan Q., Yan B., Cheng S., & Shi Y. (2023). A Survey on Automated Design of Metaheuristic Algorithms. arXiv preprint arXiv:2303.06532.