An Estimation of Distribution Algorithm-Based Hyper-Heuristic for the Distributed Assembly Mixed No-Idle Permutation Flowshop Scheduling Problem

Fuqing Zhao, Bo Zhu, and Ling Wang, *Member, IEEE*

***Abstract*—The distributed assembly mixed no-idle permutation flowshop scheduling problem (DAMNIPFSP), a common occurrence in modern industries like integrated circuit production, ceramic frit production, fiberglass processing, and steel-making, is a new model that considers mixed machines with no-idle restrictions as well as conventional machines. This article introduces an estimation of distribution algorithm-based hyper-heuristic (EDA-HH) to solve the DAMNIPFSP. Ten simple heuristic rules as low-level operations are utilized to search the solution space. The estimation of distribution algorithm is integrated into the framework of hyper-heuristic as the high-level strategy to control the low-level heuristics sequence in the solution space. The destruction and construction procedures are conducted on products and jobs in order to enhance the exploitation competence of EDA-HH. The computational simulation is carried out and the experimental results show that the proposed EDA-HH is significantly superior to the competitors in the statistical sense. The results of the 810 large-scale problem instances show the effectiveness of the EDA-HH in solving the DAMNIPFSP. Moreover, the CPLEX solver is utilized to verify the correctness of the model with some small instances.**

***Index Terms*—Distributed assembly scheduling, estimation of distribution, hyper-heuristic, mixed no-idle flowshop.**

I. Introduction

SCHEDULING is an extremely important topic in manufacturing systems [1], [2]. A robust scheduling scheme may increase the efficacy and efficiency of industrial processes. Besides, an effective optimization algorithm for scheduling problems is a crucial method to enhance the efficiency of manufacturing processing. Under the background of market competition and economic globalization, the application of distributed manufacturing and distributed assembly system is an irreversible trend [3]. The collaborative production is inevitable and important for the manufacturing industry enterprise considering the technical and economic factors [4]. Therefore, it is significantly important to study the optimization method for the scheduling problem under the distributed environment.

The distributed scheduling arises under the background of globalization. Distributed shop scheduling problems can be classified into the following categories: distributed parallel machine scheduling [5], distributed flow shop scheduling [6], [7], distributed job shop scheduling [8], distributed hybrid flow shop scheduling [9], [10], and distributed flexible job shop scheduling [11]. The distributed flow shop scheduling problem (DFSP) is a hot topic that attracted much attention from researchers.

The distributed manufacturing systems overcome the drawbacks of traditional production in a single factory [12], and performed with an advantage in product quality, production cost, and risk management [13], [14]. However, these advantages come along with the much higher scheduling difficulty in the distributed manufacturing environment. In the distributed manufacturing system, solutions must be discovered for two coupling problems, i.e., assignment of the jobs to the factories and the job sequence in the factories. The distributed permutation flowshop scheduling problem (DPFSP) is already an NP-hard problem according to the conclusion of [15].

The distributed assembly flowshop scheduling problem (DAFSP) exists in many manufacturing systems. Since the pioneering work of Hatami et al. [16], many DAFSP-related works have been published. Wang and Wang [17] first proposed an estimation of distribution algorithm (EDA)-based algorithm for the distributed assembly permutation flowshop scheduling problem (DAPFSP). Hatami et al. [18] investigated the DAPFSP under the constraint of sequence-dependent setup times (DAPFSP-SDST). Two constructive heuristics, a metaheuristic based on variable neighborhood decent, and a simplified iterated greedy algorithm have been developed for DAPFSP-SDST. In [19], the backtracking search is introduced as the control strategy under the framework of the hyper-heuristic. The effectiveness of the algorithm was verified on the DAPFSP. Sang et al. [20] first considered the total flowtime in the DAPFSP. A cooperative water wave optimization (WWO) is presented by Zhao et al. [21] for addressing the

Manuscript received 26 December 2022; accepted 27 April 2023. Date of publication 17 May 2023; date of current version 18 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62063021 and Grant 62273193; in part by the High-Level Foreign Experts Project of Gansu Province under Grant 22JR10KA007; in part by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province under Grant 21YF5WA086; in part by the Lanzhou Science Bureau Project under Grant 2018-rc-98; and in part by the Project of Gansu Natural Science Foundation under Grant 21JR7RA204. This article was recommended by Associate Editor D. O. Olson. *(Corresponding authors: Fuqing Zhao; Ling Wang.)*

Fuqing Zhao and Bo Zhu are with the School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China (e-mail: zhaofq@lut.edu.cn; 1532116227@qq.com).

Ling Wang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: wangling@tsinghua.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2023.3272311.

Digital Object Identifier 10.1109/TSMC.2023.3272311

distributed assembly no-idle flowshop scheduling problem (DANIFSP). Zhang et al. [22] proposed a matrix cube-based EDA for the energy-efficient DAPFSP (EEDAPFSP). Huang et al. [23] presented a two-phase evolution algorithm for the multiobjective DAPFSP.

Pan and Ruiz [24] proposed the mixed no-idle permutation flowshop scheduling problem (MNIPFSP), which is more conform to the realistic scenario than the conventional NIPFSP. Rossi and Nagano [29] built the mathematical model for MNIPFSP and proposed an acceleration method for calculating the insertions. To address the MNIPFSP, the authors designed an effective iterated greedy algorithm and the computational results on a total of 1750 instances show the superiority of the algorithm. Recently, Rossi and Nagano [25], [26], [27] conducted the research on this topic further, including introducing the setup times on regular machines in MNIPFSP. The distributed mixed no-idle flowshop (DMNIFSP) was first studied in [28]. The authors introduced the cloud theory-based temperature reduction acceptance criterion in the iterated greedy method (CTBIG) to enhance the search capability of the algorithm. Compared with the classical iterated greedy algorithm, the CTBIG performed better on the test instances. Rossi and Nagano [29] first attempted to consider the setup times in the DMNIPFSP and they enhanced the quality of the solutions through the newly designed reconstruction strategy.

This article aims to find a promising result for the distributed assembly MNIPFSP (DAMNIPFSP). As a generalization of DANIFSP, the work related to the DAMNIPFSP is very limited. The DAMNIPFSP has strong constraint conditions. First of all, the operation on the job sequence needs to avoid randomness to satisfy the assembly condition of products. Second, the mixed no-idle constraint makes it more difficult to solve the problem than the permutation constraint or no-idle constraint. Finally, the distributed factory makes the problem become difficult. The DAMNIPFSP combines the DMNIPFSP [28] with assembly systems. There are three subproblems to consider: factory assignment, product scheduling, and job scheduling.

The difference between the DANIFSP and DMNIPFSP is that the DAMNIPFSP considers the mixed no-idle restriction, which is more conformed to a real production scenario than the DANIFSP. In practice, the start times of jobs must be delayed to satisfy the no-idle constraint in some industries. Due to the reason of technique and economics, it is unfeasible to stop a machine between jobs. However, the assumption of all the machines in a factory are no-idle machines is unreasonable. Therefore, it is worth researching the model that is close to the practice. In fact, real production scenarios existed at present, such as integrated circuit production, ceramic frit production, fiberglass processing, steel making industry, and the production of truck engine blocks in a foundry [24], [30], [31], [32] are worth to be considered deeply. In these production scenarios, only some machines are very expensive, and it is necessary to avoid idling at all costs. As aforementioned, the DAMNIPFSP is likewise NP-hard when $n > f$.

In this article, inspired by the learning mechanism of EDA and versatility of hyper-heuristic, an EDA algorithm-based hyper-heuristic (EDA-HH) for the DAMNIPFSP is proposed. The EDA algorithm is integrated into the framework of hyper-heuristic as the control strategy. The knowledge of the problem is utilized implicitly by the way of building a probability model rather than the explicit representation of knowledge. The low-level heuristics play a critical role in the algorithm because they are applied to the solution space directly. In order to enhance the exploitation of the proposed EDA-HH, the destruction and construction are applied in the hyper-heuristic scheme. The main contributions are summarized below.

1) The EDA is utilized as a high-level heuristic for solving the DAMNIPFSP.
2) A group heuristics rules are developed to explore the solution space. The simulated annealing (SA) operator is introduced to enhance the fine search capability.
3) A probability updating mechanism is developed for EDA-HH to achieve incremental learning.

The remainder of this article is structured as follows: Section II provides a systematic review of the related works. In Section III, the mathematical model of DAMNIPFSP is established and the numerical illustration is provided in the supplementary materials. Section IV presents a detail of the proposed EDA-HH. In Section V, a comprehensive evaluation is conducted on the proposed algorithm through comparative experiments and statistical analysis. The conclusion of this article and some interesting directions are discussed in Section VI.

TABLE I
REVIEW OF WORKS FOR THE DISTRIBUTED ASSEMBLY SCHEDULING

Problem	Ref.	Objectives	Methods
DAPFSP	[16]	Makespan	VND
	[37]	Makespan	GA
	[38]	Makespan	VNSMA
	[39]	Makespan	HBBO
	[40]	Makespan	MCEDA
	[41]	Makespan	BRS
	[19]	Makespan	BS-HH
	[17]	Makespan	EDAMA
	[42]	Makespan	SSS
	[33]	Total tardiness	SS-MA
	[34]	Total flowtime	gIGA
	[20]	Total flowtime	TDIWO, HDIWO, HDIWOp
DABFSP	[2]	Total tardiness	KWWO
	[35]	Makespan	KDH
DANIFSP	[21]	Makespan	CWWO
	[36]	Makespan	CHs, ILS, VNS
DANWFSP	[43]	Makespan	BKBSA
DAPFSP-SDST	[44]	Makespan	GP-HH
	[18]	Makespan	CHs, IG
EE-DAPFSP	[22]	Makespan, Total carbon emission	MCEDA
MO-DAPFSP	[23]	Total flowtime, Total tardiness	TEA
DAMNIPFSP	[1]	Total tardiness	RIG
	This work	Makespan	EDA-HH

II. RELATED WORK

The DAFSP is evolved from the DFSP (i.e., there is not only the distributed factory constraint but also the assembly constraint). In recent years, the related works about the DAFSP are mainly focused on the DAPFSP and the makespan objective. Table I provides a review of the DAFSP, which

contains mathematical models, optimization objectives, and solving methods. Note that the last line of the table is the research contents of this article.

Many methods have been developed for the DAPFSP with the objective of makespan criterion. The details of these works are presented in [16], [17], [19], [37], [38], [39], [40], [41], and [42]. Besides, Yang et al. [33] proposed a memetic algorithm, which combined with the scatter search, for the DAPFSP with the objective of total tardiness. Huang et al. [34] considered the total flowtime in the DAPFSP. Zhao et al. [2] considered the blocking constraint and total tardiness criterion in the DAFSP model. The knowledge of the problem was utilized in the enhanced WWO to balance the exploration and exploitation. Each individual in the population selects search strategy adaptively to obtain a promising solution. Yang and Li [35] presented a knowledge-driven constructive heuristic for the DABFSP with the makespan criterion. Shao et al. [36] considered the no-idle constraint in the DPFSP. The ILS and VNS are adopted in their research. Additionally, the assignment rule NR_a is first proposed for the constructive heuristic. In [43], a backtracking search algorithm (BSA)-based method is proposed to solve the DANWFSP with the makespan criterion.

The DAPFSP-SDST is a more realistic model than the classical DAPFSP. Hatami et al. [18] and Song and Lin [44] considered the DAPFSP-SDST with the makespan criterion. The DAPFSP with multiobjectives is investigated in [22] and [23]. From Table I, although the DAMNIPFSP exists in many manufacturing scenarios, the research about DAMNIPFSP is very limited.

The EDA is a novel and potential meta-heuristic. The way of generating the offspring is different from the other evolution algorithms [45], which produce a new population through the crossover and mutation operators [46]. The most important module of the EDA is the probability model, which is constructed through superior individuals. The information implicit in the probability model assists to obtain a promising population for the next iteration [40], [47]. Various algorithms based on the mechanism of EDA have been developed for shop scheduling problems [17], [48], [49], [50], [51], [52], [53]. From the previous literature, it is noteworthy that the EDA is promising for solving the shop scheduling problem, however, the computational and statistical results show that the CPU time spent by the EDA is larger than the other compared algorithms several or hundreds of times. The reason is that the population size used in the EDA usually larger than the other meta-heuristics and the EDA manipulates the jobs directly in the shop scheduling problems. In our study, the EDA is integrated into the framework of the hyper-heuristic to address the DAMNIPFSP.

The hyper-heuristic is a promising research topic, which is an approach that boosts the versatility of the search algorithm [54], [55]. However, the related research on the hyper-heuristic for solving the flowshop scheduling problem is very limited. In [54], the SA was integrated into the framework of the hyper-heuristic to solve the job shop scheduling problems. The SA is adopted to train the low-level heuristic to solve the instances. The test results exhibit the superior of the suggested algorithm. In [56], the PSO is brought into the hyper-heuristic and the PSO is utilized to optimize the parameters for dispatching rules. Song and Lin [44] considered a combination of genetic programming and the hyper-heuristic (GP-HH) to address the DAPFSP. In each generation of the evolution, GP can make the low-level heuristic search in a promising direction. The review and analysis mentioned above inspired our work.

TABLE II
NOTATION USED IN THE MODEL

Notation	Description
Indexes:	
i	Index for machines where $i \in \{1,2,\cdots,m\}$.
j	Index for jobs, $j \in \{1,2,\cdots,n\}$.
k	Position index
l,s	Index for products, $l,s \in \{1,2,\cdots,t\}$.
f	Index for factories, $f \in \{1,2,\cdots,F\}$.
Parameters:	
n	Number of jobs.
m	Number of machines.
F	Number of factories.
h	Number of products.
M'	The collection of no-idle machines.
$M-M'$	The collection of regular machines.
$p_{i,j}$	The processing time of the job J_j on machine M_i.
pp_s	Assembly time of product s at the assembly stage.
L	A sufficiently large positive number.
Variables:	
$G_{j,s}$	Binary parameter, 1 if job j is a part of products s, and 0 otherwise.
$Y_{l,s}$	The binary variable that takes value 1 if product l is an immediate predecessor of product s, and 0 otherwise.
$C_{i,k,f}$	The continuous variable represents the completion time of the job in position k of the machine M_i in factory f.
CA_s	The continuous variable represents the completion time of product s in the assembly stage.
C_k	Assembly completion time of position k in the assembly factory
$X_{j,k,f}$	The binary variable that takes value 1 if the job J_j is assigned for processing in position k of the factory f, and otherwise equals 0.
C_{max}	The makespan of a schedule.

III. PROBLEM DEFINITION AND MODEL

A. Notation Definition

The notations used in the model of DAMNIPFSP are listed in Table II.

The mathematical model of DAMNIPFSP is extended from the DAPFSP [16] and MNIPFSP [24]. The model of DAMNIPFSP is more conformed to a practical production scenario than the previous models.

B. Problem Definition

As aforementioned, the DAMNIPFSP is a generalization of the DANIPFSP, which is assumed that all of the machines in a factory have a no-idle constraint. The production and assembly stages are essential procedures of the DAMNIPFSP. There are three subproblems that need to be considered: 1) factory assignment; 2) product scheduling; and 3) job scheduling. The DAMNIPFSP under consideration in this article meets the following key assumptions.

1) There is more than one flowshop factory. All factories are mixed no-idle flow shops. There are n independent jobs processed on m machines based on the same workflows and n, m, and F are known.
2) The processing capacity of each flowshop is known in advance and the flowshops are isomorphism.
3) Initially, all jobs are available. To meet the no-idle restriction, some jobs must be delayed in their execution. During the procedure, jobs cannot be moved to other factories.
4) Each machine can process only one job at a time and each job can be processed by only one machine at a time. The processing procedure cannot be broken off since it starts.
5) The setup time, which is sequence-independent, is neglected or included in the processing time of each job.
6) There are no breakdowns and each machine is always available.
7) During the first stage, the type of the machines is a combination of the no-idle and normal machines.
8) The assembly process begins only after all the jobs of a product have been performed in the production stage.

The illustration of the DAMNIPFSP is shown in Fig. 10 of the supplementary material.

The mixed-integer linear programming model for DAMNIPFSP is formulated as follows:

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

Subject to

$$\sum_{k=1}^{n}\sum_{f=1}^{F} X_{j,k,f} = 1 \ \forall j \tag{2}$$

$$\sum_{j=1}^{n}\sum_{f=1}^{F} X_{j,k,f} \le 1 \ \forall k \tag{3}$$

$$C_{i,k,f} \ge \sum_{j=1}^{n} X_{j,k,f} \cdot P_{i,j} \ \forall f, i = 1, k = 1 \tag{4}$$

$$C_{i,k,f} \ge C_{i-1,k,f} + \sum_{j=1}^{n} X_{j,k,f} \cdot P_{i,j} \ \forall f, k, i \in \{2, 3, \ldots, m\} \tag{5}$$

$$C_{i,k,f} \ge C_{i,k-1,f} + \sum_{j=1}^{n} X_{j,k,f} \cdot P_{i,j} \ \forall f$$
$$k \in \{2, 3, \ldots, n\}, i \in M - M' \tag{6}$$

$$C_{i,k,f} = C_{i,k-1,f} + \sum_{j=1}^{n} X_{j,k,f} \cdot P_{i,j} \ \forall f$$
$$k \in \{2, 3, \ldots, n\}, i \in M' \tag{7}$$

$$\sum_{l=0}^{h} Y_{l,s} = 1 \ \forall s \tag{8}$$

$$\sum_{s=0}^{h} Y_{l,s} \le 1 \ \forall l \tag{9}$$

$$\mathrm{CA}_s \ge C_{m,j} + pp_s - \left(1 - G_{j,s}\right) \cdot L \ \forall j, s \tag{10}$$

$$C_k \ge \mathrm{CA}_l - \left(1 - Y_{l,k}\right) \cdot L \ \forall l, k \tag{11}$$

$$C_k \ge C_{k-1} + pp_l - \left(1 - Y_{l,k}\right) \cdot L \ \forall l, k > 1 \tag{12}$$

$$C_{\max} \ge C_k \ \forall k \tag{13}$$

$$C_{i,k,f} \ge 0 \ \forall i, k, f \tag{14}$$

$$\mathrm{CA}_s \ge 0 \ \forall s \tag{15}$$

$$X_{j,k,f} \in \{0, 1\} \ \forall j, k, f \tag{16}$$

$$Y_{l,s} \in \{0, 1\} \ \forall l, s, l \ne s \tag{17}$$

$$G_{j,s} \in \{0, 1\} \ \forall j, s. \tag{18}$$

Equation (1) indicates the optimization objective of the problem. Constraint (2) defines that a job can not appear at more than one factory at the same time. Constraint (3) clarifies that a job can appear at only one position in a certain factory. Constraints (4) and (5) assure that job j processing in the current machine cannot begin until job j processing in the preceding machine is completed. On regular machines, the completion time relationship between adjacent jobs satisfies the constraint (6). Constraint set (7) specifies that the completion time of a job at an idle machine is equal to its processing time plus the completion time of the preceding job in the permutation. Constraints (8) and (9) ensure that each product in the assembly procedure has just one predecessor and no more than one following product. The product assembly satisfies the constraint set (10), which stipulates the assembly stage only starts after the production stage is finished. Constraint (11) defines the completion time of jobs on the assembly line. Constraint (12) stipulates the sequential relationship of the products. Constraint (13) defines the makespan (i.e., the maximum completion time) refers to the completion time of the last product on the machine of the assembly stage. It is obvious that minimizing the makespan can reduce the total production run and save production resources [16], [57]. Constraints (14)–(18) indicate the domain of the decision variables. The CPLEX solver is adopted to verify the mathematical model and the details can be seen in the supplementary material.

IV. Estimation of Distribution Algorithm-Based Hyper-Heuristic

The first published research that applied EDA for DPFSP is presented by Wang et al. [51]. They developed a memetic algorithm that embedded the EDA, which possesses an extended sampling mechanism, to address the DAPFSP [17]. In contrast to sophisticated algorithms that contain problem-specific knowledge, the EDA's key characteristic is its evolutionary mechanism based on a probability model.

The EDA is made up of a few phases. First, the initial population is created by using a uniform distribution. Then the truncated selection approach is employed on the population to select superior individuals according to the fitness values.

Algorithm 1: Procedure of EDA-HH

Inputs: The population size ps, the annealing rate ξ, the truncation rate τ, the destruction length of products d, the destruction length of jobs dj;
Initialize the population randomly and assign a random low-level heuristic sequence for each individual; Initialization the probability matrix P;
Apply the improved LLHs successively to each individual;
Calculate C_{max} of individuals;
While the terminal condition is not met **do**
    Generate the superior population by truncation selection;
    Calculate the cumulative probability;
    Generate the next generation of low-level heuristic sequences by the roulette wheel selection for individuals;
    Update the probability matrix through the incremental learning method;
    Apply the improved LLHs successively to each individual;
    Calculate C_{max} of individuals;
    Perform the destruction and construction to the best solution obtained currently; Retain the best result and solution;
End while
Output: The best result and scheduling solution.

The probability model can be established by the statistical magnitude of the selected superior individuals. Following the completion of the modeling, a collection of individuals is drawn from the generated model.

To solve the DAMNIPFSP, the EDA-HH is proposed in this part. The EDA-HH framework is presented initially, followed by the solution representation and initialization, low-level heuristics, destruction and construction, and updating mechanism.

A. Framework of the EDA-HH

This section describes the details of the framework of the proposed algorithm. The execution procedure of the EDA-HH is illustrated in Fig. 16 (supplementary materials). The pseudocode of EDA-HH is presented in Algorithm 1.

The estimation of distribution method is used as a high-level heuristic technique in the EDA-HH to manipulate the low-level heuristics. The low-level heuristics are responsible for the fine search in the solution space, whereas EDA is used to search the solution space indirectly by regulating the low-level heuristics. The primary technique of the EDA-HH is as follows.

1) The starting population is formed at random, with each heuristic appearing only once in each individual.
2) To identify a solution, each low-level heuristic is applied sequentially to the solution space. The best solution evolved by low-level heuristics sequence is utilized to evaluate each individual, and the solution is decoded to obtain a workable schedule to calculate the objective.
3) By sorting individuals and employing truncation selection, the superior subpopulation is obtained. The information from the superior low-level heuristic sequence is used to build the probability model, and the next generation is obtained by sampling the probability model.
4) The destruction and construction phases are applied to the best solution provided by the preceding step to obtain a more promising neighborhood solution.

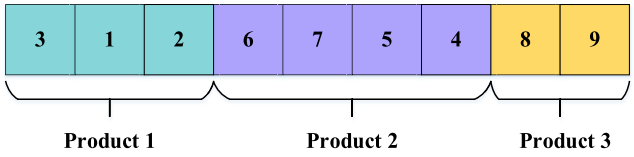


Fig. 1. Encoding scheme.

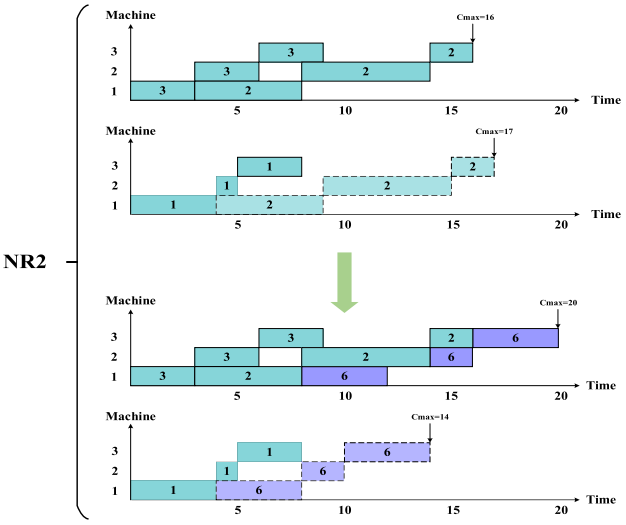


Fig. 2. Decoding scheme.

B. Solution Encoding and Decoding Schemes

In the design of an optimization algorithm, the representation of a solution is critical. The individual of EDA-HH is comprised of a solution and the low-level heuristics sequence. The low-level heuristics are employed on the individuals to evolve at each iteration. For solving the DAMNIPFSP, the two-level representation is adopted in this article. Note that the jobs for the same product are intended to be tightly linked during the two stage. An example is provided in Fig. 1 to illustrate the diagram of the solution encoding scheme.

To decide the assignment of tasks to factories, an effective decoding rule is known as NR_2 [16] is used. A job j is assigned to the factory which has the lowest $C_{\max}$ after including job j. The diagram of the solution decoding scheme is presented in Fig. 2. The job sequence can then be decoded to a workable schedule, and the completion time is computed.

C. Probability Model and Updating Mechanism

The distinction between EDA and other evolutionary algorithms is that EDA generates the next generation using a probability model [58]. The probability model in this study

is built as a probability matrix P to obtain the information inherent in the low-level heuristics sequence.

The element of the probability matrix, $p^g_{i,j}$, indicates the probability that a low-level heuristic j appears before or in position i at generation g. The value of $p^g_{i,j}$ denotes the significance of a low-level heuristic in determining the sequence of low-level heuristics. For all i and j values, $p^0_{i,j}$ is set to $1/\theta$, implying that the entire low-level heuristics space can be sampled equally. θ is the total number of low-level heuristics.

New individuals are generated in each generation of the EDA-HH by sampling the low-level heuristics space based on the probability matrix. For each position i a low-level heuristic j is chosen with the probability $p_{i,j}$. If a low-level heuristic j appears, it signifies that the low-level heuristic j has previously been used. The jth column of the probability matrix P is then set to zero, and all elements of P are normalized so that each row sums to 1. A low-level heuristic sequence is built until all of the low-level heuristics are available.

In addition, the probability updating mechanism is developed in this study to achieve incremental learning. The probability matrix P is updated according to the following formula:

$$p^{g+1}_{i,j} = (1-\alpha)p^g_{i,j} + \alpha\left(\sum_{k=1}^{ps'} I^k_{i,j}/i \times ps'\right) \forall\ i,j \tag{19}$$

$$I^k_{i,j} = \begin{cases} 1, & \text{if LLH appears before or in position } i \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where $\alpha \epsilon (0, 1)$ is the learning rate, which is set to 0.5 in this article. ps' signifies the number of individuals of the superior population. The procedure for updating the probability matrix is illustrated in Fig. 15 (supplementary materials).

Different from the other evolutionary algorithm that iteratively maintains the building block through crossover and mutation procedures, the probability model of EDA can catch the structural characteristic of the problem and evolve the next generation by implicitly utilizing the information from the superior population. In this article, the probability model is utilized in the policy space instead of the solution space. This statistic-learning mechanism is employed to learn promising policies rather than operate the job sequence.

D. Low-Level Heuristics

Low-level heuristics are commonly accepted as an essential component of a hyper-heuristic. To a considerable extent, whether a hyper-heuristic can obtain a promising result is dependent on the search capability of the low-level heuristics [59]. In this section, ten low-level heuristics are devised to build the collection of LLHs, which are described in detail below.

1) Select two distinct products randomly and exchange their positions. The relative positions between jobs belonging to the same product will keep unchanged.
2) Select two distinct products randomly and put the former after the latter. The relative positions between jobs belonging to the same product will keep unchanged.
3) Select two distinct products randomly and put the latter before the former. The relative positions between jobs belonging to the same product will keep unchanged.
4) Select two positions randomly in product permutation and place the products between the two positions in reverse order. The relative positions between jobs belonging to the same product will keep unchanged.
5) Select two adjacent positions randomly in product permutation and swap the corresponding products. The first and last positions in a permutation are regarded as the adjacent positions. The relative positions will maintain between jobs belonging to the same product.
6) Select a product from the permutation and select two distinct job positions of the product. Swap the corresponding jobs.
7) Select a product randomly from the permutation and select two distinct job positions of the product. Put the latter job before the former job.
8) Select a product randomly from the permutation and select two distinct job positions of the product. Put the former job after the latter job.
9) Select a product randomly from the permutation and select two distinct job positions of the product. Place the jobs between the two positions in reverse order.
10) Select a product randomly from the permutation and select two adjacent job positions of the product. Swap the corresponding jobs.

Before conducting job-related LLHs, a product is randomly picked to ensure that jobs belonging to the same product are not split after the LLHs. Furthermore, SA is included in each LLH to improve the performance of the fine search. The improved LLH is executed according to Algorithm 2. The product or job sequence that needs to be processed is denoted by φ. ξ is the annealing rate. The beginning temperature is presented by T_o and the terminal temperature is presented by T_f.

Algorithm 2: LLH

```
The initial temperature is set to T_0
π = φ
The LLH is applied on π to generate π'
While T_0 > T_f
    The LLH is applied on π' to generate π''
    If ΔT < 0 (ΔT = C_max(π'') − C_max(π')) then
        π' = π''
    Else
        If rand(0, 1) < exp(−ΔT/T_0) then
            π' = π''
        End if
    End if
    T_0 = ξ × T_0
End while
If C_max(π') < C_max(φ) then
    φ = π'
End if
return φ
```

E. Destruction and Construction Phases

This effective mechanism is introduced into our research. The related works published at present illustrate the significance of destruction and construction phases for every IG algorithm [31], [60], [61]. There are two types of destruction for the solution of the DAMNIPFSP, i.e., the destruction and construction of jobs and products. The destruction length for products and jobs are denoted by d and dj, respectively. The removed products and jobs must then be reinserted to create a complete product permutation and job sequence. Jobs from the same product are not divided to minimize assembly completion time delays. Jobs pertaining to the same product are designated in the same color, as shown in Figs. 12 and 13 of the supplementary material.

In this article, the elitism strategy is adopted in the destruction and reconstruction phase. The destruction and reconstruction are only performed on the current best solution. On the one hand, this strategy can improve the current optimum, and save computational resources on the other hand.

V. Computational Results

A. Experimental Setup

In this section, the computational simulation is implemented to test the performance of the EDA-HH. At present, there is no standard DAMNIPFSP test set. The test set generated by ourselves can be download at https://pan.baidu.com/s/1hD1FgcWcDzMiCbWuzNVkrA?pwd=1wg0. The large-scale benchmark instances are utilized in our experiments. There are 810 large-scale instances which is the combinations of $n \in \{100, 200, 500\}$, $m \in \{5, 10, 20\}$, $f \in \{4, 6, 8\}$, and $t \in \{30, 40, 50\}$. Where the number of jobs, machines, factories, and products are indicated by n, m, f, and t, respectively. In our study, seven combinations of machine constraints with different proportions as suggested by Pan and Ruiz [24] were used to test algorithms. The last group set, setting all the machines are no-idle machines, is excluded from our experiments. The CPU time is adopted as the stopping criterion in experiments. The max CPU time is set to $T = c \times m \times n$ milliseconds, where $c = \{20, 40, 60\}$. Five state-of-the-art algorithms are utilized to compare with the proposed EDA-HH on the benchmark test suite. All experiments are implemented by using Java (jdk12.0.2) language in IntelliJ IDEA. To evaluate the performance of the proposed EDA-HH and other comparison algorithms, experimental results are evaluated by the average relative percentage deviation (ARPD) as follows:

$$ARPD = \frac{1}{R}\sum_{i=1}^{R}\frac{\text{Cmax}_i - \text{Cmax}_*}{\text{Cmax}_*} \times 100\% \tag{21}$$

where R is the execution times of each algorithm. The result of ith run is denoted by Cmax_i and Cmax_* indicates the best result obtained currently.

B. Parameter Calibration

This section calibrates the parameter of the EDA-HH to perform at its best performance. The critical parameters of EDA-HH include the population size (ps), truncation rate τ, annealing rate ξ, d, and dj. Various values for each parameter are as follows: $ps = \{5, 10, 15, 20\}$, $\tau = \{0.2, 0.3, 0.4, 0.5\}$, $\xi = \{0.6, 0.7, 0.8, 0.9\}$, $d = \{2, 3, 4, 5\}$, and $dj = \{5, 6, 7, 8, 9\}$. Therefore, there are $4 \times 4 \times 4 \times 4 \times 5 = 1280$ combinations of parameters. There are 81 instances selected based on various combinations from the benchmark for this experiment. The proposed algorithm with a different combination of parameters is executed five times.

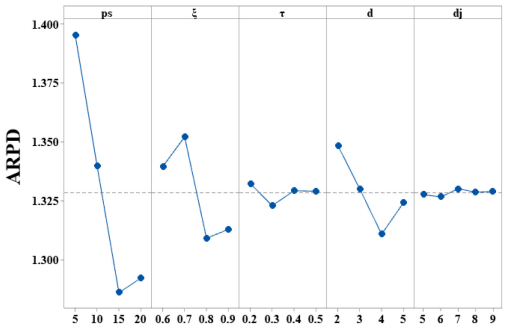


Fig. 3. Parameter analysis for the EDA-HH.

The design of the experiment (DOE) [62] is implemented to investigate the influence of the control parameters on the proposed EDA-HH. The analysis of variance (ANOVA) is employed to analyze the experimental results. The results of ANOVA are presented in Table X (supplementary materials). As can be seen in Table X, the $pvalue$ of the four parameters reflect that all the four parameters are significantly affect the experimental results. From Table X, there exist interaction effects among some parameters. The main effects plot is shown in Fig. 3 and the interaction plots for parameters are shown in Fig. 14 (supplementary materials).

From Fig. 3, it can be seen that ps has a significant impact on the test instances and the best results can be obtained when the ps is set to 15. The setting of ps needs to tradeoff the diversity of the population and the computation resources. As can be seen in Fig. 3, a greater value of ξ leads to a lower ARPD. A large ξ, on the other hand, will result in a significant computational cost. The truncation rate τ determines the proportion of the superior individuals comprises in the population. The results show that EDA-HH can obtain more promising results than the other settings when τ equals 0.3. The setting of τ needs to consider trading off the diversity of the population and search efficiency. The small value of τ can ensure the quality of the selected individuals but lost the diversity and vice versa. To avoid the loss of accuracy of the probability model, the setting of t is usually smaller than 0.5. The interaction plots show that the EDA-HH performed well when the parameters setting of ps, ξ, d, and dj are 15, 0.8, 4, and 6, respectively. Through the above analysis, the parameters setting are suggested as follows: $ps = 15$, $\xi = 0.8$, $\tau = 0.3$, $d = 4$, and $dj = 6$. T_0 and T_f are set to 2 and 1, respectively.

C. Comparative Experimental Analysis

In this part, the suggested EDA-HH is compared against the algorithms for the DAPFAP from the literature. We choose the best-competing algorithms from the DAPFSP literature

TABLE III
PARAMETER CONFIGURATION OF OTHER ALGORITHMS

Algorithms	Parameter setting
RIG	$ConType = 1; LSType = 2; dp = 3; dj = 5; \beta = 0$
BS-HH	$Itr_{max} = 150; \lambda = 5; r_{max} = 1; \xi = 0.8;\ T_0 = 2;\ T_f = 1$
TDIWO	$PS_0 = 10$; $PS_{max} = 50$; $\delta = 5$; $S_{max} = 20$; $\rho = 0.9$; $S_{min} = 1$
HDIWO	$PS_0 = 10; PS_{max} = 50; \delta = 5; S_{max} = 20; S_{min} = 1$
HDIWOp	$PS_0 = 10$; $PS_{max} = 50$; $\delta = 5$; $S_{max} = 20$; $\alpha = 0.9$; $S_{min} = 1$

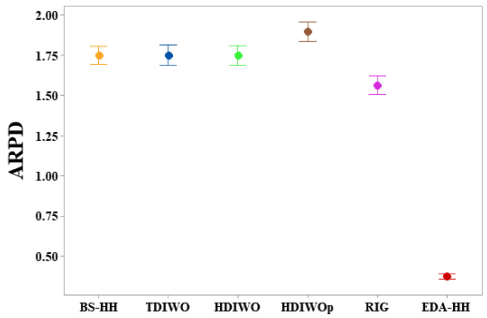


Fig. 4. Interval plot of EDA-HH and competitors.

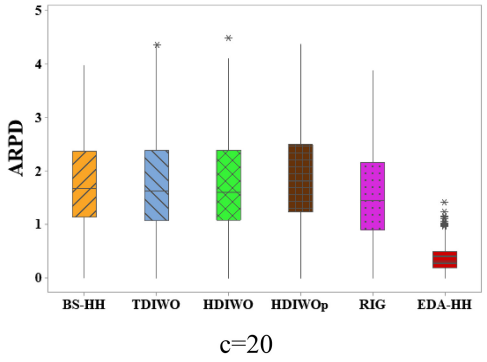


Fig. 5. Box plot of the algorithms.

because there is just one DAMNIPFSP approach proposed so far. The techniques listed below have been completely reimplemented: RIG [1], BS-HH [19], TDIWO [20], HDIWO [20], and HDIWOp [20]. We rigorously reimplement the above algorithms based on the original paper. The parameter configuration of these algorithms, which is listed in Table III, is taken from the original literature to ensure the optimal performance of the algorithm. All algorithms are executed five times under the same CUP time $T = c \times m \times n$. The ARPD values of the six algorithms on the large-scale instances are listed in Table IX (supplementary materials). The results are grouped based on n, m, f, and t. Table IX shows that the proposed EDA-HH performed best on large-scale instances.

From Fig. 4 and Fig. 19 (supplementary materials), the proposed EDA-HH has the advantage compared with other algorithms. The box plots of these algorithms are shown in Fig. 5 and Fig. 20 (supplementary materials). From the smallest interquartile range (IQR) and compact outliers of EDA-HH, the EDA-HH has small fluctuation and strong stability. Based on the comprehensive analysis, the proposed EDA-HH performed better than the other competitors in terms of efficiency and stability.

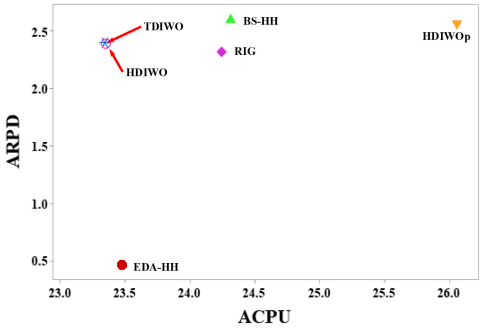


Fig. 6. ARPD versus ACPU of compared algorithms on benchmark instances.

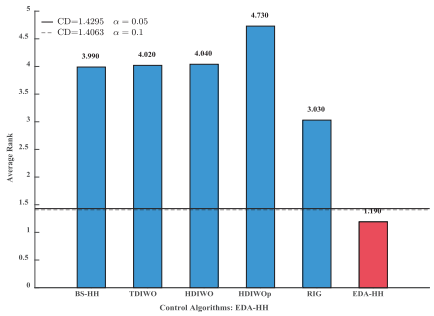


Fig. 7. Friedman test results on $c = 20$.

Furthermore, the average CPU (ACPU) running time on the instances with 100 jobs is given in Table IV. Note that each algorithm is executed for five epochs for each instance to compute the ACPU. From Table IV, the ARPD of the proposed EDA-HH is the smallest by comparison. For ACPU, the HDIWO performs best on large-scale instances. The ACPU results of the proposed EDA-HH are slightly inferior to the HDIWO. Although the proposed EDA-HH consumes more computational effort, it is acceptable due to the obtained scheduling solutions with high quality. As shown in Fig. 6, the proposed EDA-HH can maintain the balance of the ARPD and ACPU, and can perform competitively on time effort. The ACPU time is defined as follows:

$$ACPU_j = \left(\sum CPU_{i,j}\right)/I \tag{22}$$

where I is the number of instances and $\text{CPU}_{i,j}$ denotes CPU time (in seconds) that algorithm j consumes in instance i.

D. Nonparametric Test

In this section, two nonparametric tests are adopted to evaluate the performance of the proposed EDA-HH and the experimental results are statistically analyzed to understand the difference among comparison algorithms.

The Friedman results of the compared algorithms are shown in Table V. The average ranking of the proposed EDA-HH is smaller than the other competitors. The results illustrated that the proposed EDA-HH performed with an advantage in the comparison of the competitors on large-scale instances and show that the differences are significant in the statistical sense. Fig. 7 and Fig. 18 (supplementary materials) exhibit the visualized statistical test results. Note that Bonferroni–Dunn's method is employed as a post hoc procedure to calculate the

TABLE IV
ARPD VERSUS ACPU RESULTS OF COMPARED ALGORITHMS

		BS-HH		TDIWO		HDIWO		HDIWOp		RIG		EDA-HH	
		ARPD	ACPU	ARPD	ACPU	ARPD	ACPU	ARPD	ACPU	ARPD	ACPU	ARPD	ACPU
n	100	2.5976	24.3101	2.3944	23.3479	2.4025	**23.3446**	2.5748	26.0793	2.3252	24.2504	**0.4591**	23.4741
	5	1.5681	10.9124	1.3496	10.0295	1.3806	**10.0241**	1.6907	12.2318	1.2987	10.7882	**0.3661**	10.0773
m	10	2.7031	20.9439	2.5203	20.0095	2.5347	**20.0078**	2.6800	22.9330	2.3993	20.9169	**0.4842**	20.1331
	20	3.5215	41.0740	3.3133	40.0046	3.2923	**40.0019**	3.3537	43.0734	3.2775	41.0462	**0.5269**	40.2121
	4	2.3961	24.4207	2.1377	23.3423	2.1550	**23.3403**	2.3538	26.2657	2.0886	24.2012	**0.4490**	23.4430
f	6	2.5845	24.0337	2.3528	23.3531	2.3625	**23.3488**	2.5304	25.8395	2.2922	24.2653	**0.4757**	23.4769
	8	2.8674	24.4760	2.7046	23.3482	2.7074	**23.3446**	2.8123	26.1329	2.5980	24.2848	**0.4861**	23.5025
	30	2.5298	24.2407	2.2933	23.3459	2.2993	**23.3430**	2.4959	24.7192	2.2392	23.9906	**0.4628**	23.4655
s	40	2.5905	24.3201	2.3859	23.3487	2.3896	**23.3458**	2.5780	26.0359	2.3090	24.2465	**0.4682**	23.4775
	50	2.6524	24.3778	2.4796	23.3488	2.4856	**23.3450**	2.6430	27.2702	2.3833	24.4352	**0.4697**	23.4776
Average		2.6011	24.3109	2.3931	23.3478	2.4009	**23.3445**	2.5712	26.0580	2.3211	24.2425	**0.4647**	23.4739

TABLE V
FRIEDMAN-TEST RESULTS

c	Average rank						Chi-Square	p-value	CD (α=0.05)	CD (α=0.1)
	BS-HH	TDIWO	HDIWO	HDIWOp	RIG	EDA-HH				
20	3.99	4.02	4.04	4.73	3.03	**1.19**	1835.414	0.0	1.4295	1.4063
40	4.39	3.62	3.59	5.09	3.27	**1.05**	2195.677	0.0	1.2895	1.2663
60	4.06	4.35	3.81	4.62	3.13	**1.03**	2008.202	0.0	1.2263	1.2061

TABLE VI
WILCOXON RESULTS OF THE ALGORITHMS

c	EDA-HH vs.	R−	R+	Z	p-value	$\alpha = 0.1$	$\alpha = 0.05$
20	BS-HH	18	789	-24.55	3.77E-133	**Yes**	**Yes**
	TDIWO	38	768	-24.34	6.06E-131	**Yes**	**Yes**
	HDIWO	37	770	-24.35	5.49E-131	**Yes**	**Yes**
	HDIWOp	19	788	-24.55	3.91E-133	**Yes**	**Yes**
	RIG	33	775	-24.36	4.19E-131	**Yes**	**Yes**
40	BS-HH	1	806	-24.60	1.01E-133	**Yes**	**Yes**
	TDIWO	6	797	-24.54	4.95E-133	**Yes**	**Yes**
	HDIWO	6	795	-24.51	1.09E-132	**Yes**	**Yes**
	HDIWOp	4	799	-24.54	4.68E-133	**Yes**	**Yes**
	RIG	5	798	-24.54	4.84E-133	**Yes**	**Yes**
60	BS-HH	2	804	-24.59	1.50E-133	**Yes**	**Yes**
	TDIWO	3	798	-24.51	1.00E-132	**Yes**	**Yes**
	HDIWO	1	798	-24.48	2.06E-132	**Yes**	**Yes**
	HDIWOp	1	801	-24.53	6.59E-133	**Yes**	**Yes**
	RIG	1	802	-24.54	4.56E-133	**Yes**	**Yes**

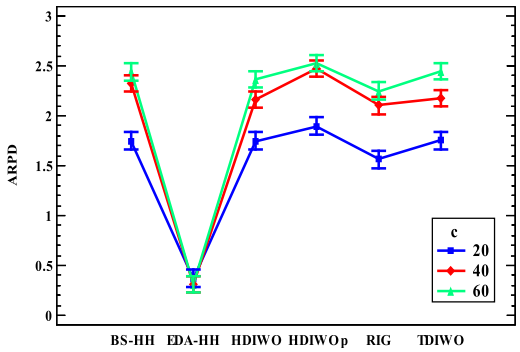


Fig. 8. Interval plot of interaction between compared algorithms and instance characteristics.

critical difference (CD) as show in (23). In (23), k is the number of comparison algorithms and N is the number of instances. The calculation of q_a is detailed in [63]

$$\mathrm{CD} = q_a \sqrt{\frac{k(k+1)}{6N}}. \tag{23}$$

As can be seen in Table VI, the results of the Wilcoxon sign rank test show the proposed EDA-HH performed more effectively than the competitors under the 90% and 95% confidence levels. In Table VI, $R+$ means that the EDA-HH performed better than the comparison algorithms on R+ instances, and R- indicates that the other algorithm performed better than EDA-HH. The "Yes" in Table VI illustrated that there exists a significant difference when EDA-HH compared with the other competitors.

In addition, the multifactor ANOVA, which considers the relationship between various factors and ARPD value, is employed to analyze the test results further. The means plots and interval plots with 95% confidence are reported in Fig. 8 and Fig. 21 (supplementary materials). From these figures, the performance of all algorithms gets better with the increasing number of jobs and gets worse with the increasing number of machines. As shown in Fig. 21(d), the change of the number of products has not affect the ARPD value and the test result of ED-HH is relatively better than the other algorithms. It is noteworthy that the ARPD is decreased for the proposed EDA-HH from the perspective of the time factor. The ARPD of the other compared algorithms is not decreased with the increase of the time factor. The reason for this fact is that the results obtained by the proposed EDA-HH get better with the increase of the time factor, which is contrary to the other compared algorithms. On the other hand, there exist differences between the results of each execution due to the randomness of the

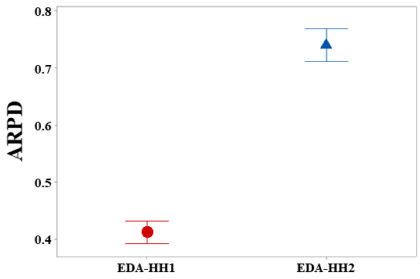


Fig. 9. Interval plot on $c = 20$.

algorithm. The ARPD value of EDA-HH and the number of factories are inverse ratios. Comprehensively, the above results and analysis demonstrated the effectiveness of the proposed EDA-HH on the DAMNIPFSP with the objective of the $C_{\max}$ criterion.

E. Effectiveness Analysis of Components

The effect of destruction and construction on the performance of the proposed EDA-HH is analyzed in this section. Initialization, destruction, construction, and acceptance criterion are the four stages of the IG algorithm [24], [64]. The IG relies heavily on destruction and construction. During the destruction phase, various components of the existing solution are taken at random or according to predetermined rules. The removed parts are reinserted into the partial solution based on certain rules to establish a complete solution. In this study, the destruction and construction are incorporated into the EDA-HH method to improve algorithm exploitation. As illustrated in Table VII of the supplementary material, the EDA-HH with destruction and construction and the EDA-HH without destruction and construction are tested on the 810 large-scale benchmark instances to verify the effectiveness of the destruction and construction. The configuration of the parameters is adopted according Section V-B and the CPU time of execution is adopted to be the stopping condition in this test. Each algorithm is executed five times independently for the 810 instances.

The ARPD values of the experimental results are listed in Table VII. As shown in Fig. 9 and Fig. 17 (supplementary materials), the EDA-HH with destruction and construction and the EDA-HH without destruction and construction are represented by EDA-HH 1 and EDA-HH 2, respectively. The confidence interval of EDA-HH 1 is not overlapped with the intervals of EDA-HH 2, which means that EDA-HH 1 is statistically better than EDA-HH 2. The results show that the EDA-HH with destruction and construction is significantly superior to another one. From Table VII, Fig. 9, and Fig. 17, the EDA-HH with destruction and construction obtained good performance from the perspective of jobs, machines, factories, and products. Thus, the proposed EDA-HH has good robustness.

F. Complexity Analysis

The analysis of the computational complexity of the proposed EDA-HH is as follows: the computational complexity of the task of applying the LLHs (lines 3 and 10) to each individual is $\mathcal{O}(ps \cdot A \cdot \log_{\xi} 1/2)$, where ps is the population size, A is the number of low-level heuristics, and ξ is the annealing rate. The computational complexity of the calculation of cumulative probability (line 7) is $\mathcal{O}(A(A-1))$, where A is the number of low-level heuristics. Updating the probability matrix (line 9) has a computational complexity of $\mathcal{O}(A^2)$. The computational complexity of destruction and construction for products and jobs are $\mathcal{O}(d^3)$ and $\mathcal{O}(dj^3)$, respectively.

VI. Conclusion and Discussion

The DAMNIPFSP with $C_{\max}$ criteria is considered in this article. We used CPLEX to verify the proposed MILP model of DAMNIPFSP with the makespan criterion. A novel hyper-heuristic is presented to address the DAMNIPFSP. The time-consuming flaw of classic EDA can be alleviated by the EDA-based hyper-heuristic approach. The inclusion of the destruction-construction enhances the exploitation capacity of the proposed EDA-HH. The comparative experiments of EDA-HH and the state-of-the-art algorithm on the large-scale benchmark exhibit the effectiveness of the proposed algorithm. To illustrate the substantial difference between the suggested EDA-HH and competitors, the nonparametric test is used. The experimental comparison and statistical analysis show that the proposed EDA-HH is an effective algorithm for addressing the DAMNIPFSP. The proposed EDA-HH still has room for improvement. There are some limitations to the proposed algorithm such as the sole probability model that restrict the performance of the algorithm and more effective low-level heuristics need to be designed. The interaction between the high-level heuristic and the low-level heuristic can be enhanced. Moreover, the performance of the proposed algorithm in the practice needs to be tested. Although the model has advanced significantly, there are still some limitations to it. For instance, it does not take into account multiobjective optimization, complex constraint conditions, and dynamic scenarios.

In the future, other outstanding algorithms and techniques will be introduced to enhance the capability of the components under the framework of the hyper-heuristic. The incorporation of some fitness landscape analysis methods to assess the properties of the problems is an intriguing direction that can boost the performance of the algorithm even further. The redesign of the updating mechanism and enhancement of the probability model is attractive for future research. It is worth noting that the DAMNIPFSP with many assembly factories has not recently been investigated. Furthermore, solving the multiobjective DAMNIPFSP [65] is a challenging job. Especially distributed green scheduling has become a hot topic in the research community. Under the background of globalization and carbon neutrality, manufacturing should not only focus on profit growth but also the harmony between humans and nature. Manufacturing enterprises need to consider reducing industrial pollution such as noise to defend the health of the industrial workers. The combination of reinforcement learning and the scheduling problem is promising.

REFERENCES

[1] Y.-Z. Li, Q.-K. Pan, R. Ruiz, and H.-Y. Sang, "A referenced iterated greedy algorithm for the distributed assembly mixed no-idle permutation flowshop scheduling problem with the total tardiness criterion," *Knowl.-Based Syst.*, vol. 239, Mar. 2022, Art. no. 108036, doi: 10.1016/j.knosys.2021.108036.

[2] F. Zhao, D. Shao, L. Wang, T. Xu, N. Zhu, and Jonrinaldi, "An effective water wave optimization algorithm with problem-specific knowledge for the distributed assembly blocking flow-shop scheduling problem," *Knowl.-Based Syst.*, vol. 243, May 2022, Art. no. 108471, doi: 10.1016/j.knosys.2022.108471.

[3] Y.-Z. Li, Q.-K. Pan, K.-Z. Gao, M. F. Tasgetiren, B. Zhang, and J.-Q. Li, "A green scheduling algorithm for the distributed flowshop problem," *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107526, doi: 10.1016/j.asoc.2021.107526.

[4] H. Guo, H. Sang, B. Zhang, L. Meng, and L. Liu, "An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times," *Knowl.-Based Syst.*, vol. 242, Apr. 2022, Art. no. 108328, doi: 10.1016/j.knosys.2022.108328.

[5] J.-Q. Li et al., "Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2425–2439, Jun. 2020, doi: 10.1109/tcyb.2019.2943606.

[6] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete Jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 12675–12686, Dec. 2022, doi: 10.1109/tcyb.2021.3086181.

[7] J. Chen, L. Wang, and Z. Peng, "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100557, doi: 10.1016/j.swevo.2019.100557.

[8] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems," *Comput. Oper. Res.*, vol. 80, pp. 101–112, Apr. 2017, doi: 10.1016/j.cor.2016.11.021.

[9] Y. Li et al., "A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 59, no. 13, pp. 1–20, 2020, doi: 10.1080/00207543.2020.1753897.

[10] J.-Q. Li, X.-L. Chen, P.-Y. Duan, and J.-H. Mou, "KMOEA: A knowledge-based multiobjective algorithm for distributed hybrid flow shop in a prefabricated system," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5318–5329, Aug. 2022, doi: 10.1109/tii.2021.3128405.

[11] Y. Du, J. Li, C. Luo, and L. Meng, "A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100861, doi: 10.1016/j.swevo.2021.100861.

[12] G. Jules and M. Saadat, "Agent cooperation mechanism for decentralized manufacturing scheduling," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 12, pp. 3351–3362, Dec. 2017, doi: 10.1109/tsmc.2016.2578879.

[13] J.-J. Wang and L. Wang, "A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1805–1819, May 2020, doi: 10.1109/tsmc.2017.2788879.

[14] Q.-K. Pan, L. Gao, L. Wang, J. Liang, and X.-Y. Li, "Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem," *Expert Syst. Appl.*, vol. 124, pp. 309–324, Jun. 2019, doi: 10.1016/j.eswa.2019.01.062.

[15] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, 2010, doi: 10.1016/j.cor.2009.06.019.

[16] S. Hatami, R. Ruiz, and C. Andrés-Romano, "The distributed assembly permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5292–5308, 2013, doi: 10.1080/00207543.2013.807955.

[17] S.-Y. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 1, pp. 139–149, Jan. 2016, doi: 10.1109/tsmc.2015.2416127.

[18] S. Hatami, R. Ruiz, and C. Andrés-Romano, "Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times," *Int. J. Prod. Econ.*, vol. 169, pp. 76–88, Nov. 2015, doi: 10.1016/j.ijpe.2015.07.027.

[19] J. Lin, Z.-J. Wang, and X. Li, "A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 36, pp. 124–135, Oct. 2017, doi: 10.1016/j.swevo.2017.04.007.

[20] H.-Y. Sang et al., "Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion," *Swarm Evol. Comput.*, vol. 44, pp. 64–73, Feb. 2019, doi: 10.1016/j.swevo.2018.12.001.

[21] F. Zhao, L. Zhang, J. Cao, and J. Tang, "A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 153, Mar. 2021, Art. no. 107082, doi: 10.1016/j.cie.2020.107082.

[22] Z.-Q. Zhang, R. Hu, B. Qian, H.-P. Jin, L. Wang, and J.-B. Yang, "A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem," *Expert Syst. Appl.*, vol. 194, May 2022, Art. no. 116484, doi: 10.1016/j.eswa.2021.116484.

[23] Y.-Y. Huang, Q.-K. Pan, L. Gao, Z.-H. Miao, and C. Peng, "A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem," *Swarm Evol. Comput.*, vol. 74, Oct. 2022, Art. no. 101128, doi: 10.1016/j.swevo.2022.101128.

[24] Q.-K. Pan and R. Ruiz, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem," *Omega*, vol. 44, pp. 41–50, Apr. 2014, doi: 10.1016/j.omega.2013.10.002.

[25] F. L. Rossi and M. S. Nagano, "Heuristics for the mixed no-idle flowshop with sequence-dependent setup times and total flowtime criterion," *Expert Syst. Appl.*, vol. 125, pp. 40–54, Jul. 2019, doi: 10.1016/j.eswa.2019.01.057.

[26] F. L. Rossi and M. S. Nagano, "Heuristics for the mixed no-idle flowshop with sequence-dependent setup times," *J. Oper. Res. Soc.*, vol. 72, no. 2, pp. 417–443, 2019, doi: 10.1080/01605682.2019.1671149.

[27] F. L. Rossi and M. S. Nagano, "Heuristics and metaheuristics for the mixed no-idle flowshop with sequence-dependent setup times and total tardiness minimisation," *Swarm Evol. Comput.*, vol. 55, Jun. 2020, Art. no. 100689, doi: 10.1016/j.swevo.2020.100689.

[28] C.-Y. Cheng, K.-C. Ying, H.-H. Chen, and H.-S. Lu, "Minimising makespan in distributed mixed no-idle flowshops," *Int. J. Prod. Res.*, vol. 57, no. 1, pp. 48–60, 2018, doi: 10.1080/00207543.2018.1457812.

[29] F. L. Rossi and M. S. Nagano, "Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107337, doi: 10.1016/j.cie.2021.107337.

[30] W. Shao, D. Pi, and Z. Shao, "Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion," *Appl. Soft Comput.*, vol. 54, pp. 164–182, May 2017, doi: 10.1016/j.asoc.2017.01.017.

[31] K.-C. Ying, S.-W. Lin, C.-Y. Cheng, and C.-D. He, "Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 110, pp. 413–423, Aug. 2017, doi: 10.1016/j.cie.2017.06.025.

[32] N. E. H. Saadani, A. Guinet, and M. Moalla, "Three stage no-idle flow-shops," *Comput. Ind. Eng.*, vol. 44, no. 3, pp. 425–434, 2003, doi: 10.1016/S0360-8352(02)00217-6.

[33] Y. Yang, P. Li, S. Wang, B. Liu, and Y. Luo, "Scatter search for distributed assembly flowshop scheduling to minimize total tardiness," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 861–868, doi: 10.1109/cec.2017.7969399.

[34] Y.-Y. Huang, Q.-K. Pan, J.-P. Huang, P. Suganthan, and L. Gao, "An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 152, Feb. 2021, Art. no. 107021, doi: 10.1016/j.cie.2020.107021.

[35] Y. Yang and X. Li, "A knowledge-driven constructive heuristic algorithm for the distributed assembly blocking flow shop scheduling problem," *Expert Syst. Appl.*, vol. 202, Sep. 2022, Art. no. 117269, doi: 10.1016/j.eswa.2022.117269.

[36] W. Shao, D. Pi, and Z. Shao, "Local search methods for a distributed assembly no-idle flow shop scheduling problem," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1945–1956, Jun. 2019, doi: 10.1109/jsyst.2018.2825337.

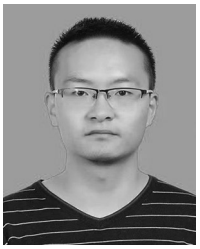
[37] X. Li, X. Zhang, M. Yin, and J. Wang, "A genetic algorithm for the distributed assembly permutation flowshop scheduling problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2015, pp. 3096–3101, doi: 10.1109/cec.2015.7257275.

[38] B. Liu, K. Wang, and R. Zhang, "Variable neighborhood based memetic algorithm for distributed assembly permutation flowshop," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, 2016, pp. 1682–1686, doi: 10.1109/cec.2016.7743990.

[39] J. Lin and S. Zhang, "An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem," *Comput. Ind. Eng.*, vol. 97, pp. 128–136, Jul. 2016, doi: 10.1016/j.cie.2016.05.005.
[40] Z. Zi-Qi, Q. Bin, H. Rong, J. Huai-Ping, and W. Ling, "A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100785, doi: 10.1016/j.swevo.2020.100785.
[41] E. M. Gonzalez-Neira, D. Ferone, S. Hatami, and A. A. Juan, "A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times," *Simulat. Model. Pract. Theory*, vol. 79, pp. 23–36, Dec. 2017, doi: 10.1016/j.simpat.2017.09.001.
[42] A. Hamzadayı, M. A. Arvas, and A. Elmi, "Distributed assembly permutation flow shop problem; single seekers society algorithm," *J. Manuf. Syst.*, vol. 61, pp. 613–631, Oct. 2021, doi: 10.1016/j.jmsy.2021.10.012.
[43] F. Zhao, J. Zhao, L. Wang, and J. Tang, "An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem," *Appl. Soft Comput.*, vol. 112, Nov. 2021, Art. no. 107750, doi: 10.1016/j.asoc.2021.107750.
[44] H.-B. Song and J. Lin, "A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100807, doi: 10.1016/j.swevo.2020.100807.
[45] W. Shi et al., "A coevolutionary estimation of distribution algorithm for group insurance portfolio," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 11, pp. 6714–6728, Nov. 2022, doi: 10.1109/tsmc.2021.3096013.
[46] Y. Liang, Z. Ren, X. Yao, Z. Feng, A. Chen, and W. Guo, "Enhancing Gaussian estimation of distribution algorithm by exploiting evolution direction with archive," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 140–152, Jan. 2020, doi: 10.1109/tcyb.2018.2869567.
[47] L. Bao, X. Sun, Y. Chen, D. Gong, and Y. Zhang, "Restricted Boltzmann machine-driven interactive estimation of distribution algorithm for personalized search," *Knowl.-Based Syst.*, vol. 200, Jul. 2020, Art. no. 106030, doi: 10.1016/j.knosys.2020.106030.
[48] B. Jarboui, M. Eddaly, and P. Siarry, "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2638–2646, 2009, doi: 10.1016/j.cor.2008.11.004.
[49] Q.-K. Pan and R. Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times," *Omega*, vol. 40, no. 2, pp. 166–180, 2012, doi: 10.1016/j.omega.2011.05.002.
[50] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 62, no. 4, pp. 917–926, 2012, doi: 10.1016/j.cie.2011.12.014.
[51] S. Wang, L. Wang, M. Liu, and Y. Xu, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 387–396, 2013, doi: 10.1016/j.ijpe.2013.05.004.
[52] B. Qian, Z. Q. Zhang, R. Hu, H. P. Jin, and J. B. Yang, "A matrix-cube-based estimation of distribution algorithm for no-wait flow-shop scheduling with sequence-dependent setup times and release times," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 3, pp. 1492–1503, Mar. 2023.
[53] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progr. Artif. Intell.*, vol. 1, no. 1, pp. 103–117, 2012, doi: 10.1007/s13748-011-0005-3.
[54] F. Garza-Santisteban et al., "A simulated annealing hyper-heuristic for job shop scheduling problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2019, pp. 57–64, doi: 10.1109/cec.2019.8790296.
[55] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *Eur. J. Oper. Res.*, vol. 285, no. 2, pp. 405–428, 2020.
[56] S. Nguyen and M. Zhang, "A PSO-based hyper-heuristic for evolving dispatching rules in job shop scheduling," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 882–889, doi: 10.1109/cec.2017.7969402.
[57] J. M. Framinan, R. Leisten, and R. Ruiz-Usano, "Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation," *Eur. J. Oper. Res.*, vol. 141, no. 3, pp. 559–569, 2002, doi: 10.1016/S0377-2217(01)00278-8.
[58] F. Zhao, B. Zhu, L. Wang, T. Xu, N. Zhu, and J. Jonrinaldi, "An offline learning co-evolutionary algorithm with problem-specific knowledge," *Swarm Evol. Comput.*, vol. 75, Dec. 2022, Art. no. 101148, doi: 10.1016/j.swevo.2022.101148.
[59] X. Hao, R. Qu, and J. Liu, "A unified framework of graph-based evolutionary multitasking hyper-heuristic," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 35–47, Feb. 2021, doi: 10.1109/tevc.2020.2991717.
[60] M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and A. H.-L. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Inf. Sci.*, vol. 181, no. 16, pp. 3459–3475, 2011.
[61] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 55, pp. 795–816, Nov. 2008.
[62] J. R. Smith and C. Larson, "Statistical approaches in surface finishing. Part 3. Design-of-experiments," *Trans. IMF*, vol. 97, no. 6, pp. 289–294, 2019, doi: 10.1080/00202967.2019.1673530.
[63] J. Zar, *Biostatistical Analysis*. Hoboken, NJ, USA: Prentice-Hall, 2010.
[64] Y.-Z. Li, Q.-K. Pan, J.-Q. Li, L. Gao, and M. F. Tasgetiren, "An adaptive iterated greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems," *Swarm Evol. Comput.*, vol. 63, Jun. 2021, Art. no. 100874, doi: 10.1016/j.swevo.2021.100874.
[65] X.-L. Zheng and L. Wang, "A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 5, pp. 790–800, May 2018.

Fuqing Zhao received the B.Sc. degree in mechanical design and manufacturing and the Ph.D. degree in material processing engineering from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively.

Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technology, where he became a Full Professor in 2012. He has been as the Postdoctoral Fellow with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China, in 2009. He has been as a Visiting Scholar with the Exeter Manufacturing Enterprise Center, Exeter University, Exeter, U.K., and the Georgia Tech Manufacturing Institute, Georgia Institute of Technology, Atlanta, GA, USA, from 2008 to 2019 and from 2014 to 2015, respectively. He has authored two academic books and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.



Bo Zhu received the B.S. degree in computer science and technology from the Lanzhou University of Technology, Lanzhou, China, in 2019, where he is currently pursuing the M.S. degree in computer application technology.

His current research interests include intelligent optimization and scheduling algorithms.

Ling Wang (Member, IEEE) received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a Full Professor in 2008. He has authored five academic books and more than 300 refereed papers. His current research interests include intelligent optimization and production scheduling.

Prof. Wang was a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, and the Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. He is currently an Editor-in-Chief of *International Journal of Automation and Control* and an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Swarm and Evolutionary Computation*.