Applied Soft Computing 135 (2023) 110022

Contents lists available at ScienceDirect

Applied Soft Computing

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

# An effective hyper heuristic-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem

Hong-Bo Song, You-Hong Yang, Jian Lin [*], Jing-Xuan Ye

*School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou, 310018, China*

ARTICLE INFO

*Article history:*
Received 26 January 2022
Received in revised form 22 November 2022
Accepted 6 January 2023
Available online 13 January 2023

*Keywords:*
Distributed assembly flow-shop scheduling
Estimation of distribution algorithm
Hyper heuristic
Memetic algorithm
Critical-products-based local search

ABSTRACT

In this paper, an effective Hyper Heuristic-based Memetic Algorithm (HHMA) is proposed to solve the Distributed Assembly Permutation Flow-shop Scheduling Problem (DAPFSP) with the objective of minimizing the maximum completion time. A novel searching-stage-based solution representation scheme is presented for both improving the search efficiency and maintaining potential solutions. In the global search stage, Estimation of Distribution Algorithm (EDA) is employed as the high level strategy of EDA-based Hyper Heuristic (EDAHH) to find promising product sequences for further exploitation. Based on the newly found knowledge of critical-products, several efficient Low-Level Heuristics (LLHs) are well designed to construct the LLH set so that the powerful exploration ability of the EDAHH can be guaranteed. A simulated-annealing-like type of acceptance criterion is also embedded into each LLH to avoid premature convergence. Then a Critical-Products-based Referenced Local Search (CP-RLS) method is proposed to improve the quality of superior sub-population by operating on the sub-job-sequences derived from the critical products. The benefit of the presented CP-RLS lies in the excellent exploitation ability with substantially reduced computational cost. Finally, performance evaluation and comparison are both carried out on a benchmark set and the results demonstrate the superiority of HHMA over the state-of-the-art algorithms for the DAPFSP.

## 1. Introduction

During the past decades, distributed assembly production systems can be commonly found in practical manufacturing processes such as auto-mobiles and semi-conductor devices [1,2]. In a distributed assembly production system, jobs are allocated to distributed factories for processing at the first stage and then are assembled into the final products at the second assembly stage. Both the benefits of distributed production and assembly are covered for such systems, including low cost, high quality and anti-risk capacity, flexibility and easy maintenance [3–5]. Thus, increasing attention has been paid to distributed assembly production systems by various manufacturing enterprises.

Production scheduling has long been a hot research topic in manufacturing systems [6,7]. The production efficiency can be significantly improved if appropriate scheduling schemes are chosen. The Permutation Flow-shop Scheduling Problem (PFSP) is one of the most general scheduling problems that models practical production scheduling process. As well known, PFSP is a typical NP-hard combinatorial optimization problem and has extensively been studied since 1960s. Fruitful research results of PFSP can be found in [8–15] and the references therein. More recently, PFSP for the distributed and assembly production systems, which is called the Distributed Assembly Permutation Flow-shop Scheduling Problem (DAPFSP), has received considerable research attention. Apparently, scheduling in each factory can be seen as a PFSP in the DAPFSP while the distributed and assembly features make the problem much more complicated. Therefore, the DAPFSP is an interesting yet challenging research direction that deserves investigation.

Actually, the bottleneck of resources in the DAPFSP is the assembly machine and making use of it as efficiently as possible is essential for optimizing objectives such as maximum completion time (makespan). Based on this principle, a new concept called critical-products is presented for the DAPFSP in this paper. The key knowledge for critical-products is that any operations within the non-critical-products and the jobs used to assemble them have no effects on improving the utilization efficiency of the assembly machine, and hence no reduction of makespan can be made. Therefore, it is reasonable to restrict the searching areas and spend more computational resources on the critical products and the related jobs. However, the critical products can only be determined when the assembly order of products is obtained. Thus, we expect to find potential product assembly orders first and then use an appropriate local search method that can fully exploit the jobs within the critical-products. Such

* Corresponding author.
*E-mail address:* linjian1001@126.com (J. Lin).

https://doi.org/10.1016/j.asoc.2023.110022
1568-4946/© 2023 Elsevier B.V. All rights reserved.

idea has not been considered in the literature [1,16–26] on the DAPFSP, including our previous published papers [17,18,26], and is very promising to obtain improved results, which motivates this research work.

From the general description of our idea, it can be seen that Memetic Algorithm (MA) is a perfect framework to design the corresponding algorithm. Up to date, many practical problems have been efficiently solved within the MA framework, such as multi-mode resources-constrained project scheduling problems and periodic capacitated arc routing problems [27–29]. Within the MA framework, global search is combined with problem-dependent local search to solve problems under consideration. Thus, incorporating the prior knowledge of critical-products into the MA is helpful for improving the scheduling efficiency of the DAPFSP. On the other hand, although evolutionary algorithms are commonly employed as the global search method of MAs, no systematic way to use prior knowledge and premature convergence are general issues for them [30].

In recent years, hyper-heuristic has become more and more popular for optimization problems [31–33]. Instead of searching the solution space directly, hyper-heuristic operates on a LLH set to generate heuristic sequences and applies them to find better solutions. The two mentioned issues can be relieved to certain a extent by designing LLHs with prior knowledge and embedding a simulated-annealing-like type of acceptance criterion into each LLH, respectively. As for the high level strategy, an Estimation of Distribution Algorithm (EDA) is employed due to its nice features with respect to the DAPFSP [16,20]. In EDA, new population is generated by sampling an updated probability model at each iteration and more promising areas are tracked by the information of superior sub-population. Thus, an EDA-based Hyper Heuristic (EDAHH) is developed as the global search strategy for the DAPFSP in this paper to both maintain the benefits of EDA and compensate for the limitations.

For the local search, a Critical-Products-based Referenced Local Search (CP-RLS) with powerful exploitation ability is proposed to perform on the superior sub-populations. It should be pointed out that the computational cost of traditional Referenced Local Search (RLS) algorithm is high and the CP-RLS significantly reduces it by taking the advantages of the critical-products-based knowledge without sacrificing the exploitation ability. According to the above description, a Hyper Heuristic-based Memetic Algorithm (HHMA) is proposed for the DAPFSP with EDAHH and CP-RLS being the global and local search algorithms, respectively.

Moreover, the characteristics of global and local search of the HHMA make the coding scheme conflicting. A novel searching-stage-based solution representation method is then presented to handle this issue. Specifically, a Multi-Permutation Representation (MPR) of a solution is used in the global search stage to improve search efficiency while a Full Permutation Representation (FPR) is used for the local search to maintain potential solutions. It is worth pointing out that either MPR or FPR was used in the existing algorithms for the DAPFSP and hence the benefits of them cannot be fully utilized. Parameter testing is carried out for the HHMA based on the Taguchi method of Design-of-Experiment (DOE). Finally, a benchmark instance set is employed to evaluate the performance of the proposed HHMA. Comparison results show the superiority of HHMA over the state-of-the-arts for the DAPFSP.

The main contribution of this paper is summarized as follows: (1) A new critical-products-based knowledge is found for the DAPFSP to help reasonably restrict the searching areas and allocate the computational resources. (2) An effective HHMA with EDAHH and CP-RLS being the global and local search strategy, respectively, is proposed to solve the DAPFSP based on the critical-products-based knowledge. (3) A novel searching-stage-based coding scheme is presented to improve search efficiency of the EDAHH and maintain potential solutions within CP-RLS. (4) Evaluation results of the HHMA show the effectiveness and superior performance over the state-of-the-art algorithms for the DAPFSP. New best solutions are updated for 39 instances on the benchmark instance set.

The rest of the paper is organized as follows. Section 2 presents a literature review. In Section 3, the DAPFSP is formulated in detail. The HHMA is illustrated in Section 4. Both the evaluation and comparison results are presented in Section 5. Finally, a conclusion and future research directions are given in Section 6.

## 2. Literature review

The study of Assembly Flow-shop Scheduling Problem (AFSP) and distributed scheduling problem are closely related to the DAPFSP. In [34], a three-machine AFSP in which two machines are used for job processing and the other for assembly was investigated and three heuristics were proposed to find approximate solutions. Parallel machines were considered at the job processing stage in [35] and a heuristic with a compact vector summation technique was presented to obtain guaranteed worst-case performance. In [36], the transportation between production and assembly stage were taken into consideration for AFSP and several heuristics were proposed with worst-case ratio bound analysis. In [37], the AFSP with multiple non-identical assembly machines was studied and a hybrid algorithm combining the variable neighborhood search with a heuristic was presented to solve it. The aforementioned literature considered the objective being makespan minimization. In [38], AFSP aiming at minimizing the maximum lateness was addressed and a self-adaptive differential evolution heuristic was proposed. More research results on AFSP can be referred to the survey paper [5].

The distributed scheduling, also called multi-factory scheduling or scheduling for multi-site manufacturing systems in the literature such as [39–41], has gained significant research attention recently. The production environment in each factory was categorized by single-machine, parallel machines, job shop, flow-shop and open shop in [39] and the research results on them were reviewed. In [40], the supply chain scheduling problem for multi-site manufacturing systems was studied by considering the transportation between the supplier and the manufacturing centers. This problem can be viewed as the integration of the production scheduling and vehicle routing. The optimization objective was to minimize the total delivery time and a Reference Group Genetic Algorithm (RGGA) was presented to solve it. A case study of a drug company was given to show the effectiveness of the RGGA. In [41], the optimization objectives were considered to be total tardiness of orders and total traveled distance by vehicles for the same problem and a Multiple League Championship Algorithm (MLCA) was proposed for the multi-objective problem. In [42], an improved genetic algorithm with a novel local search operator was presented to solve the distributed flexible job shop scheduling problem. Distributed PFSP was introduced in [43] and constructive heuristics and Variable Neighborhood Descent (VND) methods were proposed for it. In [44], the issue of energy cost was considered for distributed PFSP and an improved NSGAII was presented for the multi-objective optimization problem with criteria of total flow time and energy consumption.

For the DAPFSP with the objective to minimize the makespan, the first work was done in [1] and a mixed integer linear programming model was presented. Then, three heuristics and a VND were designed to solve the problem, accordingly. In the next few years, several meta-heuristic and hyper-heuristic algorithms were developed to improve the scheduling efficiency of the DAPFSP [16–22]. In [16,17], an Estimation of Distribution Algorithm-based Memetic Algorithm (EDAMA) and a Hybrid

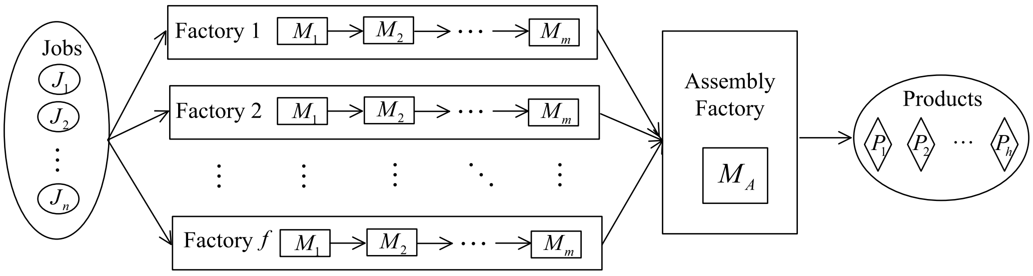

**Fig. 1.** Illustration of the DAPFSP.

**Table 1**
Notations in the formulations.

| Parameters | Description |
|---|---|
| $n$ | Number of jobs |
| $m$ | Number of machines |
| $h$ | Number of products |
| $f$ | Number of factories |
| $PJ_{i,j}$ | Processing time of job $J_i$ on machine $M_j$. |
| $PA_l$ | Assembly time of product $P_l$ |
| $N_l$ | Set consisting of the indices of jobs that are used for the assembly of product $P_l$, $l = 1, 2, \ldots, h$. |
| **Variables** | **Description** |
| $\pi = [\pi_1, \pi_2, \ldots, \pi_h]$ | Product sequence, where $\pi_l \in \{1, 2, \ldots, h\}$ is the index of the $l$th product |
| $\eta_k = [\eta_k(1), \eta_k(2), \ldots, \eta_k(D_k)]$ | Job-in-factory sequence for factory $k = 1, 2, \ldots, f$. |
| $D_k$ | Total number of jobs in factory $k$ |
| $\eta_k(i)$ | Index of the $i$th job in factory $k$, $i = 1, 2, \ldots, D_k$ |
| $CJ_i^j$ | Completion time of job $J_i$ on machine $M_j$. |
| $CP_{\pi_l}^m$ | Maximum among the completion time of jobs belonging to product $P_{\pi_l}$ on the machine $M_m$. |
| $C_{M_A}^{\pi_l}$ | Completion time of the assembly of product $P_{\pi_l}$. |
| $C_{\max}(\Gamma)$ | Makespan of a feasible schedule $\Gamma$. |
| $\Phi(\pi) = \{P_A, P_{\pi_{cp}}\}$ | Critical products for a product sequence $\pi$ |
| $P_{\pi_{cp}}$ | The first product in the last sub-product-sequence with no idle time between any two successive products |
| $P_A$ | The product set including the products assembled before $P_{\pi_{cp}}$ |
| $\theta(\pi)$ | Sub-job-sequence derived from the critical products of $\pi$ |
| $\pi_M$ | Multi-Permutation Representation of a solution |
| $\pi_F$ | Full Permutation Representation of a solution |

Biogeography-Based Optimization (HBBO) were presented for the DAPFSP, respectively. Both algorithms were designed with balanced global and local search ability. In [18], a Backtracking Search Hyper Heuristic (BS-HH) was developed with backtracking algorithm being the high level strategy that manipulates a properly-designed LLH set. In [19], a Biased-Randomized Iterated Local Search (BR-ILS) algorithm was proposed with the advantage of fewer required parameters. In [20], a Matrix-Cube-based EDA (MCEDA) was presented for the DAPFSP with a problem-dependent VND and several speedup strategies. These algorithms were all tested on benchmark sets presented in [1], and were shown to be effective. Other algorithms for the DAPFSP include genetic algorithm and single seekers society algorithm [21,22]. On the other hand, DAPFSP with other optimization objectives and constraints were considered by researchers [23–26,45–51]. To name a few, DAPFSP with total flow-time criterion was considered in [23,24] and several heuristics and meta-heuristics were presented to solve it. The DAPFSP with sequence dependent setup times was investigated in [25,26] and the presented algorithms included VND, iterative greedy algorithm and genetic programming-based hyper heuristic. The distributed two-stage assembly scheduling problem was studied in [45–47] and several meta-heuristic algorithms were proposed for it. The DAPFSP with an assembly line in each factory was introduced by [48]. Several heuristics, a VND and an iterative greedy algorithm were presented for the problem.

## 3. Problem description

Before describing the DAPFSP, the notations used in the formulations are given in Table 1. The considered DAPFSP is illustrated in Fig. 1, where $n$ jobs $\{J_1, J_2, \ldots, J_n\}$ are first assigned to $f$ distributed factories for processing and then assembled into $h$ products $\{P_1, P_2, \ldots, P_h\}$ by the assembly machine $M_A$ in the assembly factory. Each product is assembled by some defined jobs and each job is only used for the assembly of a defined product. We denote $h$ sets $N_l$ to represent which jobs are used for the assembly of product $P_l$, $l \in H = \{1, 2, \ldots, h\}$. Then, it can be seen that $\cup_{l\in H} N_l = N$ and $N_i \cap_{i,l\in H, i\neq l} N_l = \varnothing$, where $N = \{1, 2, \ldots, n\}$ and $\varnothing$ is empty set. The $f$ factories are assumed to be identical and capable of processing all jobs. If one job is assigned to a factory, then it cannot be transferred to the other factories during the processing. Scheduling in each factory can be regarded as a PFSP with $m$ machines $\{M_1, M_2, \ldots, M_m\}$. Job $J_i$ requires $m$ operations $\{O_{i1}, O_{i2}, \ldots, O_{im}\}$ to be processed in any of the $f$ factories. Operation $O_{ij}$ has to be executed on machine $M_j$ with processing time $PJ_{i,j}$.

In the assembly phase, the assembly machine $M_A$ assembles product $P_l$ with assembly time $PA_l$. The assembly of product $P_l$ can only be started when all the jobs belonging to $N_l$ are finished and $M_A$ is free. $PJ_{i,j}$ and $PA_l$ are assumed to be non-negative integrals that are deterministic and known in advance. Standard assumptions for flow-shop scheduling are also made as follows: all the jobs are available at time 0. Each machine can only process one job at a time and each job can only be processed by one machine at a time. No machine break-down and preemption are allowed. Setup times are sequence independent and included in the processing times. The optimization objective is to minimize the makespan in this paper.

For the DAPFSP, a feasible schedule $\Gamma$ is composed of $f$ job-in-factory sequences and a product sequence $\pi$, which represent the processing order of jobs in each factory and the assembly order of products, respectively. The job-in-factory sequences are denoted by $\eta_k = [\eta_k(1), \eta_k(2), \ldots, \eta_k(D_k)]$, $k = 1, 2, \ldots, f$, where $\eta_k(i)$ and $D_k$ are the indices of the $i$th job and the total number of jobs in factory $k$, respectively. The product sequence is denoted by $\pi = [\pi_1, \pi_2, \ldots, \pi_h]$, where $\pi_l$ is the index of the $l$th product to be assembled.

The makespan of a feasible schedule of $\Gamma$ is denoted by $C_{\max}(\Gamma)$ and can be calculated by the following steps:

$$CJ^{j}_{\eta_k(i)} = \max\left\{CJ^{j-1}_{\eta_k(i)}, CJ^{j}_{\eta_k(i-1)}\right\} + PJ_{\eta_k(i),j} \tag{1}$$

where $CJ^{j}_{\eta_k(i)}$ is the completion time of job $\eta_k(i)$ on machine $j$, $i = 1, 2, \ldots, D_k$, $k = 1, 2, \ldots, f$, $j = 1, 2, \ldots, m$, and $CJ^{0}_{\eta_k(1)} = CJ^{1}_{\eta_k(0)} = 0$.

$$CP^{m}_{\pi_l} = \max_{\eta_k(i)\in N_{\pi_l}, i=1,2,\ldots,D_k, k=1,2,\ldots,f} \left\{CJ^{m}_{\eta_k(i)}\right\} \tag{2}$$

where $CP^{m}_{\pi_l}$ is the maximum among the completion time of jobs belonging to product $P_{\pi_l}$ on the last machine.

$$C^{\pi_l}_{M_A} = \max\left\{CP^{m}_{\pi_l}, C^{\pi_{l-1}}_{M_A}\right\} + PA_{\pi_l} \tag{3}$$

where $C^{\pi_l}_{M_A}$ is the completion time of product $P_{\pi_l}$ on the assembly machine $M_A$, and $C^{\pi_0}_{M_A} = 0$.

$$C_{\max}(\Gamma) = C^{\pi_h}_{M_A} \tag{4}$$

As stated in [1], the DAPFSP is an NP-hard problem and obtaining a global optimal solution in polynomial time is unrealistic. Thus, a commonly used way is to find solutions as satisfactory as possible in a reasonable computing time. In the following section, a novel HHMA will be presented to achieve this goal for the DAPFSP with makespan criterion.

## 4. HHMA for the DAPFSP

In this section, design of HHMA for the DAPFSP will be presented. The critical-products-based knowledge is given at first since it is the foundation of HHMA. Operations within non-critical-products and the related jobs are eliminated to restrict the searching areas with the help of critical-products. Then, based on a newly presented coding scheme, global and local search algorithms of HHMA are both designed with the principle of spending computational resources on more promising areas to improve search efficiency. Specifically, an EDAHH with EDA generating LLH sequences from a well-designed LLH set is presented as the global search method. Potential product sequences with the corresponding critical products are obtained in this stage. As for the local search, a CP-RLS is proposed to fully exploit the sub-job-sequences within the critical products. After the procedure of HHMA, computational complexity analysis is carried out.

In Section 4.1, the critical-products-based knowledge is presented with an illustrative example, followed by the searching-stage-based solution representation in Section 4.2. Then, the designs of EDAHH and the LLH set are proposed in Sections 4.3 and 4.4, respectively. The CP-RLS is described in Section 4.5. Finally, the main procedure of HHMA and computational complexity analysis are given in Sections 4.6 and 4.7, respectively.

### *4.1. Critical-products-based knowledge*

The proposed HHMA is first finding potential product sequences in the global search stage and then using local search to fully exploit the search areas of the jobs belonging to the critical products of the product sequences. This is motivated by the critical-products-based knowledge that any operations within the non-critical-products and the jobs belonging to them have no effects on the reduction of makespan. The following definition, proposition and proof are given to formulate the finding.

**Definition 1** (*Critical-Products*)**.** For a product sequence $\pi$ of the DAPFSP, the products in set $\Phi(\pi) = \{P_A, P_{\pi_{cp}}\}$ are called critical-products, where $P_{\pi_{cp}}$ is the first product in the last sub-product-sequence with no idle time between any two successive products, and $P_A$ is the product set including the products assembled before $P_{\pi_{cp}}$. The remaining products in $\pi$ are called non-critical-products.

**Proposition 1** (*Critical-Products-Based Knowledge*)**.** *For a product sequence $\pi$ of the DAPFSP and critical products $\Phi(\pi) = \{P_A, P_{\pi_{cp}}\}$, any operations within the non-critical-products and the jobs belonging to them have no effects on the reduction of makespan.*

**Proof.** It can be obtained by (3)–(4) and Definition 1 that

$$C_{\max}(\pi) = C^{\pi_{cp}}_{M_A}(\pi) + \sum_{i=cp+1}^{h} PA_{\pi_i} \tag{5}$$

Since there are no idle times between non-critical-products, it is clear that any operations within them only change the assembly order of the non-critical products, not the sum of the assembly times of them $\sum_{i=cp+1}^{h} PA_{\pi_i}$. On the other hand, $C^{\pi_{cp}}_{M_A}(\pi)$ is determined by the critical products. Thus, it can be inferred by (5) that the makespan $C_{\max}(\pi)$ does not reduce for any operations within non-critical-products.

As for operations among the jobs within the non-critical products, we can see by (2)–(3) that for a given $C^{\pi_{cp}}_{M_A}(\pi)$, inequality $C^{\pi_{cp+i}}_{M_A}(\pi) \geq C^{\pi_{cp+i-1}}_{M_A}(\pi) + PA_{\pi_i}$ holds for $i = 1, 2, \ldots, h - cp$. Therefore, it can be obtained by (5) that $C_{\max}(\pi)$ does not reduce for any operations on the jobs within non-critical-products. The proof is completed.

The concept of critical-products extends the critical-path in [16] with deeper investigation for the characteristic of the DAPFSP. In the following, an example is given to illustrate the critical-products and critical-products-based knowledge more clearly. There are 15 jobs, 3 machines, 2 factories and 6 products in the example, that is, $n = 15$, $m = 3$, $f = 2$ and $h = 6$. The sets of jobs for assembling the products are $N_1 = \{2, 6, 14\}$, $N_2 = \{5, 15\}$, $N_3 = \{7, 9, 11\}$, $N_4 = \{1, 13\}$, $N_5 = \{3, 4, 10\}$ and $N_6 = \{8, 12\}$, respectively. The processing time of jobs and assembly time of products are given in Table 2. The Gantt chart of a feasible solution is shown in Fig. 2.

It can be clearly seen by Fig. 2 that the last sub-product-sequence with no idle time between any two successive products is $\left[P_2 \quad P_5 \quad P_6\right]$. Thus, we can obtain that $cp = 4$, $\pi_{cp} = 2$ and the critical products are $\{P_1, P_3, P_4, P_2\}$ in the example according to Definition 1. Then by Proposition 1 it can be seen that any operations within the non-critical products $P_5$ and $P_6$, such as swap them, and operations within the jobs belonging to them, will definitely not reduce the makespan value.

### *4.2. Encoding and decoding schemes*

When applying algorithms of intelligent computing to solve the DAPFSP, two solution representations were commonly used in

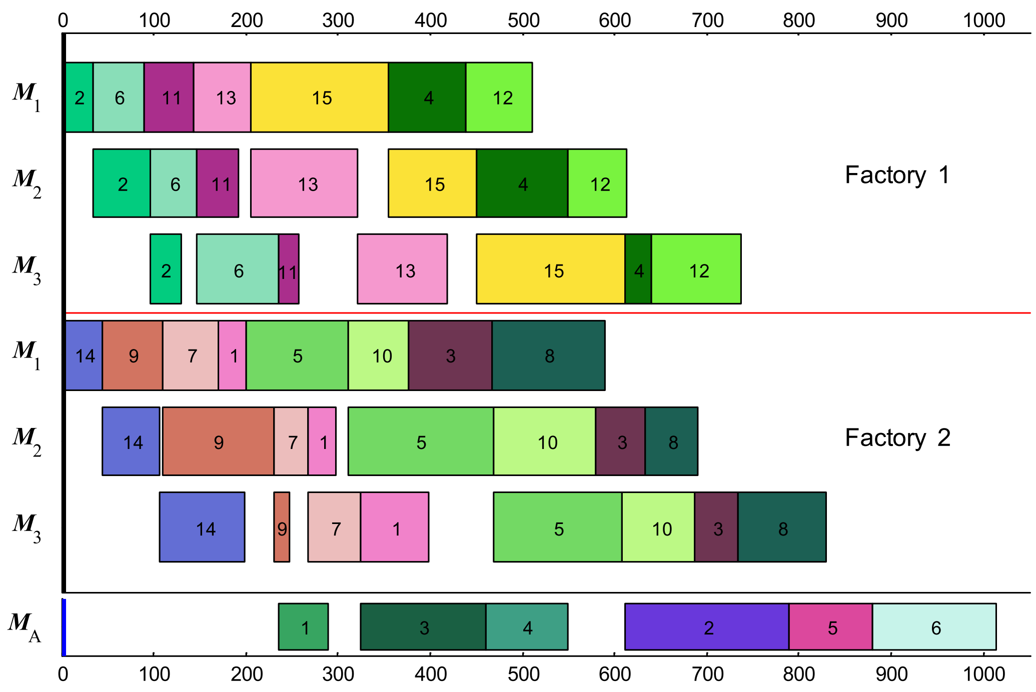

**Fig. 2.** Illustration of critical-products-based knowledge in the DAPFSP.

**Table 2**
Processing and assembly time for the example.

| Product | Job | Processing time | | | Assembly time |
|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_A$ |
| $P_1$ | $J_2$ | 35 | 61 | 44 | 53 |
| | $J_6$ | 55 | 51 | 89 | |
| | $J_{14}$ | 44 | 62 | 93 | |
| $P_2$ | $J_5$ | 111 | 157 | 140 | 178 |
| | $J_{15}$ | 149 | 96 | 162 | |
| $P_3$ | $J_7$ | 61 | 38 | 56 | 136 |
| | $J_9$ | 66 | 120 | 17 | |
| | $J_{11}$ | 54 | 45 | 22 | |
| $P_4$ | $J_1$ | 29 | 30 | 75 | 89 |
| | $J_{13}$ | 61 | 116 | 98 | |
| $P_5$ | $J_3$ | 91 | 55 | 47 | 90 |
| | $J_4$ | 85 | 100 | 28 | |
| | $J_{10}$ | 65 | 111 | 79 | |
| $P_6$ | $J_8$ | 123 | 56 | 96 | 134 |
| | $J_{12}$ | 71 | 63 | 97 | |

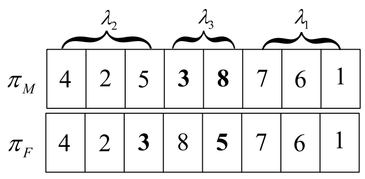

**Fig. 3.** Illustration of MPR and FPR.

the existing literature [1,16–26]. The first one is a permutation of all jobs and called Full Permutation Representation (FPR). Clearly there are $n!$ possible solutions in this case. The second makes an extra constraint that the jobs belonging to the same product are never separated and is called Multi-Permutation Representation (MPR). The number of total solutions is $h! \times \prod_{l=1}^{h} N_l!$ for MPR and less than FPR.

Generally speaking, the search efficiency of MPR is higher than FPR since the assembly machine can be more effectively utilized when the jobs belonging to the same product are processed as close as possible [25]. However, the solution space is reduced by the constraint and some potential solutions may be excluded for MPR. Due to the design principle of HHMA, it can be seen that MPR and FPR are more suitable for the global and the local search, respectively. Thus, a novel searching-stage-based solution representation scheme is proposed for HHMA. MPR is employed for the exploration efficiency in the global search stage while FPR is used for the local search so that some potential solutions are maintained for exploitation.

Specifically, a solution is represented by $\pi_M = [\lambda_{\pi_1}\ \lambda_{\pi_2}\ \ldots\ \lambda_{\pi_h}]$ for MPR, where $\lambda_{\pi_i} = [\lambda_{\pi_i,1}\ \ \lambda_{\pi_i,2}\ \ \ldots\ \ \lambda_{\pi_i,|N_{\pi_i}|}]$ is the sub-job-sequence for product $P_{\pi_i}$, $\lambda_{\pi_i,j}$ is the index of the $j$th job of $\lambda_{\pi_i}$ and $\left|N_{\pi_i}\right|$ is the total number of jobs used for the assembly of product $P_{\pi_i}$. In this case, any operations on jobs belonging to product $P_{\pi_i}$ can only be executed on sub-job-sequence $\lambda_{\pi_i}$. While FPR is simply a permutation of all jobs $\pi_F = [\xi_1\ \ \xi_2\ \ \ldots\ \ \xi_n]$, where $\xi_i$ is the index of the $i$th job, $\forall i \in N$, and all the operations on jobs can be executed on the complete job sequence $\pi_F$.

A simple example of the coding scheme with three products and eight jobs is illustrated in Fig. 3, where $N_1 = \{1, 6, 7\}$, $N_2 = \{2, 4, 5\}$ and $N_3 = \{3, 8\}$. The solution representations for MPR and FPR are $\pi_M = \big[[4\ \ 2\ \ 5]\ \ [8\ \ 3]\ \ [7\ \ 6\ \ 1]\big]$ and $\pi_F = \big[4\ \ 2\ \ 5\ \ 8\ \ 3\ \ 7\ \ 6\ \ 1\big]$, respectively. If job $J_3$ is selected for swap operation, then $J_3$ can only be swapped with $J_8$ for MPR and with any other jobs for FPR.

A solution can be decoded into a feasible schedule with a decoding scheme. The NR$_2$ rule proposed in [43] was shown to be effective and will be used in this paper. The principle of the NR$_2$ rule is assigning a job to the factory with minimal completion time after the assignment. If assigning a job to more than one factory results in the same completion time, then the job will be assigned to the factory with the smallest factory index.

### 4.3. Estimation of distribution algorithm-based hyper heuristic

Probability model is crucial for the design of EDA-based algorithms. Since hyper heuristic operates on the LLH set rather than the solution space directly, a LLH sequence should be obtained by sampling the probability model of EDAHH for each individual in the population. Therefore, the model is designed to be as $\times r$ probability matrix $\Psi(l)$ given as follows:

$$\Psi(l)=\begin{bmatrix} q_{11}(l) & q_{12}(l) & \cdots & q_{1r}(l) \\ q_{21}(l) & q_{22}(l) & \cdots & q_{2r}(l) \\ \vdots & \vdots & \ddots & \vdots \\ q_{s1}(l) & q_{s2}(l) & \cdots & q_{sr}(l) \end{bmatrix} \tag{6}$$

where $s$ is the length of heuristic sequence and $r$ is the total number of the heuristics in the LLH set, $q_{ij}(l)$ is the probability that the $i$th heuristic in heuristic sequence is heuristic $L_j$ at $l$th generation. The value of $q_{ij}(l)$ indicates the importance of heuristic $L_j$ on the $i$th position of heuristic sequence. The initial values of the probability matrix $\Psi(0)$ are chosen to be $1/r$ for $q_{ij}(0), \forall i, j$.

For kth individual, a heuristic sequence $S(k) = [S(k,1) \quad S(k,2) \quad \ldots \quad S(k,s)]$, is obtained by sampling the probability matrix $\Psi(l)$. Applying the heuristics in the sequence successively on the individual leads to a new solution. Repeating this procedure to all the individuals results in a new population. In EDAHH, the probability matrix $\Psi(l)$ should be adjusted to generate more effective heuristic sequences for the next generation. Thus, a mechanism based on a superior sub-population and historical information is used to update $\Psi(l)$. The superior sub-population includes *SP_Size* elite solutions and they are obtained by two-tournament selection method. Then the elements in $\Psi(l)$ are updated according to following equation:

$$q_{ij}(l+1)=(1-\alpha)q_{ij}(l)+\frac{\alpha}{SP\_Size}\sum_{t=1}^{SP\_Size} I_{ij}^{t} \tag{7}$$

where $\alpha \in (0,1)$ is the learning rate, $SP\_Size = \eta\% \times P\_Size$, $\eta$ is the percentage superior sub-population, *P_Size* is the population size, and $I_{ij}$ is an indicator function that

$$I_{ij}^{t}=\begin{cases}1, & \text{if ith position of } L(t) \text{ is heuristic } L_j \\ 0, & \text{else}\end{cases} \tag{8}$$

where $L(t)$ is the heuristic sequence of $t$th individual in the superior sub-population. $I_{ij}^{t} = 1$ means that operating heuristic $L_j$ at $i$th position has positive effects on finding solutions with better quality. Then it can be obtained by (7) that more solutions with $I_{ij}^{t} = 1$ in the superior sub-population leads to the larger $q_{ij}(l+1)$, indicating larger probability of heuristic $L_j$ existing in the *ith* position of the heuristic sequences for the next generation.

To further improve the quality of solutions in this stage, a simulated-annealing-type acceptance criterion is employed from [25] to avoid premature convergence. For a solution $\pi_M$ and a new one $\pi'_M$, if $C_{\max}(\pi'_M) < C_{\max}(\pi_M)$, then the new solution is accepted. Otherwise, it is still possible to accept the new solution if $rand < e^{-pp}$, where *rand* is a random variable uniformly distributed between (0, 1), and

$$pp=\frac{C_{\max}(\pi'_M)-C_{\max}(\pi_M)}{C_{\max}(\pi_M)}\times 100\times\frac{1}{Temp} \tag{9}$$

and *Temp* is a parameter that requires tuning in practice. The pseudo code of EDAHH is given in Fig. 4.

### 4.4. Low-level heuristics

It is recognized that the quality of LLH set is important for search efficiency and hence should be well designed [31,32]. Since MPR is used in the global search stage and jobs belonging to the same product are never separated, the LLHs are related to products and jobs, respectively. Due to design principle of HHMA, no LLHs should be performed within the non-critical products and the jobs belonging to them. The LLHs for products aim to provide potential product sequences with diverse critical products while the LLHs for jobs are operated on jobs belonging to the critical products. Then, seven easy-to-implemented LLHs are designed to search high quality solutions for further exploitation and they are given as follows:

$L_1$: Critical-Product-Swap: Randomly select a critical product $P_{\pi_a}$ and a product $P_{\pi_b}$, $a \neq b$, and swap them.

$L_2$: Critical-Product-Forward-Insert: Randomly select a critical product $P_{\pi_a}$ and product $P_{\pi_b}$, $a \neq b$, and insert $P_{\pi_b}$ before $P_{\pi_a}$.

$L_3$: Critical-Product-Backward-Insert: Randomly select a critical product $P_{\pi_a}$ and a product $P_{\pi_b}$, $a \neq b$, and insert $P_{\pi_a}$ before $P_{\pi_b}$.

$L_4$: Critical-Product-Inverse: Randomly select a critical product $P_{\pi_a}$ and a product $P_{\pi_b}$, $a < b$, and inverse the sub-sequence $\left[P_{\pi_a} \quad P_{\pi_{a+1}} \quad \ldots \quad P_{\pi_b}\right]$.

$L_5$: Critical-Product-Adjacent-Swap: Randomly select a critical product $P_{\pi_a}$ and swap it with the adjacent product $P_{\pi_{a+1}}$. If $a = h$, then swap $P_{\pi_a}$ with the first product $P_{\pi_1}$.

$L_6$: Jobs-in-Critical-Product-Swap: Randomly select a critical product $P_{\pi_c}$, then randomly select two jobs with index $\lambda_{\pi_c,a}$ and $\lambda_{\pi_c,b}$, in $P_{\pi_c}$, $a \neq b$ and swap them.

$L_7$: Jobs-in-Critical-Product-Inverse: Randomly select a critical product $P_{\pi_c}$, then randomly select two jobs with index $\lambda_{\pi_c,a}$ and $\lambda_{\pi_c,b}$, $a < b$ from $P_{\pi_c}$, and inverse the sub-sequence $\left[\lambda_{\pi_c,a} \quad \lambda_{\pi_c,a+1} \quad \ldots \quad \lambda_{\pi_c,b}\right]$.

### 4.5. Critical-products-based referenced local search

In this subsection, a CP-RLS method is presented to cooperate with EDAHH to form the HHMA. In the DAPFSP, the most promising area that deserves fully exploitation is the sub-job-sequence derived from the critical products. The Referenced Local Search (RLS) algorithm which was successfully applied to the PFSP is considered as the foundation of local search in HHMA. Generally speaking, RLS has powerful exploitation ability while costs much CPU time due to its computational complexity [12]. Thus, a neighborhood structure $LS_{CP}$ is presented here to reduce the computational time without sacrificing the efficiency of CP-RLS.

Note that the solution representation has been transformed from MPR to FPR at the local search stage and the critical products are also determined for the solutions in the superior sub-population. For a solution $\pi_F$ changed from $\pi_M$, the sub-job-sequence $\theta(\pi_F)$ within the critical products can be obtained as follows ($\theta(\pi_F)$ is given in Box I).

Then, $LS_{CP}$ exploits $\theta(\pi_F)$ to improve the makespan. The total number of jobs in $\theta(\pi_F)$ is denoted by $d$, and it can be seen that $d = \sum_{i=1}^{cp} \left|N_{\pi_i}\right|$. Here, we present the following proposition to help design $LS_{CP}$ and reduce the computational cost of local search.

**Proposition 2.** *For any operation on $\theta(\pi_F)$, if no improvement is made for $C_{M_A}^{\pi_{cp}}(\pi_F)$, then neither does the makespan $C_{\max}(\pi_F)$.*

**Proof.** Make an operation on $\theta(\pi_F)$ results in a new solution $\pi'_F$ and the corresponding $\theta(\pi'_F)$. If there is no improvement for $C_{M_A}^{\pi_{cp}}(\pi_F)$, that is $C_{M_A}^{\pi_{cp}}(\pi_F) \leq C_{M_A}^{\pi_{cp}}(\pi'_F)$, then it can be obtained by (5) that $C_{\max}(\pi_F) \leq C_{\max}(\pi'_F)$ holds. The proof is completed.

On the other hand, even better $C_{M_A}^{\pi_{cp}}(\pi_F)$ is found for $\theta(\pi_F)$, it cannot be guaranteed the corresponding $C_{\max}(\pi_F)$ is improved. The reason is that the critical products may be changed by the operation. Thus, we choose to calculate $C_{\max}(\pi_F)$ only if better $C_{M_A}^{\pi_{cp}}(\pi_F)$ is obtained in CP-RLS. It can be clearly seen that computational time can be substantially saved in this way.

Based on the above observation, the neighborhood structure $LS_{CP}$ is presented as follow. Remove the first job in $\theta(\pi_F)$ and

```
Procedure EDAHH ($l$-th generation of population $\pi_M(1), \pi_M(2), \ldots, \pi_M(P\_Size)$, and $\Psi(l)$)
    For $i$=1 to $s$
        Compute the accumulated probability $accprob_{ij}(l)$ by $\Psi(l)$, $j = 1, 2, \ldots, r$.
        For $k$=1 to $P\_Size$
            Generate a random scalar $rand_k \in (0,1)$.
            For $j$=1 to $r$
                If $rand_k < accprob_{ij}(l)$, then $S(k,i) = L_j$, Break.
                End If
            End For
            Perform heuristic $S(k,i)$ on $\pi_M(k)$ and obtain $\pi'_M(k)$.
            If $C_{\max}(\pi'_M(k)) < C_{\max}(\pi_M(k))$, then $\pi_M(k) = \pi'_M(k)$.
            Else generate a scalar $rand \in (0,1)$
                If $rand < e^{-pp}$, then $\pi_M(k) = \pi'_M(k)$. $pp$ is calculate by (9).
                End If
            End If
        End For
    End For
    For $t$=1 to $SP\_Size$
        Randomly select two individuals $\pi_M(\sigma_{t,1})$ and $\pi_M(\sigma_{t,2})$ from population, $\sigma_{t,1} \neq \sigma_{t,2}$.
        If $C_{\max}(\pi_M(\sigma_{t,1})) \leq C_{\max}(\pi_M(\sigma_{t,2}))$, then $L(t) = S(\sigma_{t,1})$
        Else $L(t) = S(\sigma_{t,2})$
        End If
    End For
    $l$=$l$+1
    For $i$= 1 to $s$
        For $j$=1 to $r$
            Calculate $q_{ij}(l)$ according to (7) and update $\Psi(l)$.
        End For
    End For
End Procedure
```

**Fig. 4.** Pseudo code of EDAHH.

$$\theta(\pi_F) = \left[ \lambda_{\pi_1,1} \quad \lambda_{\pi_1,2} \quad \ldots \quad \lambda_{\pi_1,|N_{\pi_1}|} \quad \lambda_{\pi_2,1} \quad \lambda_{\pi_2,2} \quad \ldots \quad \lambda_{\pi_2,|N_{\pi_2}|} \quad \ldots \quad \lambda_{\pi_{cp},1} \quad \lambda_{\pi_{cp},2} \quad \ldots \quad \lambda_{\pi_{cp},|N_{\pi_{cp}}|} \right]$$

**Box I.**

insert it at any possible position in the sequence. If smaller values of $C_{M_A}^{\pi_{cp}}$ are obtained, then compute the makespan for the solutions and accept the one with the smallest makespan as the new solution. Go back to operate on the first job of the solution if a new solution is found, otherwise execute the same operations for the next job. Repeat this procedure until no improvement of makespan is found for executing the operations on each job in the sub-job-sequence $\theta(\pi_F)$. At last, update the best solution $\pi_{best}$ if $C_{\max}(\pi_F) < C_{\max}(\pi_{best})$.

Additionally, CP-RLS only operates on the superior sub-population $SP$ to further reduce complexity. To ensure that the local optimal solution is found, an additional step is presented to add the subsequent product for the critical products into local search after CP-RLS is completed. If smaller makespan is obtained, then add the subsequent one for the product into local search. Repeat this procedure until no more improvement is made for the makespan. However, CP-RLS is enough for most of the time. The pseudo code of CP-RLS is given in Fig. 5.

## 4.6. Procedure of HHMA for the DAPFSP

Based on the detailed description on the components of HHMA designed in the above subsections, the flowchart of the proposed algorithm is illustrated in Fig. 6. Main procedure is given as follows:

**Step1:** Set parameters and randomly generate the initial population based on MPR solution representation. Set $l = 0$ and initialize the probability matrix $\Psi(l)$.

**Step2:** For each individual, sample the probability matrix $\Psi(l)$ to generate the corresponding heuristic sequence. Then perform the LLHs in the heuristic sequence successively on the individual. Repeat this for the whole individuals leads to a new population.

**Step3:** Obtain superior sub-population by tournament selection from the new population. Change the solution representation from MPR to FPR of the superior sub-population and record the critical products and sub-job-sequences of them.

```
Procedure CP-RLS ($SP$, $\pi_{best}$)
  For $k$=1 to $SP\_Size$
    Let $\pi_F = SP(k)$, and obtain the critical products and the corresponding sub-job-sequence
    $\theta(\pi_F)$ of $\pi_F$. The total number of elements in $\theta(\pi_F)$ is $d$.
    Compute $C_{M_A}^{\pi_{cp}}(\pi_F)$ and $C_{\max}(\pi_F)$.
    Let $\pi_F^B = \pi_F$, $\theta(\pi_F^B) = \theta(\pi_F)$, $C_{M_A}^{\pi_{cp}}(\pi_F^B) = C_{M_A}^{\pi_{cp}}(\pi_F)$, $i$=1, and $j$=1.
    While $i \le d$
      For $j$=1 to $d$ and $j \ne i$
        Removing the $i$-$th$ job from $\theta(\pi_F)$ and inserting it at the $j$-$th$ position of $\theta(\pi_F)$
        results in $\pi_F'$ and $\theta(\pi_F')$. Compute $C_{M_A}^{\pi_{cp}}(\pi_F')$.
        If $C_{M_A}^{\pi_{cp}}(\pi_F') < C_{M_A}^{\pi_{cp}}(\pi_F^B)$, then $\pi_F^B = \pi_F'$, $\theta(\pi_F^B) = \theta(\pi_F')$, and $C_{M_A}^{\pi_{cp}}(\pi_F^B)$=$C_{M_A}^{\pi_{cp}}(\pi_F')$.
        End If
      End For
      If $C_{M_A}^{\pi_{cp}}(\pi_F^B) < C_{M_A}^{\pi_{cp}}(\pi_F)$, then compute $C_{\max}(\pi_F^B)$.
        If $C_{\max}(\pi_F^B) < C_{\max}(\pi_F)$, then $C_{\max}(\pi_F)$=$C_{\max}(\pi_F^B)$, $\pi_F = \pi_F^B$, $\theta(\pi_F) = \theta(\pi_F^B)$ and $i$=1.
        Else $i$=$i$+1
        End if
      Else
        $i$=$i$+1
      End if
    End while
    If $C_{\max}(\pi_F) < C_{\max}(\pi_{best})$, then $\pi_{best} = \pi_F$, and $C_{\max}(\pi_{best}) = C_{\max}(\pi_F)$.
    End If
  End For
End procedure
```

**Fig. 5.** Pseudo code of CP-RLS.

**Step4:** Perform the proposed CP-RLS on the superior sub-population and update the best solution if a better solution is found.
**Step5:** Set $l = l + 1$ and update $\Psi(l)$ by (6) based on the information of superior sub-population. If the stopping condition is met, then output the best solution, otherwise go to Step2.

### 4.7. Computational complexity analysis

The computational complexity of the proposed HHMA is analyzed in this subsection. At each generation, the complexity is $O(P\_Size \times r \times s)$ to obtain the LLH sequences in the sampling process. Then, product sequences can be obtained with the computational complexity of $O(P\_Size \times h \times s)$ by applying the LLHs. The complexity for the superior sub-population selection and the updating process are $O(SP\_Size)$ and $O(r \times s)$, respectively. While the computational complexity is $O(SP\_Size \times (d^2 + n) \times f)$ for the CP-RLS. Thus, it can be seen that the computational complexity is acceptable for the DAPFSP.

## 5. Computational evaluation

In this section, numerical simulations and comparison are carried out to evaluate the performance of HHMA. The large-scaled benchmark instance set presented in [1] is used for the evaluation and comparison. Four factors are considered for the generation of the set, including the total number of jobs $n$, machines $m$, factories $f$ and products $h$. For each factor, three levels are considered with $n = \{100, 200, 500\}$, $m = \{5, 10, 20\}$, $f = \{4, 6, 8\}$ and $h = \{30, 40, 50\}$, respectively. The processing time of jobs and assembly time of products are randomly generated by $U[1, 99]$ and $U[1 \times |N_i|, 99 \times |N_i|]$, $i \in H$, respectively, where $U$ denotes a uniform distributed operator. The total number of combinations is $3^4 = 81$ and 10 replications are generated for each combination. Thus, there are 810 instances for the set in total. The data and best-known solutions for them are available at http://soa.iti.es/.

To the best of our knowledge, there are five state-of-the-art algorithms for the DAPFSP with makespan minimization being the optimization objective in the existing literature, namely, EDAMA [16], HBBO [17], BS-HH [18], BR-ILS [19] and MCEDA [20]. They are all stochastic optimization algorithms, and best and average results are always used for evaluation with several independent runs. Among them, EDAMA was evaluated by the best solution within all runs in [16] while HBBO, BS-HH and MCEDA were tested by the average value of all runs in [17,18,20], respectively. BR-ILS considered both the average and best results in [19]. In this paper, we follow the way of BR-ILS in [19] to make more complete evaluation. Based on the Relative Percentage Deviation (RPD) criterion introduced in [52], Average RPD (ARPD) and Best RPD (BRPD) are defined to measure the average and best results as follows:

$$\text{ARPD} = \sum_{i=1}^{Y} \left( \frac{W_i - W_{\text{best}}}{W_{\text{best}}} \right) \times \frac{100}{Y} \tag{10}$$

$$\text{BRPD} = \min_{i=1,2,\ldots,Y} \frac{W_i - W_{\text{best}}}{W_{\text{best}}} \times \frac{100}{Y} \tag{11}$$

where $W_i$ and $W_{\text{best}}$ are the solution at $i$th run and the best-known solution, respectively, and $Y$ is the number of running times. Additionally, the HHMA is implemented with an Intel Core i7/2.6 GHz processor and coded in Visual C++ 6.0.

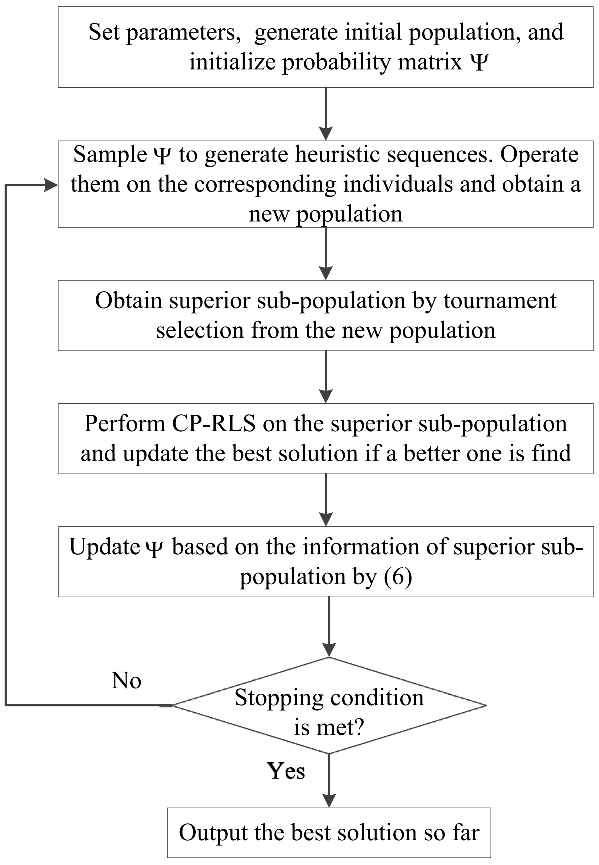
Set parameters, generate initial population, and initialize probability matrix Ψ
Sample Ψ to generate heuristic sequences. Operate them on the corresponding individuals and obtain a new population
Obtain superior sub-population by tournament selection from the new population
Perform CP-RLS on the superior sub-population and update the best solution if a better one is find
Update Ψ based on the information of superior sub-population by (6)
Stopping condition is met?
No
Yes
Output the best solution so far

**Fig. 6.** Flowchart of the proposed HHMA.

### 5.1. Parameter setting

In the proposed HHMA, four key parameters are tested, including the percentage of superior sub-population $\eta$, learning rate $\alpha$, length of heuristic sequence $s$ and acceptance-criterion-related parameter $Temp$. For fair comparison with other algorithms, the population size and maximum iteration number are set to be 50 and 150, respectively. The Taguchi method of Design-of-Experiment (DOE) in [53] is employed to evaluate the influence of these parameters on the performance of HHMA. Instance I_200_5_4_40_3 is used for the testing, where $n = 200$, $m = 5$, $f = 4$ and $h = 40$, respectively.

Four factor levels are employed for the parameters and they are listed in Table 3. Accordingly, the orthogonal array $L_{16}(4^4)$ is chosen for the significance test. All the combinations are run for 20 times independently and the corresponding ARPDs can be obtained. The array, ARPDs and average CPU times are shown in Table 4. Then, ARPDs for each factor level of the parameters, $F$ values and the significance rank are calculated and given in Table 5. Additionally, trends are also depicted in Fig. 7 for each factor level of the parameters. On the other hand, CPU time is another factor that should be taken into consideration for the level chosen of the parameters. Performance and computational cost need to be balanced to certain extent. Thus, average CPU times and the corresponding trends for each factor level of the parameters are shown in Table 6 and Fig. 8, respectively.

It can be obtained by Table 5 that the percentage of superior sub-population $\eta$ is most significant parameter. The ARPD decreases as $\eta$ increases and the reason is quite obvious since more individuals are operated by CP-RLS with larger $\eta$. However, by comparing the ARPD and CPU time trends of $\eta$ in Figs. 7 and 8 we can see that $\eta = 30$ is more suitable than $\eta = 40$. The trends indicate that more computational cost is required for smaller ARPD decreasing after $\eta = 30$. The parameter that ranks second is $Temp$ and $Temp = 3$ is best choice according to the trends. The impact of the length of heuristic sequence $s$ and the significance of learning rate $\alpha$ are relatively minor and the selection principle of the two parameters are similar to $\eta$ and $Temp$, respectively. Therefore, the suggested combination of parameters for the instance set are $\eta = 30$, $\alpha = 0.3$, $s = 10$ and $Temp = 3$, respectively.

**Table 3**
Parameters values for each factor levels.

| Parameters | Factor levels | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $\eta$ | 10 | 20 | 30 | 40 |
| $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 |
| $s$ | 5 | 10 | 15 | 20 |
| $Temp$ | 1 | 2 | 3 | 4 |

**Table 4**
Orthogonal array and ARPDs.

| Number | Factor level | | | | ARPD | Average CPU times (s) |
|---|---|---|---|---|---|---|
| | $\eta$ | $\alpha$ | $s$ | $Temp$ | | |
| 1 | 1 | 1 | 1 | 1 | 0.0019 | 2.459 |
| 2 | 1 | 2 | 2 | 2 | −0.0201 | 3.7851 |
| 3 | 1 | 3 | 3 | 3 | −0.0345 | 5.2417 |
| 4 | 1 | 4 | 4 | 4 | −0.0403 | 6.6891 |
| 5 | 2 | 1 | 2 | 3 | −0.0412 | 4.4391 |
| 6 | 2 | 2 | 1 | 4 | −0.0240 | 2.9497 |
| 7 | 2 | 3 | 4 | 1 | −0.0297 | 7.4613 |
| 8 | 2 | 4 | 3 | 2 | −0.0355 | 5.9521 |
| 9 | 3 | 1 | 3 | 4 | −0.0441 | 6.5429 |
| 10 | 3 | 2 | 4 | 3 | −0.0460 | 7.9384 |
| 11 | 3 | 3 | 1 | 2 | −0.0412 | 3.7359 |
| 12 | 3 | 4 | 2 | 1 | −0.0317 | 5.1869 |
| 13 | 4 | 1 | 4 | 2 | −0.0460 | 8.8913 |
| 14 | 4 | 2 | 3 | 1 | −0.0412 | 7.685 |
| 15 | 4 | 3 | 2 | 4 | −0.0480 | 5.7263 |
| 16 | 4 | 4 | 1 | 3 | −0.0403 | 4.4842 |

**Table 5**
ARPD values and significance rank.

| Factor level | $\eta$ | $\alpha$ | $s$ | $Temp$ |
|---|---|---|---|---|
| 1 | −0.0233 | −0.0324 | −0.0259 | −0.0252 |
| 2 | −0.0326 | −0.0329 | −0.0352 | −0.0357 |
| 3 | −0.0408 | **−0.0384** | −0.0388 | **−0.0405** |
| 4 | **−0.0439** | −0.0369 | **−0.0405** | −0.0391 |
| $F$ value | **1.8440** | 0.1912 | 0.9247 | 1.0400 |
| Rank | 1 | 4 | 3 | 2 |

**Table 6**
Average CPU times of instance I_200_5_4_40_3.

| Factor level | $\eta$ | $\alpha$ | $s$ | $Temp$ |
|---|---|---|---|---|
| 1 | 4.5437 | 5.5831 | 3.4072 | 5.6981 |
| 2 | 5.2006 | 5.5896 | 4.7844 | 5.5911 |
| 3 | 5.8510 | 5.5413 | 6.3554 | 5.5259 |
| 4 | 6.6967 | 5.5781 | 7.7450 | 5.4770 |

### 5.2. Ablation study for key components of HHMA

The effectiveness of key components of HHMA are tested by ablation study in this subsection. Four variants of HHMA are chosen by removing the EDAHH, CP-RLS, searching-stage-based coding scheme, and simulated-annealing-type acceptance criterion, respectively. They are called by HHMA_v1, HHMA_v2, HHMA_v3 and HHMA_v4, respectively, and all run for 20 times independently with suggested parameter values in Section 4.1.

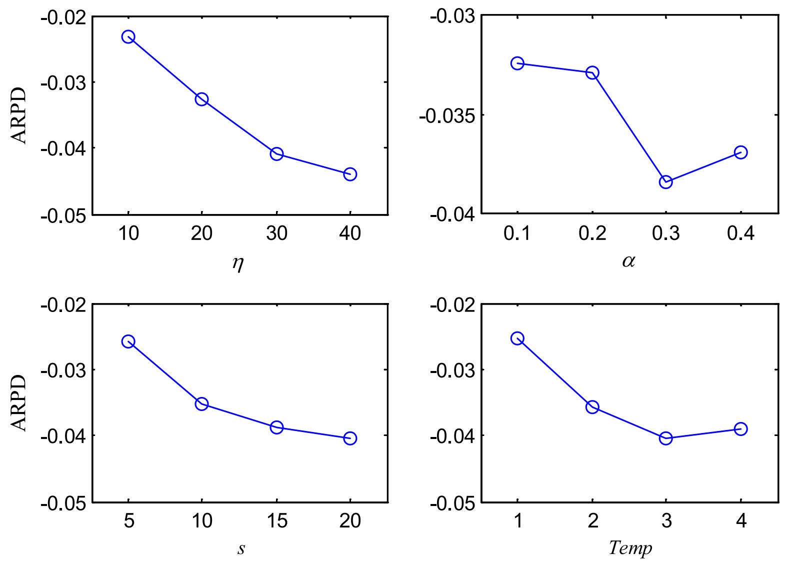

**Fig. 7.** Trends of factor levels of instance I_200_5_4_40_3.

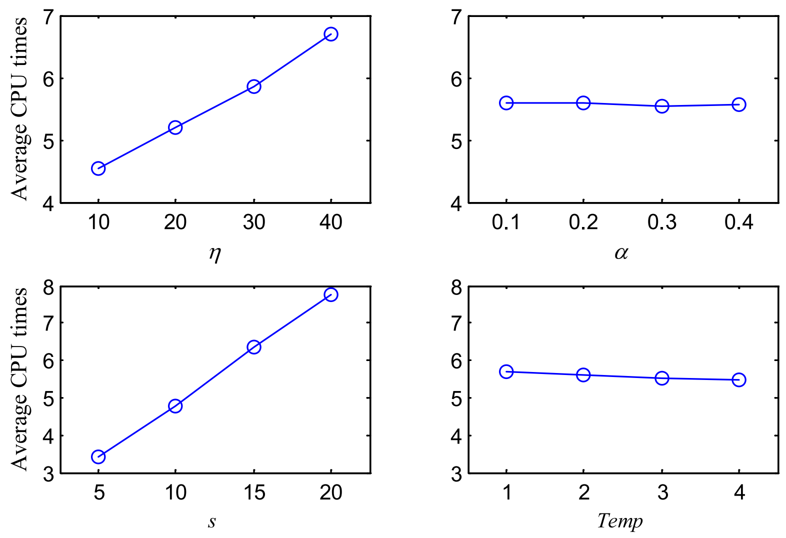

**Fig. 8.** Trends of average CPU times for instance I_200_5_4_40_3.

The testing results are given in Table 7 by the three levels of the number of factories, products and jobs.

It can be seen by Table 7 that the performance both degrades severely without EDAHH or CP-RLS, respectively, since they are global and local search algorithm of HHMA and work in a collaborative way. As for the coding scheme, if MPR is used during the whole process instead of the searching-stage-based one, then ARPD increases from −0.0013 to −0.0010. Additionally, it is also clear that the simulated-annealing-type acceptance criterion effectively avoids premature convergence for instances with small $n$, $h$ and $f$.

### 5.3. Comparison of HHMA to the state-of-the-art algorithms

In this subsection, the performance of the proposed HHMA is compared with the state-of-the-art algorithms for DAPFSP, which are EDAMA, HBBO, BS-HH, BR-ILS and MCEDA, respectively. The four key parameters are chosen as the suggested values and other parameters are set to be the same in the parameter testing. The instances are run for 20 times independently and the corresponding ARPD and BRPD can be obtained. Results of the compared algorithms come from the corresponding literature directly.

**Table 7**
Results of ablation study for components of HHMA.

| | | ARPD | | | | |
|---|---|---|---|---|---|---|
| | | HHMA_v1 | HHMA_v2 | HHMA_v3 | HHMA_v4 | HHMA |
| Factories | 4 | 0.005 | −0.002 | −0.015 | −0.018 | **−0.020** |
| | 6 | 0.003 | 0.000 | −0.010 | −0.012 | **−0.014** |
| | 8 | 0.004 | 0.002 | −0.003 | −0.005 | **−0.005** |
| Products | 30 | 0.004 | −0.001 | −0.019 | −0.021 | **−0.024** |
| | 40 | 0.004 | 0.000 | −0.006 | −0.010 | **−0.010** |
| | 50 | 0.004 | 0.001 | −0.003 | −0.004 | **−0.004** |
| Jobs | 100 | 0.007 | 0.001 | −0.003 | −0.004 | **−0.007** |
| | 200 | 0.004 | 0.001 | −0.003 | −0.005 | **−0.006** |
| | 500 | 0.001 | −0.002 | −0.023 | −0.025 | **−0.025** |
| Average | | 0.004 | 0.000 | −0.010 | −0.012 | **−0.013** |

Comparison results are summarized in Table 8 by three levels of the number of factories, products and jobs. The best result for each category is marked in bold. Then it can be seen that HHMA performs better than the compared algorithms for most of the categories. As for the average values, HHMA refreshes the

**Table 8**
Results of HHMA and the compared algorithms.

| | | BRPD | | | ARPD | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EDAMA | BR-ILS | HHMA | HBBO | BS-HH | BR-ILS | MCEDA | HHMA |
| Factories | 4 | −0.013 | −0.023 | **−0.024** | −0.013 | −0.014 | −0.015 | −0.014 | **−0.020** |
| | 6 | −0.004 | −0.007 | **−0.016** | −0.005 | −0.004 | −0.003 | −0.005 | **−0.014** |
| | 8 | −0.004 | −0.005 | **−0.006** | **−0.005** | **−0.005** | −0.003 | **−0.005** | **−0.005** |
| Products | 30 | −0.011 | −0.017 | **−0.027** | −0.012 | −0.013 | −0.010 | −0.014 | **−0.024** |
| | 40 | −0.008 | −0.012 | **−0.013** | −0.006 | −0.007 | −0.008 | −0.008 | **−0.010** |
| | 50 | −0.003 | **−0.006** | **−0.006** | **−0.004** | −0.003 | −0.003 | **−0.004** | **−0.004** |
| Jobs | 100 | −0.008 | **−0.011** | −0.008 | −0.003 | −0.003 | **−0.010** | −0.008 | −0.007 |
| | 200 | −0.006 | **−0.007** | **−0.007** | −0.003 | −0.004 | **−0.006** | **−0.006** | **−0.006** |
| | 500 | −0.007 | −0.017 | **−0.030** | −0.016 | −0.016 | −0.005 | −0.018 | **−0.025** |
| Average | | −0.007 | −0.012 | **−0.015** | −0.007 | −0.008 | −0.007 | −0.009 | **−0.013** |

best-known results by decreasing −0.003 and −0.004 for BRPD and ARPD, respectively. It is interesting to point out that HHMA shows much better performance when the number of jobs $n$ become larger. Violin plots for the BRPD and ARPD are also given in Figs. 9 and 10 to further illustrate the superiority of HHMA. The convergence curves for HHMA, HBBO and BS-HH are also given in Fig. 11, from which we can see HHMA converges to a better makespan faster than HBBO and BS-HH.

For average CPU times, the stopping conditions were chosen differently in the compared state-of-the-arts. Maximum number of generated solutions was used for EDAMA in [16] and maximum iteration number was adopted in [17,18] for HBBO and BS-HH, respectively. While maximum elapsed time was used in [19,20] for BR-ILS and MCEDA, respectively. Since the CPU times of the whole execution were given for EDAMA, BR-ILS and MCEDA, we just listed them from the literature and our result in Table 9. It can be seen by Tables 7–9 that HHMA costs less CPU time but still achieves better makespan than EDAMA, BR-ILS and MCEDA. On the other hand, HBBO and BS-HH are re-run and the comparison result of the average CPU times for one execution is given in Table 10, from which we can see that HHMA also outperforms HBBO and BS-HH in both CPU times and makespan.

Based on the best-known solutions given in [1], the Number of New Best Solutions (NNBS) were obtained for each compared algorithm in the corresponding literature. The data for EDAMA, HBBO, BS-HH and MCEDA were reported to be 94, 91, 92 and 123, respectively. No such information is explicitly given for BR-ILS. However, it was shown in [19] that BR-ILS obtained 43 new best solutions compared to those of HBBO listed in [17]. Thus, the number of new best solutions is inferred to be 113 for BR-ILS accordingly. These data were listed in Table 11 together with the one of HHMA. It can be seen that HHMA obtains 131 new best solutions also performs the best from this aspect and followed by MCEDA and BR-ILS. Additionally, new best solutions are updated for 39 instances by comparing with all the new best solutions reported in the existing literature. These new best updated solutions and the corresponding RPD are given in Table 12. The solutions of HHMA for all instances can be found at https://pan.baidu.com/s/1xedJYlslo_JR5gpr8G15Og?pwd=HHMA.

### 5.4. Discussion on the results

In this subsection, further discusses are given on HHMA with respect to the state-of-the-arts for the DAPFSP. Generally speaking, the critical-products-based knowledge plays an fundamental role for the superiority of HHMA, followed by the cooperation of the designed EDAHH and CP-RLS that makes use of the knowledge as efficiently as possible. The searching areas are reasonably restricted by the knowledge and more computational resources are spent on promising areas within critical-products by EDAHH

**Table 9**
Average CPU times of HHMA and compared algorithms (whole execution of algorithm).

| | | EDAMA | BR-ILS | MCEDA | HHMA |
|---|---|---|---|---|---|
| Factories | 4 | 22.06 | 60.17 | 26.26 | 13.80 |
| | 6 | 22.84 | 60.20 | 23.35 | 18.74 |
| | 8 | 24.58 | 60.25 | 28.36 | 25.09 |
| Products | 30 | 23.22 | 60.19 | 55.42 | 22.70 |
| | 40 | 23.31 | 60.21 | 53.38 | 18.67 |
| | 50 | 22.96 | 60.22 | 53.54 | 16.27 |
| Jobs | 100 | 3.57 | 60.04 | 5.88 | 7.68 |
| | 200 | 11.27 | 60.10 | 15.89 | 12.69 |
| | 500 | 54.65 | 60.48 | 75.89 | 37.26 |
| Average | | 23.16 | 60.21 | 37.73 | 19.21 |

**Table 10**
Average CPU times of HHMA and compared algorithms (each execution of algorithm).

| | | HBBO | BS-HH | HHMA |
|---|---|---|---|---|
| Factories | 4 | 2.122 | 0.522 | 0.092 |
| | 6 | 2.217 | 0.548 | 0.125 |
| | 8 | 2.538 | 0.568 | 0.167 |
| Products | 30 | 2.288 | 0.547 | 0.151 |
| | 40 | 2.287 | 0.549 | 0.124 |
| | 50 | 2.302 | 0.543 | 0.108 |
| Jobs | 100 | 0.054 | 0.044 | 0.051 |
| | 200 | 0.295 | 0.173 | 0.085 |
| | 500 | 6.529 | 1.421 | 0.248 |
| Average | | 2.293 | 0.598 | 0.128 |

**Table 11**
NNBS of HHMA and compared algorithms.

| Algorithms | EDAMA | HBBO | BSHH | BR-ILS | MEEDA | HHMA |
|---|---|---|---|---|---|---|
| NNBS | 94 | 91 | 92 | 113 | 123 | 131 |

and CP-RLS. The compared algorithms employed no such prior-knowledge to direct the algorithm design, and computational resources were inevitably wasted on searching areas that impossibly contain better solutions. Thus, the search efficiency of HHMA is higher than the compared state-of-the-arts. Another important factor is the coding scheme. HBBO, BS-HH and BR-ILS used MPR during the whole algorithm, and some good solutions may be excluded in this way and cannot be found. EDAMA and MCEDA adopted FPR and computational complexity of the global search is hence higher. The search-stage-based coding scheme both improves the search efficiency in the global search stage and maintains potential good solutions in the local search.

Instance I_100_20_4_30_1 is given here as an example to further illustrate the HHMA in detail. In the global search stage, *SP_Size* potential product sequences are obtained and we discuss

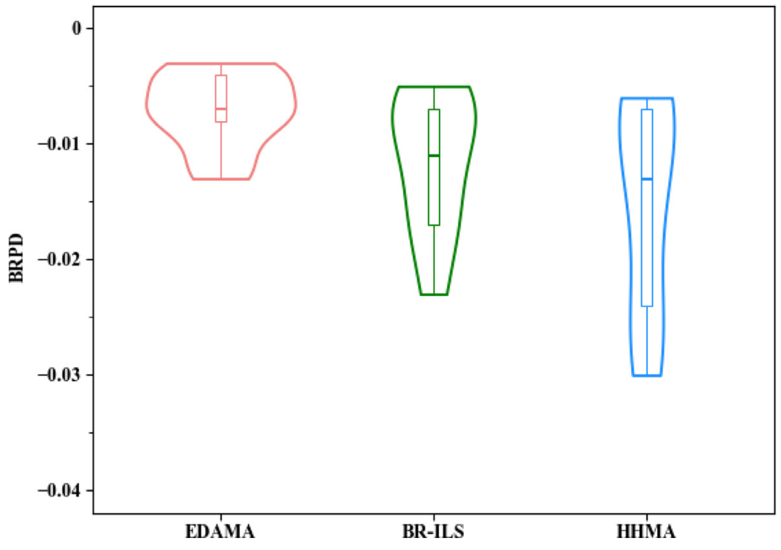

**Fig. 9.** Violin plot for BRPD of the compared algorithms.

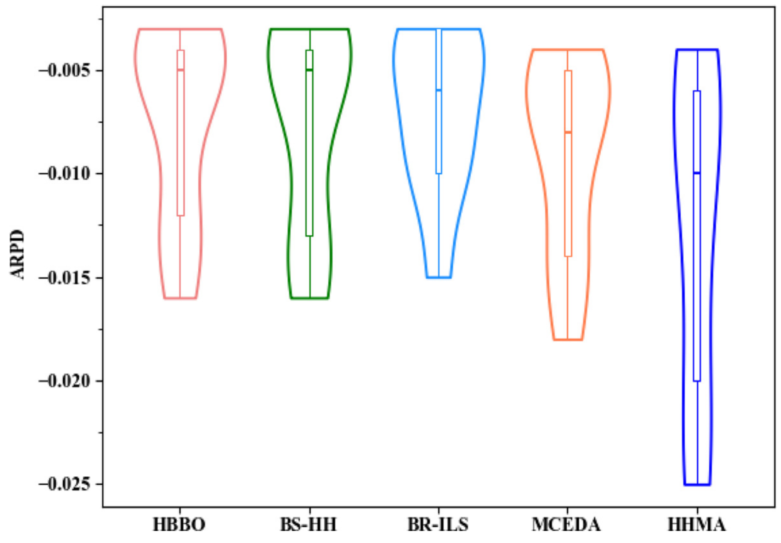

**Fig. 10.** Violin plot for ARPD of the compared algorithms.

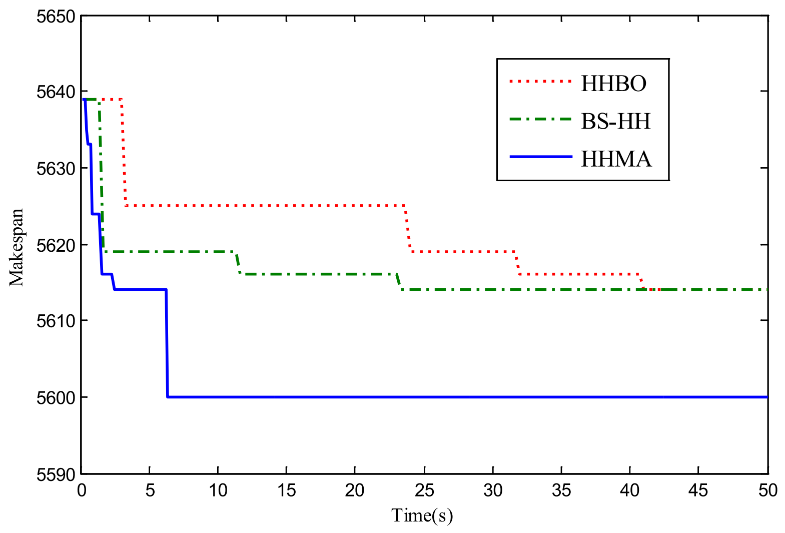

**Fig. 11.** Convergence curves for the HHMA, HBBO and BS-HH.

**Table 12**
New best updated solutions.

| Instance | Best-known | HHMA | RPD | Instance | Best-known | HHMA | RPD |
|---|---|---|---|---|---|---|---|
| I_100_20_4_30_1 | 5603 | 5600 | −0.05 | I_500_10_6_30_3 | 28 515 | 28 502 | −0.05 |
| I_200_5_4_30_10 | 10 754 | 10 752 | −0.02 | I_500_10_6_30_7 | 22 637 | 22 628 | −0.04 |
| I_200_5_4_40_3 | 10 426 | 10 420 | −0.06 | I_500_10_6_30_8 | 20 880 | 20 863 | −0.08 |
| I_200_5_4_50_4 | 9390 | 9389 | −0.01 | I_500_10_6_30_9 | 30 479 | 30 465 | −0.05 |
| I_500_5_4_30_2 | 25 024 | 25 017 | −0.03 | I_500_10_6_30_10 | 27 992 | 27 984 | −0.03 |
| I_500_5_4_30_3 | 24 794 | 24 793 | −0.00 | I_500_10_6_50_10 | 26 909 | 26 902 | −0.03 |
| I_500_5_4_30_8 | 23 770 | 23 768 | −0.01 | I_500_10_8_30_4 | 27 899 | 27 896 | −0.01 |
| I_500_5_4_30_9 | 20 571 | 20 569 | −0.01 | I_500_10_8_40_10 | 24 050 | 24 043 | −0.03 |
| I_500_5_4_40_3 | 25 913 | 25 911 | −0.01 | I_500_20_4_30_7 | 22 293 | 22 290 | −0.01 |
| I_500_5_4_40_6 | 29 693 | 29 680 | −0.07 | I_500_20_4_30_8 | 27 499 | 27 492 | −0.03 |
| I_500_5_4_50_2 | 27 780 | 27 776 | −0.01 | I_500_20_4_40_2 | 25 377 | 25 376 | −0.00 |
| I_500_5_4_50_6 | 24 492 | 24 491 | −0.00 | I_500_20_4_50_5 | 23 663 | 23 656 | −0.03 |
| I_500_5_6_30_4 | 28 179 | 28 176 | −0.01 | I_500_20_6_30_1 | 24 475 | 24 474 | −0.00 |
| I_500_5_6_40_2 | 22 356 | 22 350 | −0.03 | I_500_20_6_30_6 | 26 341 | 26 335 | −0.02 |
| I_500_5_6_40_5 | 26 763 | 26 762 | 0.00 | I_500_20_6_30_10 | 28 608 | 28 577 | −0.11 |
| I_500_5_6_40_8 | 19 702 | 19 691 | −0.06 | I_500_20_6_40_2 | 25 589 | 25 578 | −0.04 |
| I_500_5_8_30_9 | 24 883 | 24 879 | −0.02 | I_500_20_6_40_4 | 28 734 | 28 715 | −0.07 |
| I_500_10_4_30_2 | 22 016 | 22 008 | −0.04 | I_500_20_6_50_1 | 24 555 | 24 547 | −0.03 |
| I_500_10_4_40_4 | 22 892 | 22 888 | −0.02 | I_500_20_8_30_8 | 22 571 | 22 564 | −0.03 |
| I_500_10_4_40_8 | 24 642 | 24 638 | −0.02 | | | | |

the one that leads to the new best solution. The critical products in the product sequence are $P_5$, $P_{17}$ and $P_{12}$, respectively, followed by the non-critical products $P_9$ and so on. The sets of jobs used to assemble them are $N_5 = \{34, 42\}$, $N_{17} = \{57, 72, 88\}$, $N_{12} = \{43, 78\}$ and $N_9 = \{19, 35, 45, 54\}$, respectively. Before CP-RLS, we first change the solution representation to FPR and obtain the sub-job-sequence $\theta = [34, 42, 57, 72, 88, 43, 78]$ for the critical products. Clearly the jobs belonging to the same product are still together at present. Then, performing the operation of CP-RLS on the solution leads to a new best solution with makespan being 5600 and $\theta = [43, 42, 34, 72, 88, 57, 78]$. Adding $P_9$ into local search makes no more improvement for the makespan and the local search stops accordingly. The resulting sub-job-sequence in the solution is $\theta' = [\theta, 54, 45, 19, 35]$. We can clearly see that, after CP-RLS, the jobs belonging to the critical products are separated while the jobs belonging the non-critical products such as $P_9$ are not. If MPR is used during the whole procedure of HHMA, then this new best solution 5600 will not be found as in [17–19]. On the other hand, adopting FPR in both global and local search will make the search not efficient enough to find the solution with reasonable CPU times as in [16,20]. Thus, the effectiveness of our proposed encoding scheme can further be verified by this example.

## 6. Conclusions

In this paper, an effective HHMA was proposed to solve the DAPFSP with makespan criterion. The key critical-products-based knowledge is that any operations within the non-critical-products and the jobs belonging to them have no effects on the makespan reduction. In the proposed HHMA, potential product sequences were firstly obtained by EDAHH and a CP-RLS was then presented to fully exploit the corresponding sub-job-sequences. Moreover, a search-stage-based solution representation scheme was presented to both improve search efficiency and maintain potential solutions. Both high level strategy and the LLH set of EDAHH were well designed so that the exploration ability can be guaranteed. The issue of premature convergence was relieved by embedding a simulated-annealing-like type of acceptance criterion into each LLH. The CP-RLS was designed with acceptable computational cost. Based on the benchmark instance set, both numerical simulation and comparison were carried out to show the effectiveness of the proposed HHMA. Compared to the state-of-the-art algorithms, the performance of the HHMA was shown to be superior. Additionally, 39 new best solutions were updated for the reported results.

As for the future research directions, it is of practical interest to consider the transportation cost among the distributed production factories and the assembly one. The integration of distributed scheduling and vehicle routing deserves further investigation. Additionally, DAPFSP with multiple assembly factories or sequence dependent setup times are also interesting research topics. Furthermore, dynamic scheduling for the DAPFSP with machine and factory breakdown or urgent product insertion also deserves investigation.

## CRediT authorship contribution statement

**Hong-Bo Song:** Conceptualization, Methodology, Writing – original draft, Supervision. **You-Hong Yang:** Software, Investigation, Formal analysis, Writing – review & editing. **Jian Lin:** Validation, Writing – review & editing, Funding acquisition, Project administration. **Jing-Xuan Ye:** Resources, Data curation, Visualization.

## 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

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is part of a project supported by the National Natural Science Foundation of China (Grant Nos. 61973267 and 61503331), the Zhejiang Provincial Natural Science Foundation of China (Grant Nos. LY22F030020 and LY19F030007).

## References

[1] S. Hatami, R. Ruiz, C. Andrés-Romano, The distributed assembly permutation flowshop scheduling problem, Int. J. Prod. Res. 51 (2013) 5292–5308.
[2] X.L. Wu, X.J. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, Memet. Comput. 11 (2019) 335–355.

[3] Y. Fu, Y. Hou, Z. Wang, X. Wu, L. Wang, Distributed scheduling problems in intelligent manufacturing systems: a review, Tsinghua Sci. Tech. 26 (2021) 625–645.
[4] L. Wang, W. Shen, Process Planning and Scheduling for Distributed Manufacturing, Springer, London, 2007.
[5] G. Komaki, S. Sheikh, B. Malakooti, Flow shop scheduling problems with assembly operations: a review and new trends, Int. J. Prod. Res. 57 (2019) 2926–2955.
[6] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristics, European J. Oper. Res. 165 (2005) 479–494.
[7] V. Fernandez-Viagas, R. Ruiz, J.M. Framinan, A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation, European J. Oper. Res. 257 (2017) 707–721.
[8] E. Ignall, L. Schrage, Application of the branch-and-bound technique to some flowshop scheduling problems, Oper. Res. 13 (1965) 400–412.
[9] T. Gonzalez, S. Sahni, Flowshop and jobshop schedules: complexity and approximation, Oper. Res. 26 (1978) 36–52.
[10] I.H. Osman, C.N. Potts, Simulated annealing for permutation flow-shop scheduling, Omega 17 (1989) 551–557.
[11] B. Liu, L. Wang, Y.-H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, IEEE Trans. Syst. Man Cybern. B 37 (2007) 18–27.
[12] Q.-K. Pan, M.F. Tasgetiren, Y.-C. Liang, A discrete differential evolution algorithm for the permutation flowshop scheduling problem, Comput. Ind. Eng. 55 (2008) 795–816.
[13] Y.F. Liu, S.Y. Liu, A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem, Appl. Soft Comput. 13 (2013) 1459–1463.
[14] A.P. Rifai, H.T. Nguyen, S.Z.M. Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, Appl. Soft Comput. 40 (2016) 42–57.
[15] I. Benkalai, D. Rebaine, P. Baptiste, Scheduling flow shops with operators, Int. J. Prod. Res. 57 (2019) 338–356.
[16] S.Y. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans. Syst. Man Cybern. 46 (2016) 139–149.
[17] J. Lin, S. Zhang, An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem, Comput. Ind. Eng. 97 (2016) 128–136.
[18] J. Lin, Z.-J. Wang, X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, Swarm Evol. Comput. 36 (2017) 124–135.
[19] D. Ferone, S. Hatami, E.M. Gonzalez, A.A. Juan, P. Festa, A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem, Int. Trans. Oper. Res. 27 (2020) 1368–1391.
[20] Z.Q. Zhang, B. Qian, R. Hu, H.-P. Jin, L. Wang, A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem, Swarm Evol. Comput. 60 (2021) 100785.
[21] X. Li, X. Zhang, M. Yin, J. Wang, A genetic algorithm for the distributed assembly permutation flowshop scheduling problem, in: Evolutionary Computation (CEC), Proceeding of the 2015 IEEE Congress on IEEE, 2015, pp. 3096–3101.
[22] A. Hamzadayı, M.A. Arvas, A. Elmi, Distributed assembly permutation flow shop problem; Single seekers society algorithm, J. Manuf. Syst. 61 (2021) 613–631.
[23] H.Y. Sang, Q.-K. Pan, J.-Q. Li, P. Wang, Y.-Y. Han, K.-Z. Gao, P. Duan, Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, Swarm Evol. Comput. 44 (2019) 64–73.
[24] Y.Y. Huang, Q.-K. Pan, J.P. Huang, P.N. Suganthan, L. Gao, An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem, Comput. Ind. Eng. 152 (2020) 107021.
[25] S. Hatami, R. Ruiz, C. Andrés-Romano, Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times, Int. J. Prod. Econ. 169 (2015) 76–88.
[26] H.-B. Song, J. Lin, A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times, Swarm Evol. Comput. 60 (2021) 100807.
[27] Q.H. Nguyen, Y.S. Ong, M.H. Lim, A probabilistic memetic framework, IEEE Trans. Evol. Comput. 13 (2009) 604–623.
[28] L.Y. Tseng, S.C. Chen, Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem, IEEE Trans. Evol. Comput. 13 (2009) 848–857.
[29] Y. Mei, K. Tang, X. Yao, A memetic algorithm for periodic capacitated arc routing problem, IEEE Trans. Syst. Man Cybern. B 41 (2011) 1654–1667.
[30] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, Springer, Berlin Heidelberg, 2012.
[31] J. Branke, S. Nguyen, C.W. Pickardt, M. Zhang, Automated design of production scheduling heuristics: a review, IEEE Trans. Evol. Comput. 20 (2016) 110–124.
[32] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcanc, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (2013) 1695–1724.
[33] J.H. Drake, A. Kheiri, E. Özcanc, E.K. Burke, Recent advances in selection hyper-heuristics, European J. Oper. Res. 285 (2020) 405–428.
[34] C.Y. Lee, T. Cheng, B. Lin, Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem, Manage. Sci. 39 (1993) 616–625.
[35] C.N. Potts, S.V. SevastJanov, V.A. Strusevich, L.N. Van Wassenhove, C.M. Zwaneveld, The two-stage assembly scheduling problem: complexity and approximation, Oper. Res. 43 (1995) 346–355.
[36] C. Koulamas, G.J. Kyparisis, The three-stage assembly flowshop scheduling problem, Comput. Oper. Res. 28 (2001) 689–704.
[37] A. Mozdgir, S.M.T. Fatemi Ghomi, F. Jolai, J. Navaei, Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times, Int. J. Prod. Res. 51 (2013) 3625–3642.
[38] F.S. Al-Anzi, A. Allahverdi, A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times, European J. Oper. Res. 182 (2007) 80–94.
[39] J. Behnamian, S.M.T. Fatemi Ghomi, A survey of multi-factory scheduling, J. Intell. Manuf. 27 (2016) 231–249.
[40] M.A. Beheshtinia, A. Ghasemi, M. Farokhnia, Supply chain scheduling and routing in multi-site manufacturing system (case study: a drug manufacturing company), J. Model. Manag. 13 (2018) 27–49.
[41] M.A. Beheshtinia, A. Ghasemi, A multi-objective and integrated model for supply chain scheduling optimization in a multi-site manufacturing system, Eng. Optim. 50 (2018) 1415–1433.
[42] L.D. Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, European J. Oper. Res. 200 (2010) 395–408.
[43] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (2010) 754–768.
[44] Y.Z. Li, Q.K. Pan, K.Z. Gao, M.F. Tasgetiren, J.Q. Li, A green scheduling algorithm for the distributed flowshop problem, Appl. Soft Comput. 109 (2021) 107526.
[45] F.L. Xiong, K.Y. Xing, Meta-heuristics for the distributed two-stage assembly scheduling problem with bi-criteria of makespan and mean completion time, Int. J. Prod. Res. 52 (2014) 2743–2766.
[46] F.L. Xiong, K.Y. Xing, W. Feng, L. Hang, L. Han, Minimizing the total completion time in a distributed two stage assembly system with setup times, Comput. Oper. Res. 47 (2014) 92–105.
[47] J. Deng, L. Wang, S.Y. Wang, X.L. Zheng, A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem, Int. J. Prod. Res. 54 (2017) 1–17.
[48] Q.-K. Pan, G.C. Liang, X. Li, F.M. Jose, Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem, Appl. Soft Comput. 81 (2019) 105492.
[49] S. Yang, Z. Xu, The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery, Int. J. Prod. Res. 59 (2020) 4053–4071.
[50] J. Wang, D.M. Lei, J.C. Cai, An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance, Appl. Soft Comput. 117 (2022) 108371.
[51] F.Q. Zhao, J.L. Zhao, L. Wang, J.X. Tang, An optimal block knowledge driven backtracking search algorithm for distributed assembly no-wait flow shop scheduling problem, Appl. Soft Comput. 112 (2021) 107750.
[52] R. Ruiz, C. Maroto, J. Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, Omega 34 (2006) 461–476.
[53] D.C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, New Jersey, 2008.