An Estimation of Distribution Algorithm-Based Memetic Algorithm for the Distributed Assembly Permutation Flow-Shop Scheduling Problem

Sheng-Yao Wang and Ling Wang

***Abstract*—In this paper, an estimation of distribution algorithm (EDA)-based memetic algorithm (MA) is proposed for solving the distributed assembly permutation flow-shop scheduling problem (DAPFSP) with the objective to minimize the maximum completion time. A novel bi-vector-based method is proposed to represent a solution for the DAPFSP. In the searching phase of the EDA-based MA (EDAMA), the EDA-based exploration and the local-search-based exploitation are incorporated within the MA framework. For the EDA-based exploration phase, a probability model is built to describe the probability distribution of superior solutions. Besides, a novel selective-enhancing sampling mechanism is proposed for generating new solutions by sampling the probability model. For the local-search-based exploitation phase, the critical path of the DAPFSP is analyzed to avoid invalid searching operators. Based on the analysis, a critical-path-based local search strategy is proposed to further improve the potential solutions obtained in the EDA-based searching phase. Moreover, the effect of parameter setting is investigated based on the Taguchi method of design-of-experiment. Suitable parameter values are suggested for instances with different scales. Finally, numerical simulations based on 1710 benchmark instances are carried out. The experimental results and comparisons with existing algorithms show the effectiveness of the EDAMA in solving the DAPFSP. In addition, the best-known solutions of 181 instances are updated by the EDAMA.**

***Index Terms*—Critical path, distributed production scheduling, estimation of distribution algorithm (EDA), memetic algorithm (MA).**

I. Introduction

PRODUCTION scheduling [1]–[3] plays an important role in the decision making of a manufacturing system. Modeling and optimization for production scheduling are significant topics in the field of system engineering. Besides, effective and efficient algorithms for scheduling problems can improve the efficiency of the manufacturing process. Therefore, it is important to study production scheduling problems, especially when developing effective solution algorithms.

As one of the most general scheduling problems, the assembly permutation flow-shop scheduling problem (APFSP) exists in many manufacturing systems. After the pioneer work of Lee *et al.* [4] and Potts *et al.* [5], in recent years the APFSP has been widely studied. Koulamas and Kyparisis [6] considered the collection and transportation operation between the production stage and assembly stage. They analyzed the worst-case ratio bound of several heuristics and the worst-case absolute performance bound for an asymptotically optimal heuristic based on compact vector summation techniques. Aiming at minimizing the total weighted flowtime, Tozkapan *et al.* [7] incorporated a lower bounding procedure and a dominance criterion into a branch and bound procedure. They also used a heuristic procedure to derive an initial upper bound. Allahverdi and Al-Anzi [8] considered the APFSP with respect to a maximum lateness performance measure and proposed three heuristics. Then, they considered the setup times of the operations and proposed a self-adaptive differential evolution heuristic [9]. Mozdgir *et al.* [10] addressed the APFSP with multiple nonidentical assembly machines to minimize the weighted sum of makespan and mean completion time. They developed a hybrid algorithm by combining the variable neighborhood search algorithm and a heuristic.

The above research makes a common assumption that all jobs are assumed to be processed and assembled in the same factory, which means one production center environment. Nevertheless, with the development of the business concept, multiplant companies and supply chains are taking a more important role in practice [11]. Besides, coproduction between companies becomes more and more common nowadays [12]. The distributed manufacturing strategy enables companies to achieve higher product quality, lower production costs, and lower management risks [13].

A. Literature on Distributed Production Scheduling

Scheduling for distributed production systems is more difficult than classical shop scheduling. It should determine the factory assignment of jobs as well as the processing sequences in all the factories. Obviously, both sub-problems

Manuscript received November 12, 2014; revised January 23, 2015; accepted February 22, 2015. Date of publication April 3, 2015; date of current version December 14, 2015. This work was supported in part by the National Key Basic Research and Development Program of China under Grant 2013CB329503, in part by the National Science Foundation of China under Grant 61174189, and in part by the Doctoral Program Foundation of Institutions of Higher Education of China under Grant 20130002110057. This paper was recommended by Associate Editor N. Wu.

The authors are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: wangshengyao10@mails.tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2416127

are related to each other and cannot be solved sequentially if high performance is desired [14]. Compared to classical shop scheduling, literature on distributed production scheduling is relatively limited. Jia *et al.* [15], [16] studied the distributed job shop scheduling problem under different criteria by employing a standard genetic algorithm (GA). Later, Jia *et al.* [17] refined the previous GA to solve small- and medium-scaled distributed scheduling problems. Chan *et al.* [18] proposed an adaptive GA to solve large-scaled distributed job shop scheduling problems with makespan as criterion. Then, Chan *et al.* [19] presented a GA with dominant genes approach to deal with distributed flexible manufacturing system scheduling problems subject to machine maintenance constraint. De Giovanni and Pezzella [20] proposed an improved GA to solve a distributed and flexible job-shop scheduling problem (FJSP). Naderi and Ruiz [14] addressed a distributed permutation flow-shop scheduling problem (DPFSP) and presented 14 heuristics based on constructive heuristics and variable neighborhood descent (VND) methods. After the pioneer work of Naderi and Ruiz [14], some intelligent optimization algorithms have been proposed for the DPFSP, such as Tabu search [21], iterated greedy algorithm [22], and estimation of distribution algorithm (EDA) [23]. The first effort that considered the assembly flowshop problem in a distributed manufacturing environment is made by Hatami *et al.* [24]. They presented a mixed integer linear programming model for a distributed APFSP (DAPFSP) and designed a VND algorithm accordingly.

B. Literature on Memetic Algorithms and EDA

The memetic algorithms (MAs) [25], [26] have gained increasing attention and wide applications during recent years [27]–[33]. With the MA framework, the performances of evolutionary algorithms can be improved by combining problem-dependent local searches [26]. Liu *et al.* [27] proposed a particle swarm optimization (PSO)-based MA for a permutation flow-shop scheduling problem and discussed the effects of different local searches on optimization performances. Tseng and Chen [28] presented an MA by combining GA with a local search method to solve a multimode resource-constrained project scheduling problem (RCPSP). To solve the real-time multirobot path-planning problem, Rakshit *et al.* [29] proposed an adaptive MA by using differential evolution for global search and Q-learning for local refinement. Mei *et al.* [30] proposed an MA for the periodic capacitated arc routing problem, where a route-merging procedure was devised and embedded to tackle the insensitive objective. Liu *et al.* [31] proposed a new MA for multiobjective optimization by combining the global search ability of PSO with a synchronous local search heuristic for directed local fine-tuning. Tang and Yao [32] presented an MA for an integrated-circuit floorplanning problem. Hrncic *et al.* [33] applied an MA for grammatical inference in the field of domain-specific languages.

As a population-based evolutionary algorithm, the EDA [34] is a hot research topic in the field of evolutionary computation. The main characteristic of the EDA is its explicit probability model, which is employed to perform optimization procedure [34]. The probability model represents the probability distribution of promising solutions. At each generation, the new solutions are generated by sampling the probability model. To trace a more promising searching area, the probability model is adjusted by information of some superior solutions in the population. In such an iterative way, it evolves and obtains satisfactory solutions. The EDA has already been successfully applied to solve a variety of academic and engineering optimization problems [35]. As for scheduling problems, the EDAs have been proposed for solving the RCPSP [36], flow-shop scheduling problem [37], single machine scheduling problem [38], hybrid flow-shop scheduling problem (HFSP) [39], FJSP [40], nurse scheduling problem [41], and so on. The EDAs stress more on global exploration while their local exploitation needs to be enhanced. For this purpose, the MA framework is expected to be an effective method.

C. Introduction of This Paper

In our previous works, several EDA-based algorithms with different probability models have been proposed for the multimode RCPSP [36], HFSP [39], FJSP [40], and DPFSP [23], respectively. In this paper, an effective EDA-based MA (EDAMA) is proposed for solving the DAPFSP with the criterion of minimizing the maximum completion time. The new contributions of this paper are as follows.

1) To execute the searching operator efficiently, a novel bi-vector-based method is proposed to represent a solution for the DAPFSP. Compared with the solution representation in [23], an additional vector, i.e., the factory assignment vector is adopted in this paper to represent the corresponding factory for each job.
2) Because of an additional assembly stage, the DAPFSP is more complex than the DPFSP studied in [23]. To avoid invalid searching operators, critical path of the DAPFSP is analyzed in this paper. Accordingly, a critical-path-based local search (CPLS) strategy is proposed to perform exploitation in the EDAMA. The effectiveness of a CPLS strategy is demonstrated by comparative experiments.
3) In the DAPFSP, each job belongs to a defined product. A uniform sampling mechanism for the EDA in [23] is not helpful to start an assembly stage of products early. Therefore, a novel selective-enhancing sampling mechanism is proposed for the EDAMA in this paper. The results from the analysis of variance (ANOVA) show that the novel selective-enhancing sampling mechanism is significantly better than the uniform sampling mechanism [23].
4) In the EDAMA, the EDA-based exploration and CPLS-based exploitation are incorporated within the MA framework. Based on 1710 benchmark instances, numerical simulations and comparisons with the existing algorithms show the effectiveness of the EDAMA for

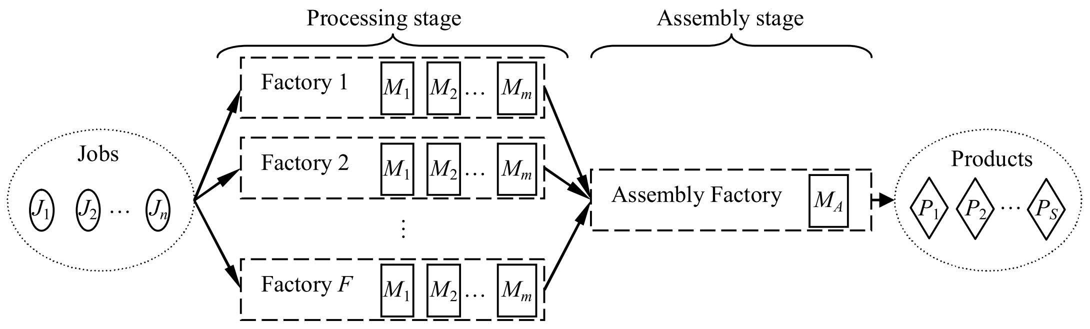

Fig. 1. Illustration of the DAPFSP.

solving the DAPFSP. In addition, the best-known solutions of 181 instances are updated by the EDAMA.

The remainder of this paper is organized as follows. In Section II, the DAPFSP is described. Then, the EDAMA for the DAPFSP is introduced in Section III. Numerical results and comparisons are provided in Section IV. Finally, the conclusion is given in Section V.

II. Distributed Assembly Permutation Flow-Shop Scheduling Problem

The DAPFSP [24] is illustrated in Fig. 1. There are n jobs $\{J_1, J_2, \ldots, J_n\}$ to be processed and assembled to produce S final products $\{P_1, P_2, \ldots, P_S\}$. Each product consists of some defined jobs and each job belongs to a defined product, i.e., $\sum_{r=1}^{S} |P_r| = n$. The production procedure consists of two stages: processing and assembly ones. There are F factories at the processing stage. Each factory contains m machines $\{M_1, M_2, \ldots, M_m\}$. Each job J_i requires a sequence of m operations $\{O_{i,1}, O_{i,2}, \ldots, O_{i,m}\}$ to be processed one after another in any one of the factories. Operation $O_{i,j}$ is executed on machine M_j with processing time $t_{i,j}$. Besides, the jobs cannot be transferred to another factory during the processing procedure. As for the assembly stage, an assembly machine M_A in an assembly factory assembles all jobs into products. After all the corresponding jobs of product P_r are finished at the processing stage, they are assembled with an assembly time t_r^A. Besides, the following assumptions for the classical flow-shop scheduling are adopted: all the jobs are independent and available at time 0. Each machine can process only one job at a time and each job can be processed by only one machine at a time. Preemption is not allowed, i.e., each operation must be completed without interruption once it is started. Setup time of machines and transportation time between operations are contained in the processing time. The DAPFSP is to assign jobs to factories and sequence jobs for all machines to optimize a certain scheduling objective. In this paper, the objective is to minimize the maximum completion time (makespan).

Denote $\boldsymbol{\lambda^k} = [\lambda^k(1), \lambda^k(2), \ldots, \lambda^k(n_k)]$ as the job sequence in factory k at the processing stage, where n_k is the total number of jobs assigned to factory k. Denotes $C_{i,j}$ as the completion time of $O_{i,j}$. For a schedule $\boldsymbol{\Lambda}$ of the DAPFSP, i.e., a set of sequences $\{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \ldots, \boldsymbol{\lambda^F}\}$, the makespan $C_{\max}(\boldsymbol{\Lambda})$ is calculated as follows:

$$C_{\lambda^k(1),1} = t_{\lambda^k(1),1}, k = 1, 2, \ldots, F \tag{1}$$

$$C_{\lambda^k(i),1} = C_{\lambda^k(i-1),1} + t_{\lambda^k(i),1} \quad k = 1, 2, \ldots, F; \quad i = 2, 3, \ldots, n_k \tag{2}$$

$$C_{\lambda^k(1),j} = C_{\lambda^k(1),j-1} + t_{\lambda^k(1),j} \quad k = 1, 2, \ldots, F; \quad j = 2, 3, \ldots, m \tag{3}$$

$$C_{\lambda^k(i),j} = \max\{C_{\lambda^k(i-1),j}, C_{\lambda^k(i),j-1}\} + t_{\lambda^k(i),j} \quad k = 1, 2, \ldots, F; \; i = 2, 3, \ldots, n_k; \; j = 2, 3, \ldots, m. \tag{4}$$

Then, the product sequence on the assembly machine is obtained according to the completion time of all jobs. Denote $\boldsymbol{\lambda^A} = [\lambda^A(1), \lambda^A(2), \ldots, \lambda^A(S)]$ as the product sequence and C_r^P as the latest completion time of the jobs belonging to product $\lambda^A(r)$. Denote C_r^A as the completion time of assembling product $\lambda^A(r)$

$$C_1^A = C_1^P + t_r^A \tag{5}$$

$$C_r^A = \max\left\{C_{r-1}^A, C_r^P\right\} + t_r^A, r = 2, 3, \ldots, S \tag{6}$$

$$C_{\max}(\boldsymbol{\Lambda}) = C_S^A. \tag{7}$$

The objective of solving the DAPFSP is to find a schedule $\boldsymbol{\Lambda}^*$ with the minimum makespan. However, as an NP-hard problem [24], the DAPFSP cannot be solved by any polynomial time algorithms. Therefore, the objective of the solution methods is to obtain an approximate optimal schedule within an acceptable computing time.

III. EDAMA for DAPFSP

A. Solution Representation

In EDAMA, the population is composed of a set of solutions for the DAPFSP. A solution is represented by a job priority vector $\boldsymbol{\pi}$ and a factory assignment vector $\boldsymbol{\xi}$ as shown in Fig. 2. The elements in the job priority vector are denoted by job numbers. In addition, the element $\pi(i)$ represents that job $J_{\pi(i)}$ has the processing priority of i. The elements in the factory assignment vector are denoted by factory numbers, which determine corresponding factories for processing jobs. Specifically, element $\xi(i)$ represents that job J_i is assigned to factory $\xi(i)$ at the processing stage.

For decoding a given solution into a feasible schedule, jobs are arranged to their corresponding factories in the order of the

Job priority vector	Factory assignment vector
$\boldsymbol{\pi} = [\pi(1), \pi(2), \cdots, \pi(n)]$	$\boldsymbol{\xi} = [\xi(1), \xi(2), \cdots, \xi(n)]$

Fig. 2. Solution representation in EDAMA.

TABLE I
DATA FOR THE EXAMPLE INSTANCE

Product	Job	Processing time			Assembly time
		M_1	M_2	M_3	M_A
P_1	J_4	11	60	41	48
	J_5	95	37	22	
	J_6	90	12	85	
	J_{10}	27	10	93	
	J_{11}	86	51	89	
	J_{12}	89	56	90	
P_2	J_1	92	27	88	401
	J_2	58	83	25	
	J_3	24	33	92	
	J_7	27	28	45	
	J_8	7	22	79	
	J_9	13	92	41	

Job priority vector

8	7	9	3	2	1	12	5	11	10	4	6

Factory assignment vector

1	2	2	1	1	2	2	1	1	2	1	2
J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}	J_{12}

Fig. 3. Feasible solution of the example instance.

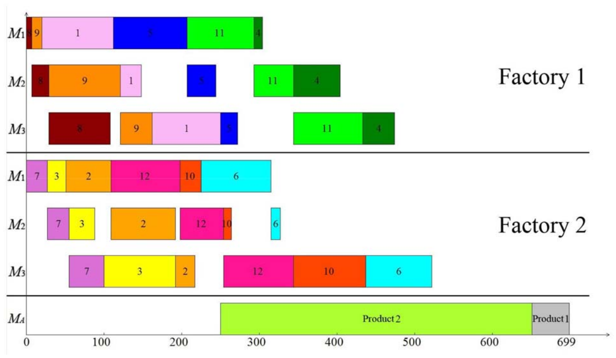

Fig. 4. Corresponding Gantt chart of the solution.

job priority vector. Then, processing sequences of jobs in all factories are obtained, i.e., $\{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \ldots, \boldsymbol{\lambda}^F\}$. The makespan value of the solution is calculated as introduced in Section II. To explain the solution representation and decoding procedure, an example instance with 12 jobs and two factories is provided in Table I. The representation of a feasible solution is illustrated in Fig. 3. Processing sequences of jobs in two factories are $\boldsymbol{\lambda}^1 = (8, 9, 1, 5, 11, 4)$ and $\boldsymbol{\lambda}^2 = (7, 3, 2, 12, 10, 6)$, respectively. The Gantt chart of the corresponding schedule is illustrated in Fig. 4.

B. Probability Model

A crucial part of the EDA is its probability model that describes the distribution of the searching space. Generally, the probability model is built based on characteristics of superior solutions. Besides, the EDA generates new solutions by sampling according to the probability model. Therefore, a proper probability model is critical to the performance of EDA-based algorithms.

In this paper, the optimization objective is to minimize the makespan for the DAPFSP. According to the above solution representation, the processing priorities of jobs affect the objective value of a solution. Therefore, the probability model is designed as a probability matrix $\boldsymbol{Q}$, which is related to the job priority vector of a solution

$$\boldsymbol{Q} = \begin{bmatrix} q_{11}(l) & q_{12}(l) & \cdots & q_{1n}(l) \\ q_{21}(l) & q_{22}(l) & \cdots & q_{2n}(l) \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1}(l) & q_{n2}(l) & \cdots & q_{nn}(l) \end{bmatrix}. \tag{8}$$

Element $q_{ij}(l)$ in the probability matrix $\boldsymbol{Q}$ represents the probability that job J_j appears before or in the ith position of the job priority vector at the lth generation. The value of $q_{ij}(l)$ refers to the importance of a job when decoding a solution into a schedule. At the initialization stage of the EDAMA, the elements in matrix $\boldsymbol{Q}$ are initialized as $q_{ij}(0) = 1/n$ (for all i and j), which implies a uniform distribution in the searching space.

C. Selective-Enhancing Sampling Mechanism

At each generation of the EDAMA, job priority vector $\boldsymbol{\pi}$ of a new solution is generated by sampling the searching space according to the probability matrix $\boldsymbol{Q}$. To be specific, it determines job number $\pi(i)$ for each position of $\boldsymbol{\pi}$ from $i = 1$ to n. In considering the decoding procedure and the optimization objective, jobs belonging to the same product are expected to be close to each other in job priority vector $\boldsymbol{\pi}$. In this way, it is helpful to start the assembly stage of the corresponding product as early as possible. Consequently, it is more probable to obtain a better makespan value. To implement this method in the sampling procedure, some elements in the probability matrix $\boldsymbol{Q}$ are selectively enhanced. Denote $p_{ij}(l)$ as the probability of selecting job J_j for the ith position. It calculates $p_{ij}(l)$ as follows:

$$p_{ij}(l) = \begin{cases} q_{ij}(l), & i = 1 \\ \dfrac{\delta(i,j) \cdot q_{ij}(l)}{\sum_{k=1}^{n} \delta(i,k) \cdot q_{ik}(l)}, & i > 1 \end{cases} \tag{9}$$

where $\delta(i, j)$ is the enhancing factor defined as follows:

$$\delta(i,j) = \begin{cases} \mu, & \text{if } J_j \text{ belongs to the same product of } J_{\pi(i-1)} \\ 1, & \text{else.} \end{cases} \tag{10}$$

Parameter μ ($\mu > 1$) can be regarded as the intensity to enhance the probability, which controls the compactness of jobs belonging to the same product. If the value of μ is too small, the corresponding jobs may not be compact so that the assembly stage of a product cannot be started early. On the other hand, a too large value of μ may cause a greedy sampling process, which results in a loss of diversity of a new population. Considering the total number of jobs to be selected in

```
Input job priority vector $\boldsymbol{\pi}$.
Output factory assignment vector $\boldsymbol{\xi}$.
Begin
For $i = 1$ to $F$
    $\xi[\pi(i)] = i$;
For $i = F + 1$ to $n$
    Find the factory $k$ that can process job $J_{\pi(i)}$ with the earliest completion time;
    $\xi[\pi(i)] = k$;
End
```

Fig. 5. Pseudocode of the ECF rule [23].

the sampling process, it sets $\mu = n$ in this paper. The effect of the sampling mechanism and the impact of μ are further investigated in Section IV. In addition, if job J_i has already been selected, the whole ith column in probability matrix $\boldsymbol{Q}$ is set as zero. Then, all the elements in $\boldsymbol{Q}$ are normalized to maintain that each row sums up to 1.

In such a sampling way, job priority vector $\boldsymbol{\pi}$ is obtained until all the job numbers are selected. Then, factory assignment vector $\boldsymbol{\xi}$ is determined according to $\boldsymbol{\pi}$ based on the earliest completion factory (ECF) rule [23]. Aiming at obtaining a small makespan value, the ECF rule arranges each job to a factory that can process it with the earliest completion time. The pseudocode of the ECF rule is illustrated in Fig. 5.

Based on the sampling mechanism, a solution is generated and decoded to calculate the makespan value. At each generation of the EDAMA, P_Size solutions are generated to form a new population.

D. Updating Mechanism

The probability model should be well adjusted to make the search procedure tract the promising searching region. As a consequence, an updating mechanism is employed to adjust the model at each generation. First, the superior sub-population that consists of SP_Size elite solutions is determined by the widely-used two-tournament selection strategy [42], where SP_Size $= \eta\% \cdot P$_Size. Then, probability matrix $\boldsymbol{Q}$ is updated based on the information of the superior sub-population and the historical information of searching. The updating process can be regarded as a kind of increased learning as follows:

$$q_{ij}(l+1) = (1-\alpha)q_{ij}(l) + \frac{\alpha}{i \times \text{SP\_Size}} \sum_{k=1}^{\text{SP\_Size}} I_{ij}^{k} \quad \forall i, j \tag{11}$$

where $\alpha \in (0, 1)$ is the learning rate of $\boldsymbol{Q}$ and I_{ij}^{k} is the following indicator function of the kth solution in the superior sub-population:

$$I_{ij}^{k} = \begin{cases} 1, & \text{if job } J_j \text{ appears before or in the } i\text{th position} \\ 0, & \text{else.} \end{cases} \tag{12}$$

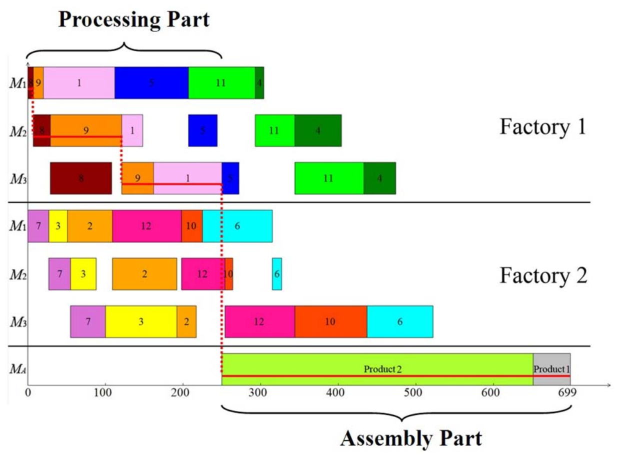

Fig. 6. Illustration of the critical path.

E. Critical-Path-Based Local Search

The EDAs pay more attention to global exploration while their exploitation capability is relatively limited. The MA framework provides an effective method to balance the exploration and exploitation abilities for EDAs. It is commonly recognized that incorporating local search into the framework of evolutionary computing can improve the optimization capability. Such idea has also been adopted in designing an MA for scheduling problems [27]. Therefore, a CPLS strategy is proposed to enhance the local exploitation of the EDA.

For shop scheduling problems, a critical path refers to a continuous job-path from the beginning to end of the solution with no idle time between any two jobs [43]. The makespan value of a solution is equal to the length of the critical path. Therefore, the only way to improve a solution is to adjust jobs of its critical path [40]. As for the DAPFSP, the critical path of a solution consists of two parts: 1) processing and 2) assembly ones. Fig. 6 illustrates the critical path of the example in Fig. 4.

In a critical path of the solution, the factory of the processing part is defined as the critical factory (e.g., factory 1 in Fig. 6). The jobs of the processing part are defined as the critical jobs (e.g., J_1, J_8, and J_9 in Fig. 6). Then, five local search operators are designed based on the critical path, including three job-based operators and two factory-based operators.

The three job-based operators adjust the job priority vector of the solution only. First, a critical job J_C and another job J_R from the critical factory are selected randomly. Then the following steps are executed.

1) *Job-Swap:* Swap the priorities of jobs J_C and J_R in the job priority vector.
2) *Job-Insert:* Insert job J_C to the position after job J_R in the job priority vector.
3) *Job-Inverse:* Invert the subsequence between the position of jobs J_C and J_R in the job priority vector.

The two factory-based operators adjust both job priority vector and factory assignment vector of a solution. First, a critical job J_C from the critical factory F_C is selected randomly. Second, another factory F_R and a job J_R from F_R are selected randomly. Then the following steps are executed.

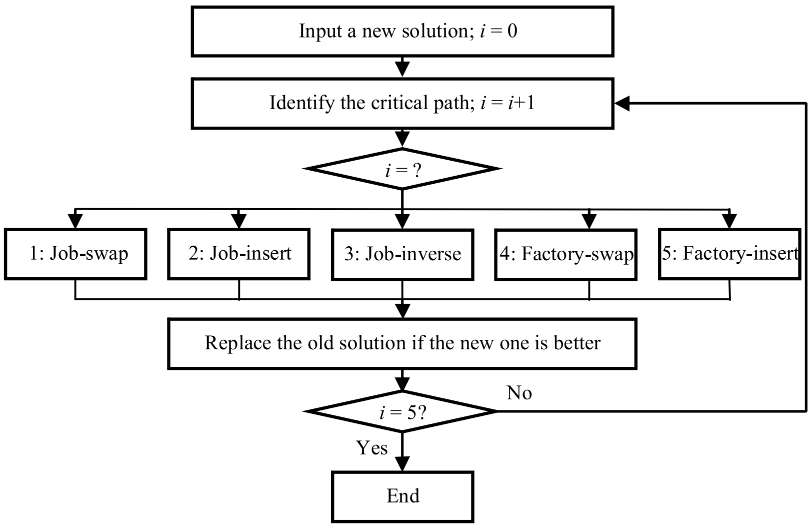
Input a new solution; $i = 0$
Identify the critical path; $i = i+1$
$i = ?$
1: Job-swap
2: Job-insert
3: Job-inverse
4: Factory-swap
5: Factory-insert
Replace the old solution if the new one is better
$i = 5?$
No
Yes
End

Fig. 7. Iteration of the CPLS strategy.

4) *Factory-Swap:* Swap the priorities of jobs J_C and J_R in the job priority vector and swap the factories of jobs J_C and J_R in the factory assignment vector.
5) *Factory-Insert:* Insert job J_C to the position after job J_R in the job priority vector and change the factory of job J_C to F_R in the factory assignment vector.

If a new solution obtained by any of the local search operators is better, the old solution is replaced. Besides, the critical path of a new solution is reidentified for the next operator. An iteration of the CPLS strategy is illustrated in Fig. 7.

At each generation of the EDAMA, the CPLS strategy is implemented by LS times to improve the best solution of the population. Considering the problem scale, the value of LS is set as $\text{LS} = \gamma \cdot n$, where γ implies the intensity of the local search.

F. EDA-Based MA

With the above design, the flowchart of the EDAMA for solving the DAPFSP is illustrated in Fig. 8.

At the initial stage of evolution process, the whole searching area is sampled uniformly. Then, the algorithm uses the EDA-based evolutionary search mechanism to sample a potential area. Moreover, a CPLS strategy is performed in a promising region, aiming to obtain better solutions. With the benefit of combining the EDA-based search and CPLS strategy, global exploration and local exploitation are balanced. As for the stopping condition of the EDAMA, the algorithm stops after a total number of Max_G solutions are generated.

G. Computational Complexity Analysis

At each generation of the EDAMA, the computational complexity can be roughly analyzed as follows.

For the updating process, first it has a computational complexity of $O(\text{SP\_Size})$ by using the tournament selection method to select the elite solutions from population; then, it has a computational complexity of $O[n(\text{SP\_Size} + n)]$ to update all the elements of $\boldsymbol{Q}$. For the sampling process,

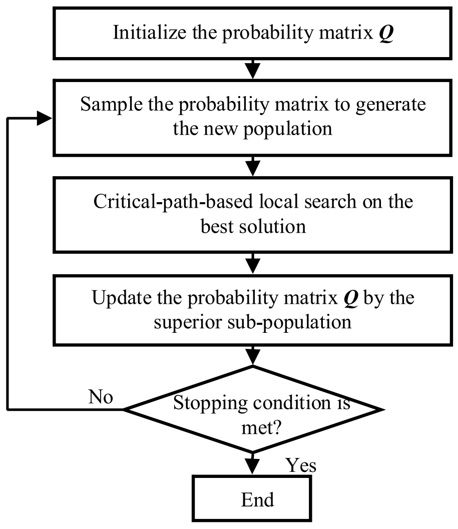
Initialize the probability matrix $\boldsymbol{Q}$
Sample the probability matrix to generate the new population
Critical-path-based local search on the best solution
Update the probability matrix $\boldsymbol{Q}$ by the superior sub-population
Stopping condition is met?
No
Yes
End

Fig. 8. Flowchart of EDAMA for DAPFSP.

it generates the job priority vector by the roulette strategy via sampling $\boldsymbol{Q}$ with a computational complexity of $O(n^2)$. Then, the factory assignment vector is determined with a computational complexity of $O(nF)$.

From the above analysis, it can be concluded that the computational complexity of the proposed EDAMA is not large.

IV. Numerical Results and Comparisons

To test the performance of the EDAMA, numerical tests are carried out by using two sets of benchmark instances [24]. The data for each instance is available at http://soa.iti.es. The first set consists of 900 small-scaled instances, where $n = \{8, 12, 16, 20, 24\}$, $m = \{2, 3, 4, 5\}$, $F = \{2, 3, 4\}$, and $S = \{2, 3, 4\}$. The second set consists of 810 large-scaled instances, where $n = \{100, 200, 500\}$, $m = \{5, 10, 20\}$, $F = \{4, 6, 8\}$, and $S = \{30, 40, 50\}$. In considering the problem scale, the value of Max_G in the stopping condition is set as 5000 and 10 000 for the first and second set, respectively.

TABLE II
PARAMETER VALUES OF EACH FACTOR LEVEL

Parameters	Factor level			
	1	2	3	4
P_Size	50	100	150	200
η	10	20	30	40
α	0.1	0.2	0.3	0.4
γ	0.25	0.50	0.75	1.00

TABLE III
ORTHOGONAL ARRAY AND RV VALUES

Experiment Number	Factor Level				RV	
	P_Size	η	α	γ	First Set	Second Set
1	1	1	1	1	1.39	0.12
2	1	2	2	2	1.46	0.13
3	1	3	3	3	1.40	0.14
4	1	4	4	4	1.25	0.19
5	2	1	2	3	1.57	0.14
6	2	2	1	4	1.47	0.18
7	2	3	4	1	1.46	0.15
8	2	4	3	2	1.45	0.14
9	3	1	3	4	1.33	0.17
10	3	2	4	3	1.69	0.13
11	3	3	1	2	2.10	0.19
12	3	4	2	1	1.83	0.18
13	4	1	4	2	1.72	0.15
14	4	2	3	1	1.79	0.15
15	4	3	2	4	1.64	0.20
16	4	4	1	3	2.16	0.20

To evaluate the performance of the EDAMA, as in [24], experimental results are evaluated by relative percentage deviation (RPD) as follows:

$$\text{RPD} = \frac{\text{alg} - \text{opt}}{\text{opt}} \times 100 \tag{13}$$

where opt is the best-known makespan from http://soa.iti.es and alg corresponds to the makespan of the solution obtained by a certain algorithm. If the obtained RPD value is less than 0, it implies that a new better solution is found.

A. Parameters Setting

The proposed EDAMA has four key parameters: 1) P_Size (population size); 2) η (percentage of the superior sub-population); 3) α (learning rate of $\boldsymbol{Q}$); and 4) γ (intensity of local search). To investigate the influence of these parameters on the performance of the EDAMA, the Taguchi method of design-of-experiment (DOE) [44] is implemented. As the stopping conditions for two sets of benchmarks are different, two DOE testings are carried out, respectively. For the first set, a moderate-scaled instance I_16_5_3_2_1 is used, where 16_5_3_2 denotes the instance scale ($n = 16, m = 5, F = 3$, and $S = 2$) and 1 denotes that it is the first instance of this scale. Similarly, instance I_200_5_6_40_7 is used for the second set.

Four factor levels are employed. Combinations of different values of these parameters are listed in Table II. Accordingly, the orthogonal array $L_{16}(4^4)$ is chosen. For each parameter combination, the EDAMA is run 20 times independently and the average RPD value of 20 runs is obtained as the response variable (RV) value. The orthogonal array and RV values are listed in Table III.

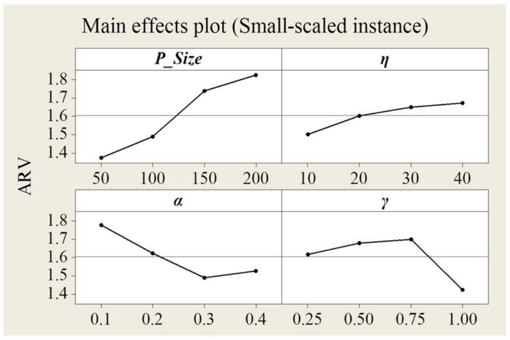

Fig. 9. Factor level trends of small-scaled instances.

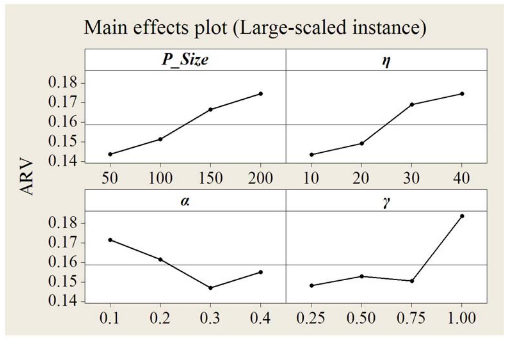

Fig. 10. Factor level trends of the large-scaled instances.

TABLE IV
RESPONSE TABLE FOR SMALL-SCALED INSTANCE

Level	*P_Size*	η	α	γ
1	**1.3729**	**1.5013**	1.7801	1.6187
2	1.4881	1.6012	1.6253	1.6802
3	1.7373	1.6495	**1.4903**	1.7011
4	1.8251	1.6714	1.5277	**1.4234**
Delta	0.4522	0.1701	0.2897	0.2777
Rank	1	4	2	3

TABLE V
RESPONSE TABLE FOR LARGE-SCALED INSTANCE

Level	*P_Size*	η	α	γ
1	**0.1436**	**0.1433**	0.1718	**0.1482**
2	0.1512	0.1490	0.1616	0.1530
3	0.1664	0.1689	**0.1471**	0.1505
4	0.1744	0.1744	0.1551	0.1839
Delta	0.0309	0.0311	0.0247	0.0357
Rank	3	2	4	1

According to the orthogonal table, trends of each factor level for different sets of benchmarks are illustrated in Figs. 9 and 10. Then, the response value of each parameter is figured out to analyze the significance rank. The results are listed in Tables IV and V.

From Fig. 9 and Table IV, it can be seen that P_Size has the most significant impact on small-scaled instances. With a fixed total number of solutions, a small value of P_Size allows sufficient evolution with more generations. The significance of learning rate α ranks the second. Small value of α could lead to slow convergence while large value could lead to premature convergence. From Fig. 10 and Table V, it can be seen that the intensity of CPLS procedure has the most

TABLE VI
SUGGESTED COMBINATIONS OF PARAMETERS

Instances	*P_Size*	η	α	γ	*Max_G*
Small-scaled	50	10	0.3	1.00	5000
Large-scaled	50	10	0.3	0.25	10000

TABLE VII
COMPARISON OF EDA WITHOUT CPLS, CPLS, AND EDAMA

n	Pure EDA		CPLS		EDAMA	
	Average RPD	CPU (ms)	Average RPD	CPU (ms)	Average RPD	CPU (ms)
8	0.11	22	0.02	22	**0.00**	22
12	0.26	37	0.08	33	**0.00**	33
16	0.35	54	0.18	42	**-0.01**	42
20	0.38	74	0.14	52	**-0.12**	52
24	0.44	97	0.14	61	**-0.23**	63
Average	0.31	57	0.11	42	**-0.07**	42

significant impact on large-scaled instances. A small value of γ also allows sufficient evolution with more generations. The significance of η ranks the second. A smaller value is helpful to update the probability model more accurately. According to the analysis, the suggested combinations of parameter values for the proposed algorithm are determined, which are listed in Table VI. This setting will also be used in the following comparison.

B. Effect of MA Method

To demonstrate the effectiveness of MA method, the pure EDA (without CPLS), CPLS strategy, and EDAMA are compared by using the first set of instances. The three algorithms have the same stopping condition. Besides, pure EDA has the same parameters as EDAMA. The CPLS strategy generates a new population randomly and implements local search procedure on the best solution at every generation. The three algorithms are run 20 times independently and the best RPD values are obtained. Table VII summarizes the results including the CPU times grouped by different values of n (180 data per average).

From Table VII, it can be seen that EDAMA is the most effective one among the three algorithms. The average RPD value obtained by EDAMA is -0.07 while pure EDA is 0.31 and CPLS is 0.11. As for the CPU time, both the EDAMA and CPLS spend an average of 42 ms to solve all instances. It takes an average of 57 ms for pure EDA, which is more than the other two algorithms. Compared with pure EDA, EDAMA is more efficient. The reason is that more solutions are generated in CPLS procedure of EDAMA at each generation. Therefore, the generation of EDAMA is less than that of pure EDA under the same stopping condition (i.e., maximum generated solution number).

In summary, within MA framework, the proposed EDAMA is more effective and efficient than pure EDA.

C. Effect of Selective-Enhancing Sampling Mechanism

Next, we investigate the effect of selective-enhancing sampling mechanism and the impact of μ by using instance I_24_5_3_2_1. The EDAMA with each different value of μ is run 20 times independently. The statistical results are listed in Table VIII, including the best, average, and worst RPD values as well as the standard deviation. Besides, the interval plots by different values of μ are illustrated in Fig. 11. Note that, the case of $\mu = 1$ means that traditional uniform sampling mechanism [23] is employed. In addition, ANOVA is carried out at 95% confidence level. The results of ANOVA are listed in Table IX.

TABLE VIII
EFFECTS OF THE SAMPLING MECHANISM

μ	Best	Average	Worst	Standard deviation
1	2.20	8.23	11.97	2.48
$0.5n$	**0**	1.45	4.67	2.03
n	**0**	**0.11**	**0.53**	**0.18**
$2n$	**0**	0.32	1.94	0.46

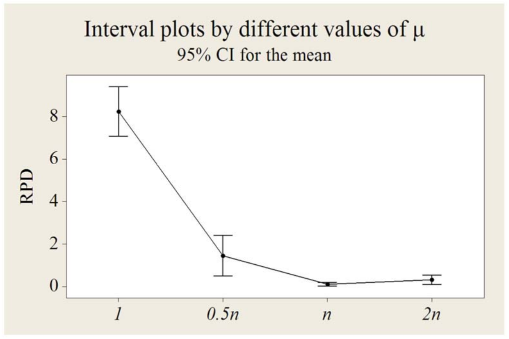

Fig. 11. Interval plots by different values of μ.

TABLE IX
ANOVA RESULTS OF THE SAMPLING MECHANISM

Source	DF	SS	MS	F	p
μ	3	887.86	295.95	112.90	0.00
Error	76	199.22	2.62		
Total	79	1087.08			

From Table VIII and Fig. 11, it can be seen that EDAMA performs better when μ is larger than 1. From Table IX, it also can be seen that p-value is $0 < 0.05$. Therefore, selective-enhancing sampling mechanism is considered to be significantly better than uniform sampling mechanism [23]. In addition, it can be seen from Fig. 11 that the value of μ should be neither too small nor too large. This is consistent with the analysis in Section III.

D. Results and Comparison for Small-Scaled Instances

By using 900 small-scaled instances, EDAMA is compared with all the existing heuristic algorithms in [24]. For each instance, EDAMA runs 20 times independently and the best RPD values are obtained. Table X summarizes the results grouped by each combination of n and F (60 data per average) as in [24], where the results of comparative algorithms are directly from literature.

From Table X, it can be seen that EDAMA is the best one among all the algorithms for small-scaled instances. The corresponding RPD values of the best solutions are nonpositive for all the instance groups except for $\{3 \times 12\}$ and $\{4 \times 16\}$.

TABLE X
RESULTS OF SMALL-SCALED INSTANCES

$F \times n$	H_{11}	H_{12}	H_{21}	H_{22}	H_{31}	H_{32}	VND_{H11}	VND_{H12}	VND_{H21}	VND_{H22}	VND_{H31}	VND_{H32}	EDAMA
2 × 8	14.62	13.61	6.91	5.99	13.55	12.17	1.00	0.76	1.00	0.76	1.02	0.78	**0.00**
2 × 12	13.70	12.78	5.74	5.17	11.58	11.05	0.93	0.87	0.93	0.87	0.93	0.87	**0.00**
2 × 16	12.52	11.40	5.77	5.10	10.00	9.16	0.73	0.55	0.72	0.53	1.09	0.53	**-0.05**
2 × 20	10.23	9.59	4.55	3.78	8.96	8.46	0.53	0.36	0.51	0.37	0.57	0.37	**-0.33**
2 × 24	8.71	8.34	5.00	4.74	7.54	7.15	0.54	0.21	0.54	0.21	0.54	0.21	**-0.48**
3 × 8	11.35	9.96	4.57	3.15	8.92	7.79	1.09	0.70	1.15	0.76	1.15	0.76	**0.00**
3 × 12	9.96	9.13	3.03	2.55	8.72	7.50	0.44	0.28	0.44	0.28	0.44	0.28	**0.01**
3 × 16	10.10	9.16	3.77	3.14	9.59	8.73	0.86	0.56	0.91	0.56	0.91	0.56	**-0.01**
3 × 20	9.86	8.93	2.72	2.19	8.53	7.84	0.43	0.43	0.43	0.43	0.43	0.43	**-0.01**
3 × 24	7.77	6.48	3.11	2.52	7.24	6.32	0.64	0.33	0.64	0.33	0.64	0.33	**-0.16**
4 × 8	9.03	8.01	2.16	1.25	6.41	5.25	1.08	0.63	0.99	0.63	0.99	0.63	**0.00**
4 × 12	5.63	4.53	1.82	1.38	4.58	3.58	0.74	0.47	0.74	0.47	0.74	0.56	**0.00**
4 × 16	7.21	6.34	2.86	2.27	6.14	5.18	0.59	0.28	0.59	0.28	0.59	0.28	**0.03**
4 × 20	6.80	6.00	2.96	2.61	5.66	5.04	1.10	0.63	1.10	0.63	1.10	0.63	**-0.01**
4 × 24	5.14	4.43	2.02	1.60	4.87	4.19	0.57	0.26	0.57	0.26	0.57	0.26	**-0.05**
Average	9.51	8.58	3.80	3.16	8.15	7.29	0.75	0.49	0.75	0.49	0.78	0.50	**-0.07**

TABLE XI
RESULTS OF LARGE-SCALED INSTANCES

		H_{11}	H_{12}	H_{21}	H_{22}	H_{31}	H_{32}	VND_{H11}	VND_{H12}	VND_{H21}	VND_{H22}	VND_{H31}	VND_{H32}	EDAMA
	4	5.57	5.09	0.32	0.19	2.96	2.56	0.06	0.03	0.05	0.01	0.05	0.01	**-0.013**
F	6	3.77	3.29	0.11	0.06	1.64	1.31	0.03	0.01	0.02	0.00	0.02	0.00	**-0.004**
	8	3.09	2.66	0.04	0.02	1.21	0.93	0.02	0.00	0.01	0.00	0.01	0.00	**-0.004**
	30	3.78	3.34	0.21	0.11	2.23	1.86	0.03	0.01	0.04	0.01	0.04	0.01	**-0.011**
s	40	4.30	3.85	0.15	0.10	1.94	1.62	0.04	0.02	0.02	0.01	0.02	0.01	**-0.008**
	50	4.36	3.85	0.11	0.05	1.65	1.32	0.04	0.01	0.02	0.00	0.02	0.00	**-0.003**
	100	6.30	5.61	0.17	0.08	2.02	1.58	0.05	0.02	0.03	0.01	0.03	0.01	**-0.008**
n	200	3.76	3.28	0.15	0.07	1.92	1.55	0.03	0.01	0.02	0.00	0.02	0.00	**-0.006**
	500	2.37	2.16	0.14	0.10	1.87	1.67	0.03	0.01	0.03	0.01	0.03	0.01	**-0.007**
Average		4.14	3.68	0.16	0.09	1.94	1.60	0.04	0.01	0.03	0.01	0.03	0.01	**-0.007**

TABLE XII
CPU TIME (SECONDS) FOR LARGE-SCALED INSTANCES

		H_{11}	H_{12}	H_{21}	H_{22}	H_{31}	H_{32}	VND_{H11}	VND_{H12}	VND_{H21}	VND_{H22}	VND_{H31}	VND_{H32}	EDAMA
	4	0.01	0.01	0.01	0.01	0.01	0.01	4.39	6.79	2.90	7.67	2.55	42.87	22.06
F	6	0.01	0.01	0.01	0.01	0.01	0.01	3.49	7.73	2.85	8.94	1.95	6.11	22.84
	8	0.01	0.01	0.01	0.01	0.01	0.01	3.26	9.56	1.86	10.21	1.83	20.64	24.58
	30	0.01	0.01	0.02	0.02	0.01	0.01	3.64	8.05	3.14	11.00	2.70	45.20	23.22
s	40	0.01	0.01	0.01	0.01	0.01	0.01	3.59	7.12	2.45	8.05	1.96	5.54	23.31
	50	0.01	0.01	0.01	0.01	0.01	0.01	3.91	8.91	2.02	7.77	1.66	18.88	22.96
	100	0.01	0.01	0.01	0.01	0.01	0.01	1.09	2.84	0.27	0.72	0.24	0.43	3.57
n	200	0.01	0.01	0.01	0.01	0.01	0.01	2.02	3.85	0.58	2.22	0.66	1.37	11.27
	500	0.03	0.02	0.03	0.04	0.03	0.02	8.03	17.39	6.76	23.88	5.41	67.81	54.65
Average		0.01	0.01	0.01	0.01	0.01	0.01	3.71	8.03	2.54	8.94	2.11	23.20	23.16

The average RPD value obtained by the EDAMA is −0.07, which implies that some of the best-known solutions are updated by EDAMA. In particular, EDAMA obtains new best makespan values for 87 small-scaled instances.

E. *Results and Comparison for Large-Scaled Instances*

Next, EDAMA is tested by using large-scaled instances. Considering number of factories, products, and jobs, a summarized result of average RPD is shown in Table XI.

Form Table XI, it can be seen that EDAMA outperforms the other algorithms in solving all the groups of large-scaled instances. In particular, EDAMA obtains new best makespan values for 94 large-scaled instances. In addition, the CPU time spent by the algorithms is listed in Table XII. The EDAMA is implemented with Intel Core i5/2.3 GHz processor. The other algorithms are carried out with Intel XEON E5420/2.5 GHz processor.

From Table XII, it can be seen that EDAMA spends more time than most heuristics. However, the efficiency of the EDAMA is at the same level of VND_{H32}. The large-scaled instances are solved by EDAMA within an average of 23.16 s. Considering the effectiveness of EDAMA, it is acceptable to spend a little more time for obtaining better solutions.

F. *Discussion of Experimental Results*

In the above sections, the performance of EDAMA is tested and compared with existing algorithms by using 900 small-scaled instances and 810 large-scaled instances. From the comparative results, it can be concluded that EDAMA outperforms the state-of-the-art in solving the benchmark instances. Especially, the best-known solutions of 87 small-scaled instances and 94 large-scaled instances are updated by EDAMA. The effectiveness of EDAMA in solving the DAPFSP owes to the following aspects.

1) The EDA-based exploration is effective by using a well-designed probability model and suitable updating mechanism. The selective-enhancing sampling mechanism is helpful to obtain a schedule with small makespan.
2) The CPLS is capable of enhancing the exploitation in the promising region. The CPLS-based exploitation is also efficient since some invalid search can be forbidden.
3) The hybridization of the EDA and CPLS within an MA framework can well balance the global and local exploitations.

V. Conclusion

In this paper, an effective EDAMA is proposed for the DAPFSP. To the best of our knowledge, this is the first reported work of the EDA-based algorithm for the DAPFSP. Within the MA framework, the EDA-based exploration and local-search-based exploitation are integrated. For the exploration phase, a probability model is built to describe the probability distribution of superior solutions. Besides, a novel selective-enhancing sampling mechanism is proposed for generating new solutions by sampling the probability model. The effectiveness of the proposed sampling mechanism is shown by comparing with uniform sampling mechanism. For the exploitation phase, a CPLS strategy is proposed to perform exploitation for potential solutions. The CPLS strategy can be adopted in other evolutionary algorithms to enhance the exploitation ability. In addition, the effect of parameter setting is investigated based on the Taguchi method of DOE. Suitable parameters are suggested for instances with different scales. Finally, numerical simulations based on 1710 benchmark instances are carried out. The results and comparison with existing algorithms show the effectiveness and efficiency of the proposed EDAMA. Moreover, the best-known solutions of 181 instances are updated. As for the future work, distributed job-shop scheduling problems as well as multiobjective scheduling problems are worth being studied based on EDA.

References

[1] B. B. Li, L. Wang, and B. Liu, "An effective PSO-based hybrid algorithm for multi-objective permutation flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 818–831, Jul. 2008.

[2] Y. Yin, M. Liu, J. Hao, and M. Zhou, "Single-machine scheduling with job-position-dependent learning and time-dependent deterioration," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 1, pp. 192–200, Jan. 2012.

[3] D. Li, M. Li, X. Meng, and Y. Tian, "A hyperheuristic approach for intercell scheduling with single processing machines and batch processing machines," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 315–325, Feb. 2015.

[4] C. Y. Lee, T. C. E. Cheng, and B. M. T. Lin, "Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem," *Manage. Sci.*, vol. 39, no. 5, pp. 616–625, May 1993.

[5] C. N. Potts, S. V. Sevast'janov, V. A. Strusevich, L. N. Van Wassenhove, and C. M. Zwaneveld, "The two-stage assembly scheduling problem: Complexity and approximation," *Oper. Res.*, vol. 43, no. 2, pp. 346–355, Mar. 1995.

[6] C. Koulamas and G. J. Kyparisis, "The three-stage assembly flowshop scheduling problem," *Comput. Oper. Res.*, vol. 28, no. 7, pp. 689–704, Jun. 2001.

[7] A. Tozkapan, O. Kirca, and C. S. Chung, "A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem," *Comput. Oper. Res.*, vol. 30, no. 2, pp. 309–320, Feb. 2003.

[8] A. Allahverdi and F. S. Al-Anzi, "A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application," *Comput. Oper. Res.*, vol. 33, no. 4, pp. 1056–1080, Apr. 2006.

[9] F. S. Al-Anzi and A. Allahverdi, "A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times," *Eur. J. Oper. Res.*, vol. 182, no. 1, pp. 80–94, Oct. 2007.

[10] A. Mozdgir, S. M. T. F. Ghomi, F. Jolai, and J. Navaei, "Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3625–3642, Jun. 2013.

[11] C. Moon, J. Kim, and S. Hur, "Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain," *Comput. Ind. Eng.*, vol. 43, no. 1, pp. 331–349, Jul. 2002.

[12] L. Wang and W. Shen, *Process Planning and Scheduling for Distributed Manufacturing*. London, U.K.: Springer, 2007.

[13] K. B. Kahn, G. A. Castellion, and A. Griffin, *The PDMA Handbook of New Product Development*. New York, NY, USA: Wiley, 2004.

[14] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, Apr. 2010.

[15] H. Z. Jia, J. Y. H. Fuh, A. Y. C. Nee, and Y. F. Zhang, "Web-based multi-functional scheduling system for a distributed manufacturing environment," *Concurr. Eng. Res. A.*, vol. 10, no. 1, pp. 27–39, Mar. 2002.

[16] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh, and Y. F. Zhang, "A modified genetic algorithm for distributed scheduling problems," *J. Intell. Manuf.*, vol. 14, nos. 13–14, pp. 351–362, Jun. 2003.

[17] H. Z. Jia, J. Y. H. Fuh, A. Y. C. Nee, and Y. F. Zhang, "Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems," *Comput. Ind. Eng.*, vol. 53, no. 2, pp. 313–320, Sep. 2007.

[18] F. T. S. Chan, S. H. Chung, and P. L. Y. Chan, "An adaptive genetic algorithm with dominated genes for distributed scheduling problems," *Expert. Syst. Appl.*, vol. 29, no. 2, pp. 364–371, Aug. 2005.

[19] F. T. S. Chan, S. H. Chung, L. Y. Chan, G. Finke, and M. K. Tiwari, "Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach," *Robot. Comput. Integr. Manuf.*, vol. 22, no. 5, pp. 493–504, Oct. 2006.

[20] L. De Giovanni and F. Pezzella, "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem," *Eur. J. Oper. Res.*, vol. 200, no. 2, pp. 395–408, Jan. 2010.

[21] J. Gao, R. Chen, and W. Deng, "An efficient Tabu search algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 3, pp. 641–651, Feb. 2013.

[22] S. W. Lin, K. C. Ying, and C. Y. Huang, "Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 5029–5038, Aug. 2013.

[23] S. Y. 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, Sep. 2013.

[24] 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, Sep. 2013.

[25] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.

[26] Q. H. Nguyen, Y. S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, Jun. 2009.

[27] B. Liu, L. Wang, and Y. H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 18–27, Feb. 2007.

[28] L. Y. Tseng and S. C. Chen, "Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 848–857, Aug. 2009.

[29] P. Rakshit *et al.*, "Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 814–831, Jul. 2013.

[30] Y. Mei, K. Tang, and X. Yao, “A memetic algorithm for periodic capacitated arc routing problem,” *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 41, no. 6, pp. 1654–1667, Dec. 2011.
[31] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, “A multiobjective memetic algorithm based on particle swarm optimization,” *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 42–50, Feb. 2007.
[32] M. Tang and X. Yao, “A memetic algorithm for VLSI floorplanning,” *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 62–69, Feb. 2007.
[33] D. Hrncic, M. Mernik, and B. R. Bryant, “Improving grammar inference by a memetic algorithm,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 5, pp. 692–703, Sep. 2012.
[34] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer, 2002.
[35] 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, Apr. 2012.
[36] L. Wang and C. Fang, “An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem,” *Comput. Oper. Res.*, vol. 39, no. 2, pp. 449–460, Feb. 2012.
[37] 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, Sep. 2009.
[38] S. H. Chen, M. C. Chen, P. C. Chang, Q. Zhang, and Y. M. Chen, “Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems,” *Expert. Syst. Appl.*, vol. 37, no. 9, pp. 6441–6451, Sep. 2010.
[39] S. Y. Wang, L. Wang, M. Liu, and Y. Xu, “An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines,” *Int. J. Adv. Manuf. Technol.*, vol. 68, nos. 9–12, pp. 2043–2056, Oct. 2013.
[40] L. Wang, S. Y. 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, May 2012.
[41] U. Aickelin and J. Li, “An estimation of distribution algorithm for nurse scheduling,” *Ann. Oper. Res.*, vol. 155, no. 1, pp. 289–309, Nov. 2007.
[42] T. Saraç and A. Sipahioglu, “Generalized quadratic multiple knapsack problem and two solution approaches,” *Comput. Oper. Res.*, vol. 43, pp. 78–89, Mar. 2014.
[43] J. Grabowski and M. Wodecki, “A very fast Tabu search algorithm for the permutation flow shop problem with makespan criterion,” *Comput. Oper. Res.*, vol. 31, no. 11, pp. 1891–1909, Sep. 2004.
[44] D. C. Montgomery, *Design and Analysis of Experiments*. Hoboken, NJ, USA: Wiley, 2005.

Sheng-Yao Wang received the B.Sc. degree from Tsinghua University, Beijing, China, in 2010, where he is currently pursuing the Ph.D. degree.

His current research interests include manufacturing scheduling and intelligent optimization methods.

Ling Wang received the B.Sc. and Ph.D. degrees 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. His current research interests include intelligent optimization and scheduling. He has authored five academic books and over 200 refereed papers.

Prof. Wang was a recipient of the Outstanding Paper Award at the International Conference on Machine Learning and Cybernetics in 2002, the Best Paper Award at the International Conference on Intelligent Computing in 2011, the Top Cited Article Award by the Engineering Applications of Artificial Intelligence (Elsevier), 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 (MOE) of China. He was the Rising Star of Science and Technology of Beijing City in 2004, the Academic Young Talent of Tsinghua University in 2009, and the Program for New Century Excellent Talents in University by the MOE of China in 2009. He is the Co-Editor-in-Chief for the *Open Operational Research Journal*. He was a reviewer for several IEEE journals such as the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON NEURAL NETWORKS, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. He is an Editorial Board Member for several journals, including *Memetic Computing*, *Swarm and Evolutionary Computation*, the *International Journal of Automation and Control*, and the *International Journal of Artificial Intelligence and Soft Computing*.