Scheduling of Stochastic Distributed Assembly Flowshop under Complex Constraints *

Xiying Du
School of Mathematical Science
Peking University
Beijing, China
1300010734@pku.edu.cn

Mengchen Ji, Zhengyang Li, Bo Liu
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
Beijing, China
bliu@amss.ac.cn

***Abstract*—The distributed assembly permutation flowshop scheduling problem (DAPFSP) is a kind of typical NP-hard combinatorial optimization problem, and represents important area in operational research. This paper presents a new type of DAPFSP with four realistic extensions, i.e., random nature of processing times and assembly times, stochastic sequence-dependent setup times (SDST) on processing machines, stochastic job release times, as well as the no-wait constraint in the processing stage. To address the above mentioned stochastic version of DAPFSP with complex constraints, our previously proposed optimization framework, labeled as PSOSAHT which is characterized by PSO-based exploration, SA-based local search and HT method for evaluating and comparing the stochastic makespan, is adapted and investigated. Computational tests are conducted on benchmark problems, demonstrating the effectiveness and efficiency of the PSOSAHT method. To the best of knowledge, it is the first attempt to study the DAPFSP with job release time and sequence-dependent setup time both of which are stochastic.**

Keywords—combinatorial optimization; distributed assembly permutation flowshop scheduling; memetic algorithm; particle swarm optimization; sequence dependent setup time; release time; stochastic; no-wait constraint

I. INTRODUCTION

With the advent of globalization, distributed production and scheduling has simulated the interests both from the academy and industry. The distributed permutation flowshop scheduling problem (DPFSP), as an extension of the well discussed permutation flowshop scheduling problem (PFSP), was proposed by Naderi and Ruiz in 2010 [1]. In DPFSP, there is a set of identical factories, each of which functions as a permutation flowshop. Wang et al. proposed a fuzzy logic-based hybrid estimation of distribution algorithm for DPFSP under machine breakdown [3]. Companys and Ribas proposed constructive procedures for DPFSP with blocking [4], [5]. Fernandez-Viagas and Framinan proposed a bounded-search iterated greedy algorithm for DPFSP [13]. Komaki et al. introduced a Variable Neighborhood Search for DPFSP with respect to minimizing makespan [14]. Rifai et al. examined a kind of multi-objective DPFSP in which jobs may need to reenter the flowshop to be produced more than once [6]. Deng et al. proposed a competitive memetic algorithm for multi-objective DPFSP distributed flow shop [8], [9]. As for the computing complexity, even when the number of machines is larger than 2, PFSP is a NP-hard problem [23]. Since DPFSP could be reduced to PFSP when there is only one factory, DPFSP is also NP-hard [1].

To make DPFSP closer to the real production, Hatami et al. proposed the distributed assembly permutation flowshop scheduling problem (DAPFSP) in which an assembly stage is supplemented as the final processing stage after the DPFSP [2]. Several studies have been investigated for variants of DAPFSP by proposing effective and efficient algorithms. For instance, via a competitive memetic algorithm, Deng et al. addressed the distributed two-stage assembly flow-shop scheduling problem in which every factory includes an assembly machine and the assembly procedure is distributed too [7]. Hatami et al. investigated the DAPFSP with sequence dependent setup times (SDST) based on constructive heuristics [10]. Furthermore Hatami et al. discussed the above problem by Variable Neighborhood Search as well as the Iterated Greedy algorithm [11]. Ji el al. discussed the DAPFSP with stochastic processing and assembly times and no wait constraint [12].

In real-practice production and manufacture, processing time of same jobs on same machines differs, so do the assembly time, setup time and job release time. Thus assuming the aforementioned four categories of time to be stochastic could make the model more realistic. In reality, after processing a job, a machine often needs some time to prepare for processing the next job, i.e., setup time which usually depends on the former and the latter work is needed [18]. Meanwhile, it's common in modern complex industrial production process that the time for jobs to get prepared to enter a shop is different, i.e., the release time [18]. In this paper, we also consider the no-wait constraint which arises from specific requirements of the production process or from the absence of adequate storage capacity between the operations of a job [17], [19]. Thus, in the context of this constraint, a job must be processed from start to completion, without any interruption either on or between machines [20-22].

To date, there has been a lack of research study on DAPFSP under the more complex constraints. The objective of this investigation is explicitly set out to fulfill this role. In this study, a new type of DAPFSP, labeled as DAPFSP_E, is presented with four realistic extensions, i.e., random nature of processing times and assembly times, stochastic sequence-

This research is partially supported by National Natural Science Foundation of China (Grant No. 71101139 and 71390331), National Science Fund for Distinguished Young Scholars of China (Grant No. 61525304), Defense Industrial Technology Development Program, and Key Research Program of Frontier Sciences, Chinese Academy of Sciences (QYZDB-SSW-SYS020).

dependent setup times (SDST) on processing machines, stochastic job release times, as well as the no-wait constraint in the processing stage. From the aspect of optimization, to well solve the DAPFSP_E, besides the good balance between global exploration and local exploitation, the optimization algorithm should be equipped with the ability of dealing with uncertainty during search. In this study, we adapt our previously proposed particle swarm optimization (PSO) based hybrid method, named PSOSAHT [15] for this complex scheduling problem. In PSOSAHT, PSO, simulated annealing (SA), and hypothesis test (HT) were well integrated for solving optimization problems with uncertainty. This algorithm has been utilized in [12] to solve no-wait stochastic DAPFSP.

The structure of the paper is as follow. In Section II, DAPFSP_E is presented. In Section III, particle swarm optimization (PSO), simulated annealing (SA) and hypothesis test (HT) are briefly reviewed. In Section IV, the PSOSAHT for DAPFSP_E is proposed. In Section V, computation tests are conducted. Finally, we conclude our study in Section VI.

II. Stochastic Distributed Assembly Flowshop under Complex Constraints

A. Problem Discription

In this section, we provide the problem features for DAPFSP_E, which is DAPFSP with four realistic extensions, i.e., random nature of processing times and assembly times, stochastic sequence-dependent setup times (SDST) on processing machines, stochastic job release times, as well as the no-wait constraint in the processing stage.

- The problem consists of a processing stage and an assembly stage. In the processing stage there are F identical factories $\{1,..,F\}$. In each factory there is an ordered sequence of M machines $\{1,...,M\}$. A set of N jobs $\{1,...,N\}$ have to be assigned to one of the factories and processed by all the machines in the factory following the predetermined order. Each factory is able to process any of the jobs. In the second stage i.e. the assembly stage, there is a single machine. The N jobs should be assembled into FP products $\{1,...,FP\}$ by that machine, following a given program. Let J_h denote the set of jobs belonging to the product h. Every job belongs to a product, thus $\bigcup J_h = \{1,...,N\}$.
- A processing machine can only process one job at a time. The processing time of job i at machine j is denoted by $p_{i,j}$. After processing a job, a machine needs some time to prepare for processing the next job. The setup time of machine j between processing job i_1 and job i_2 is denoted by $s_{i_1,i_2,j}$, which depends on the former and the latter jobs. There is also an initial setup time $s^0_{i,j}$ referring to the setup time of machine j before processing the first job in the permutation. Each job i has a given release time r_i and is available for processing only after r_i. Once a job begins to be processed, there cannot be breaks for the job during processing or between machines (no-wait constraint). In the assembly stage, the machine can only assembly one product at a time. Assembly of a product cannot begin until all the jobs belonging to this product have been processed by all M machines. The assembly time of product h is denoted by p^A_h.
- $p_{i,j}$, p^A_h, $s_{i_1,i_2,j}$, $s^0_{i,j}$ and r_i are assumed to be stochastic. It is assumed that they respectively subject to uniform distributions:

$$p_{i,j} \sim U((1-\eta)P_{i,j},(1+\eta)P_{i,j})$$

$$p^A_h \sim U((1-\eta)P^A_h,(1+\eta)P^A_h)$$

$$s_{i_1,i_2,j} \sim U((1-\eta)S_{i_1,i_2,j},(1+\eta)S_{i_1,i_2,j})$$

$$s^0_{i,j} \sim U((1-\eta)S^0_{i,j},(1+\eta)S^0_{i,j})$$

$$r_i \sim U((1-\eta)R_i,(1+\eta)R_i)$$

- The objective is to determine a job assignment and sequence to minimize the maximum completion time, i.e., the makespan.

B. Makespan Calculation

Let $\lambda_k = \{\lambda_k(1),...,\lambda_k(N_k)\}$ denote the permutation of the jobs assigned to factory k. A solution for DAPSFP_E is a set of sub-sequences $\{\lambda_1,...,\lambda_F\}$. Denote the completion time of job i on machine j as $C_{i,j}$ and let $C_{i,0}$ be the time when job i begins processing on machine 1. Makespan in factory k of a given solution is equal to $C_{\lambda_k(N_k),M}$ and can be calculated as follows:

$$C_{\lambda_k(1),0} = \max[r_{\lambda_k(1)}, s^0_{\lambda_k(1),1}, s^0_{\lambda_k(1),j} - \sum_{s=1}^{j-1} p_{\lambda_k(1),s}], \quad (1)$$
$$j = 2,...,M; k = 1,...F$$

$$C_{\lambda_k(1),j} = C_{\lambda_k(1),j-1} + p_{\lambda_k(1),j}; j = 1,...,M, k = 1,...F \quad (2)$$

$$C_{\lambda_k(i),0} = \max[r_{\lambda_k(i)}, C_{\lambda_k(i-1),j} + s_{\lambda_k(i-1),\lambda_k(i),j} - \sum_{s=1}^{j-1} p_{\lambda_k(i),s}],$$
$$j = 2,...,M; k = 1,...F, i = 1,...,N_k \tag{3}$$

$$C_{\lambda_k(i),j} = C_{\lambda_k(i),j-1} + p_{\lambda_k(i),j}; j = 1,...,M, k = 1,...F, i = 1,...,N_k \tag{4}$$

The assembly sequence of the final products is determined by the completion times of the *N* jobs. Denote the assembly sequence as $\lambda^A = [\lambda^A(1),...,\lambda^A(FP)]$. Let C_h^p be the latest processing completion time of all jobs in J_h and let C_h^A be the completion time of assembling product *h*.

$$C^A_{\lambda^A(1)} = C^p_{\lambda^A(1)} + p^A_{\lambda^A(1)} \tag{5}$$

$$C^A_{\lambda^A(h)} = \max[C^A_{\lambda^A(h-1)}, C^p_{\lambda^A(h)}] + p^A_{\lambda^A(h)}; h = 2,...,FP \tag{6}$$

The makespan $C_{\max} = C^A_{FP}$, which is the object function to minimize.

III. Brife Intruduction to Particle Swarm Optimization, Simulated Annealing and Hypothesis Test

A. Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational method characterized by individual improvement and population cooperation [24], [25]. It is started with a swarm of particles in the search space. Positions of particles represents candidate solutions. Each particle tries to find the optimal position by accelerate to the best position it has reached and the best position the swarm has reached. Let $X_i = [X_{i,1},...,X_{i,d}]$ and $v_i = [v_{i,1},...,v_{i,d}]$ denote the position and velocity of the i-th particle where d represents the dimension of search space. The best position found by particle i is the *pbest*: $X_i^p = [X_{i,1}^p,...X_{i,d}^p]$ and the best position found by the whole swarm is the *gbest*: $X^g = [X_1^g,...X_d^g]$. The new velocity of particle i is determined by X_i^p, X^g, X_i and v_i.

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1q_1(X^p_{i,j} - X_{i,j}(t)) + c_2q_2(X^g_j - X_{i,j}(t)); j = 1,...,d \tag{7}$$

where c_1 and c_2 are constants called acceleration coefficients and w is a constant called inertia factor. q_1 and q_2 are independent random numbers that are uniformly distributed in the range[0,1]. The new positions of particles are updated according to the velocities.

$$X_{i,j}(t+1) = X_{i,j}(t) + v_{i,j}(t+1); j = 1,...,d \tag{8}$$

This process is repeated until a pre-defined stopping criterion is met.

B. Simulated Annealing

Simulated annealing (SA) is a metaheuristic to search the optimal solution in a search space, which is usually discrete [26]. The algorithm generates a new position called a neighbor according to the initial position. Let ΔE be the amount by which the objective function is changed when state changes from the initial position to the neighbor. Under temperature t, the new state is accepted with probability $\min\{1, \exp(-\Delta E / t)\}$. t generally becomes lower, simulating the temperature in physical annealing.

C. Hypothesis Test

Generally, a stochastic optimization problem can be characterized as follows:

$$\min_X J(X) = E[L(X,\xi)] \tag{9}$$

where X is a feasible solution of the problem and J is the expectation of L, a function of X and ξ that is the sample performance.

Hypothesis test (HT) is an important method that can make tests for predefined hypothesis based on experiment data[16]. To compare two different solutions X_1 and X_2 in a uncertain optimization problem, multiple independent evaluations are often needed. When n independent simulations are performed for X_i, the estimated mean value of $J(X_i)$ and the variance can be calculated as follows.

$$\bar{J}_i = \bar{J}(X_i) = \sum_{j=1}^{n_i} L(X_i,\xi)/n_i \tag{10}$$

$$s_i^2 = \sum_{j=1}^{n_i} [L(X_i,\xi) - \bar{J}_i]^2/(n_i - 1) \tag{11}$$

Suppose $J(X_i) \sim N(\mu_i, \sigma_i^2)$. According to the Law of Large Number and Central Limit Theorem, the estimated

performance of X_i, denoted as $\hat{J}(X_i)$, are normally distributed when n approaches to infinite. The limiting distribution is $N(\bar{J}(X_i), s_i^2/n)$.

For a stochastic optimization problem, the theoretical performance variances of all solutions are often supposed to be the same[16]. Thus it's assumed that $\sigma_1 = \sigma_2$. Let the null hypothesis H_0 be $\mu_1 = \mu_2$ and let the alternative hypothesis be $\mu_1 \neq \mu_2$. Then the reject region of H_0 is:

$$\left|\bar{J}_1 - \bar{J}_2\right| \geq t_{\alpha/2}(n_1 + n_2 - 2) \cdot a_1 \cdot a_2 = \tau \qquad (12)$$

where

$$a_1 = \sqrt{[(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2]/(n_1 + n_2 - 2)},$$

$$a_2 = \sqrt{(n_1 + n_2)/(n_1 n_2)}$$

Thus, when the null hypothesis holds, i.e., $\left|\bar{J}_1 - \bar{J}_2\right| < \tau$, performances of the two solutions can be regarded as the same and the performances are regarded as significantly different if the alternative hypothesis holds. Specifically, for a stochastic optimization problem with an objective to minimize the function, when $\bar{J}_1 - \bar{J}_2 \geq \tau$, X_2 is taken as better than X_1 and vice versa.

IV. PSOSAHT Algorithm

A. The PSOSAHT scheme

In this study, we adapt our previously proposed particle swarm optimization (PSO) based hybrid method, named PSOSAHT [15] for this complex scheduling problem.

PSO is a computational method which search solutions with a swarm of particles moving in a continuous search space. Solutions of flowshop scheduling problem, i.e., the job permutations, are usually denoted as integer sequences and can be mapped to position values of a continuous space by using the ranked-order-value (ROV) rule. According to the ROV rule, similar job sequences tend to be mapped to close points, i.e., topological structure of the continuous search space also reflects relationship between job sequences. Thus the mapping is reasonable. Moreover, to address the flowshop scheduling problem, which has a large solution space, PSO is ponderable for its efficiency and simplicity. Therefore in the PSOSAHT scheme, PSO is used for global exploration to evolve the job permutation.

In PSOSAHT, simulated annealing (SA), and hypothesis test (HT) were well integrated with PSO for solving optimization problems with uncertainty. The SA method, which serves as local search, can search solutions by generating "neighbors" of the initial solution and avoid premature convergence by probabilistic jumping towards the generated neighbor. The objective function of stochastic flowshop scheduling problem is also a random variable, thus appropriate statistical method is required for estimating candidate solutions. In PSOSAHT, HT is utilized for comparing performance of solutions.

In this paper, the PSOSAHT algorithm is adapted to address DAPFSP_E problem. Firstly, scheduling of DAPFSP includes both job permutation and job assignment. In this paper, the job assignment is evolved by SA-based local search. Secondly, the release time and setup-time are also stochastic and contribute to uncertainty of the objective function. To estimate solutions by HT, multiple independent evaluations should be performed with consideration of every independent random variable.

B. Solution Representation

Since a solution of DAPFSP not only includes the permutation but it also requires assignment of jobs to factories. This type of solution can be denoted by a two-layer encoding: $\pi_i = [\pi_i^1, \pi_i^2]$, in which the first layer $\pi_i^1 = [\pi_{i,1}^1, ..., \pi_{i,N}^1]$ represents the priority of jobs and the second layer $\pi_i^2 = [\pi_{i,1}^2, ..., \pi_{i,N}^2]$ denotes factories which these N jobs are assigned to respectively. In each factory, jobs are processed following the order of their priority presented in π_i^1. Table I shows an example with 8 jobs and 2 factories:

TABLE I. Solution Representation

π_i^1	3	1	8	2	7	4	6	5
π_i^2	1	2	2	1	1	2	1	1

A solution in such a two-layer encoding can be decoded into a schedule Λ as described in section II. As an example, the solution in Table I can be decoded into the following scheduling:

The 1st factory: 4,1,8,7,5

The 2nd factory: 2,6,3

To address DAPFSP_E, the PSO method is carried out to optimize the first layer of the encoding, π_i^1. Since the positions of particles in the search space of PSO are continuous values while the job permutations are integer sequences, a ROV rule based on random key representation is utilized to map between job permutations and continuous position values [37]. In the ROV rule, the smallest position value of a particle is assigned the rank value 1. Then the second smallest position value of the particle is assigned the rank value 2 and so on. In this way the position values of the particle is converted to a job permutation.

In Table II an example is presented to show how ROV rule works.

C. PSO-based Search Combining HT

In this paper, we adopt the PSO-based searching operator to evolve the first layer of encoding π_i^1, i.e., the job permutation. Equation (7) and (8) are applied on position values of particles, which can be converted into permutations by using the ROV rule. To update $pbest$ and $gbest$, HT is utilized to evaluate and compare the performance of candidate solutions represented by the particles.

D. SA-based Local Search Combining HT

In this research, SA-based local search is applied to evolve both layers of encoding π_i^1 and π_i^2. To evolve π_i^1, the searching operator is applied directly on the job permutations rather than the position values. For the DAPFSP, the MUTATE operator and the SWAP operator, changing π_i^1 and π_i^2 respectively, are utilized to generate the neighbors.

TABLE II. POSITION VALUE AND CORRESPONDING RANK VALUE

Dimension	1	2	3	4	5	6	7	8
Position Value	1.72	0.60	3.81	1.84	2.02	1.40	1.77	2.86
Rank Value	3	1	8	5	6	2	4	7

- SWAP: randomly select two different jobs from a permutation and swap them.
- MUTATE: randomly select a job from the N jobs and change its assignment.

When a neighbor of $\pi_1 = [\pi_1^1, \pi_1^2]$ is generated: $\pi_2 = [\pi_2^1, \pi_2^2]$, firstly HT will be applied to compare the performances of π_1 and π_2. If the null hypothesis i.e. μ1=μ2 holds, performances of π_1 and π_2 can be regarded as the same and another neighbor should be generated and compared with π_1. Otherwise, π_1 will be replaced by π_2 with probability $\min\{1, \exp(-(\bar{J}_2 - \bar{J}_1)/t))\}$ under temperature t, which generally becomes lower. After the local search, if a better solution is found, the global best solution should be updated.

It's necessary to make sure the permutation generated by the new position of particle is the same as the permutation updated by the SA, for which the position values of particles should be updated correspondingly after the GA-based local search. As shown in Table III, when the SWAP operator in local search is applied on a pair of jobs: job 2 and job 6, the corresponding position values 0.60 and 1.40 should be swapped accordingly.

In this paper, we apply the exponential cooling schedule, $t_k = \lambda t_{k-1}$ [16]. Also, we adopt metropolis sampling process, i.e., let number of neighbor samplings be $N(N-1)$ where N represents the number of jobs, to balance between solution quality and search efficiency.

E. PSOSAHT

The PSOSAHT method is characterized by PSO-based search, SA-based local search and HT for evaluating solutions. The procedure of PSOSAHT is proposed as follows:

1) Initialization

Randomly select P_s particles in the search space. Convert the position values of the particles into job permutations by using the ROV rule and denote the corresponding permutation of the i -th particle as $\pi_i^1 = [\pi_{i,1}^1, ..., \pi_{i,N}^1]$. Randomly generate a factory assignment π_*^2. Let $\pi_i = [\pi_i^1, \pi_i^2]$ where $\pi_i^2 = \pi_*^2 (i = 1, ..., P_s)$. Thus the P_s candidate solutions have been generated. Perform multiple independent evaluations for the P_s solutions and determine the $pbest$ and $gbest$ with HT.

2) Repeat until gbest keeps fixed at consecutive L iterations.

Firstly, use (7) (8) to update the positions of all P_s particles in the swarm. Convert the new positions to job permutations $\pi_i^1 (i = 1, ..., P_s)$ by ROV rule and combine π_i^1

TABLE III. SWAPPED POSITION VALUE AND CORRESPONDING RANK VALUE

Dimension	1	2	3	4	5	6	7	8
Position Value	1.72	0.60	3.81	1.84	2.02	1.40	1.77	2.86
Rank Value	3	1	8	5	6	2	4	7
New Position Value	1.72	1.40	3.81	1.84	2.02	0.60	1.77	2.86
New Rank Value	3	2	8	5	6	1	4	7

and π_*^2 into P_s job schedules. Evaluate these P_s schedules and determine the new $pbest$ and $gbest$. Denote the $gbest$ as $\pi_* = [\pi_*^1, \pi_*^2]$.

Then perform SA-based local search for $gbest$: perform the SWAP operator for π_*^1 and perform the MUTATE operator for π_*^2 to generate a neighbor of π_*. Evaluate the neighbors and update π_* using HT.

Finally, let $t = \lambda t$, $g = g + 1$ where $0 < \lambda < 1$.

3) Output the best solution π_*

V. COMPUTATIONAL TESTS

In this section, we perform computational test to validate the performance and efficiency of the PSOSAHT for DAPFSP_E. We generate 36 test problems based on standard DAPFSP [2], where N ={8}, M ={2,3,4,5}, F ={2,3,4}, FP ={2,3,4}. For instance, test problem with N =8, M =2, F =3, and FP =4 can be denoted by 8_2_3_4. The mean values for processing time and assembly time for each job on related machine are directly adopted from DAPFSP [2]. The expectation value of SDST is assumed to be subject to folded normal distribution with location of 20 and scale of 25. And the expectation value of release time is assumed to be subject to half-normal distribution with scale of $(N/F-1)\times 30$. The stochastic processing time, assembling time, setup time and release time is supposed to be subject to uniform distributions as defined in Section II, and the noise magnitude η is set to be 5%.

Parameters for PSOSAHT are set as follows: P_s =20, w =1.0, $c_1 = c_2$ =2.0, $x_{\min}$ =0, $x_{\max}$ =4.0, $v_{\min}$ =-4.0, $v_{\max}$ =4.0, the initial temperature t_0 =3, λ =0.9, L =5, 10 independent evaluations for HT. To demonstrate the performance of PSOSAHT, we apply three different algorithms: PSOSA-1, PSOSA-2 and PSOSAHT to solve the benchmark problems. In PSOSA-1, HT is replaced by comparing results of one evaluation. In PSOSA-2, HT is replaced by comparing mean values of results of 10 independent evaluations.

For each problem, experiments are conducted independently for 20 times with each method. Two values are used to assess the algorithms: the relative percentage error of the best expected makespan (BRE) and the relative percentage error of the average expected makespan (ARE), with respect to the minimum makespan found by all the three algorithms C^*. The expected makespan is calculated with $P_{i,j}$, P_h^A, R_i, $S_{i_1,i_2,j}$ and $S_{i,j}^0$. The BRE and ARE for the 36 benchmark problems are presented in TABLE IV.

TABLE IV. PERFORMANCES OF PSOSAHT, PSOSA-1 AND PSOSA-2

Instance Index	BRE			ARE		
	PSOSA -1	***PSOSA -2***	***PSOSA HT***	***PSOSA -1***	***PSOSA -2***	***PSOSA HT***
8_2_2_2	0.13	2.07	0	8.35	14.08	5.17
8_2_2_3	6.56	0.16	0	11.20	9.33	6.15
8_2_2_4	0	0	0	6.90	8.49	3.24
8_2_3_2	1.21	7.69	0	13.79	23.57	6.35
8_2_3_3	2.61	0	0.39	9.93	14.02	5.97
8_2_3_4	0	0	0	3.33	6.36	0.00
8_2_4_2	0	0	0	7.27	3.08	0.25
8_2_4_3	0	0	0	9.86	3.99	0.52
8_2_4_4	0	0	0	1.33	3.00	0.00
8_3_2_2	0	0	0	5.00	2.16	0.11
8_3_2_3	0	0	0	10.47	5.50	1.84
8_3_2_4	0	3.32	0	20.36	14.66	7.21
8_3_3_2	0	0	0	3.50	6.73	1.13
8_3_3_3	0	0	0	22.06	14.13	5.31
8_3_3_4	0	0	0	14.46	17.71	1.15
8_3_4_2	0.37	0	0.37	3.62	1.79	0.36
8_3_4_3	0	0	0	15.37	22.11	0.00
8_3_4_4	0	0	0	5.27	8.46	0.62
8_4_2_2	0	0	0	8.03	5.35	2.73
8_4_2_3	0.08	0	0.08	21.21	14.87	6.35
8_4_2_4	0.59	0	0	11.99	24.79	4.55
8_4_3_2	0	0	0	2.36	2.62	0.00
8_4_3_3	6.68	2.53	0	17.84	17.81	4.72
8_4_3_4	0	0	0	2.03	4.13	0.02
8_4_4_2	0	2.22	0.63	13.60	14.84	3.60
8_4_4_3	0	0	0	1.80	2.67	0.00
8_4_4_4	0	0	0	3.36	3.26	0.04
8_5_2_2	0	1.08	1.19	18.62	20.93	6.77
8_5_2_3	0	0	0	13.32	6.03	3.39
8_5_2_4	0	0	0	4.71	6.10	0.71
8_5_3_2	0	0	0	8.01	5.59	0.59
8_5_3_3	0	0	0	12.63	11.76	5.08
8_5_3_4	2.50	1.88	0	21.64	28.60	5.63
8_5_4_2	0	0	0	5.89	5.55	0.70
8_5_4_3	0	0	0	1.42	2.23	0.00
8_5_4_4	0	0	0	4.44	1.63	0.00
mean value	0.57	0.56	**0.07**	9.59	9.94	**2.51**

The results show that PSOSAHT performs better than PSOSA-1 and PSOSA-2. In computational tests for 36 problems, the best solutions achieved are mostly generated by PSOSAHT. Makespan of solutions achieved by PSOSAHT is much lower on average according to TABLE IV.

VI. CONCLUSIONS

In this study, we have studied a new type of distributed assembly permutation flowshop scheduling problem (DAPFSP) with four realistic extensions, i.e., random nature of processing times and assembly times, stochastic sequence-dependent setup times (SDST) on processing machines, stochastic job release times, as well as the no-wait constraint in the processing stage. We adapted our previously proposed optimization framework, labeled as PSOSAHT for addressing the aforementioned model. In the PSOSAHT, the PSO-based exploration and SA-based local search were well balanced, and the HT method is suitable for evaluating and comparing the makespan of stochastic nature. Computational tests are conducted on benchmark problems, demonstrating the effectiveness and efficiency of the PSOSAHT method. To the best of knowledge, it is the first attempt to study the DAPFSP with job release time and sequence-dependent setup time both of which are stochastic. Our future work will investigate the performances of PSOSAHT enhanced by Meta-Lamarckian learning scheme [27], the probabilistic memetic framework [28] as well as transfer learning [29-34] on the more complex scheduling problems, respectively.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their constructive comments on the earlier version of this paper.

REFERENCES

[1] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem,", Computers & Operations Research, vol. 37, no. 4, pp. 754-768, 2010.

[2] S. Hatami, R. Ruiz and C. Andrés-Romano, "The distributed assembly permutation flowshop scheduling problem," International Journal of Production Research, vol. 51, no. 17, pp. 5292-5308, 2013.

[3] K. Wang, Y. Huang and H. Qin, "A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown," J Oper Res Soc, vol. 67, no. 1, pp. 68-82, 2015.

[4] R. Companys and I. Ribas, "Efficient constructive procedures for the distributed blocking flow shop scheduling problem," Industrial Engineering and Systems Management (IESM), 2015 International Conference on, Seville, 2015, pp. 92-98.

[5] I. Ribas Vila and R. Companys Pascual, "Best known solutions for Taillard's instances adapted to the distributed blocking flow shop scheduling problem," 2016.

[6] A. Rifai, H. Nguyen and S. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," Applied Soft Computing, vol. 40, pp. 42-57, 2016.

[7] J. Deng, L. Wang, S. Wang and X. Zheng, "A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem", International Journal of Production Research, vol. 54, no. 12, pp. 3561-3577, 2015.

[8] J. Deng and L. Wang, "A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem," Swarm and Evolutionary Computation, 2016.

[9] J. Deng, L. Wang, C. Wu, J. Wang and X. Zheng, "A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop," Conference on Intelligent Computing, pp. 92-98, 2016.

[10] S. Hatami, R. Ruiz and C. Andrés-Romano, "Simple constructive heuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times," Control, Decision and Information Technologies (CoDIT), 2014 International Conference on, Metz, 2014, pp. 019-023.

[11] S. Hatami, R. Ruiz and C. Andrés-Romano, "Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times," International Journal of Production Economics, vol. 169, pp. 76-88, 2015.

[12] M. Ji, Y. Yang, S. Wang and B. Liu, "Scheduling of no-wait stochastic distributed assembly flowshop by hybrid PSO," IEEE Congress on Evolutionary Computation, Vancouver, Canada, 2016.

[13] V. Fernandez-Viagas and J. Framinan, "A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem," International Journal of Production Research, vol. 53, no. 4, pp. 1111-1123, 2014.

[14] G.M. Komaki, S. Mobin, E. Teymourian and S. Sheikh, "A general variable neighborhood search algorithm to minimize makespan of the distributed permutation flowshop scheduling problem," World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering 9(8): 2618-2625, 2015.

[15] B. Liu, L. Wang, and Y. Jin, "Hybrid particle swarm optimization for flowshop scheduling with stochastic processing time," Computational Intelligence and Security. Springer Berlin Heidelberg, pp. 630–637, 2005.

[16] L. Wang, L. Zhang, and D. Z. Zheng, "A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time," The International Journal of Advanced Manufacturing Technology, vol. 25(11-12), pp. 1157–1163, 2005.

[17] B. Liu, L. Wang, and Y. Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," International Journal of Advanced Manufacturing Technology, vol. 31(9-10), pp. 1001-1011, 2007.

[18] M. Pinedo, Scheduling: theory, algorithms, and systems, 2nd ed. Prentice-Hall, Englewood Cliffs, New Jersey, 2002.

[19] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," J Oper Res Soc 45(4):472–478, 1994.

[20] H. Rock, "The three-machine no-wait flowshop problem is NP-complete," J Assoc Comput Machinery 31:336–345, 1984.

[21] S.K. Goyal, and C. Sriskandarajah, "No-wait shop scheduling: computational complexity and approximate algorithms," Opsearch 25(4):220–244, 1988.

[22] N.G. Hall, and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," Oper Res 44(3):510–525, 1996.

[23] M.R. Garey and D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York, 1979.

[24] E. Bonabeau, M. Dorigo and G. Theraulaz, Swarm intelligence: from natural to artificial systems. London: Oxford University Press, 1999.

[25] J. Kennedy, R.C. Eberhart and Y. Shi, Swarm intelligence. San Francisco: Morgan Kaufmann, 2001.

[26] S. Kirkpatrick, C. D. Gelat, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, 1983.

[27] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 99-110, 2004.

[28] Q. H. Nguyen, Y. S. Ong, and M. H. Lim, "A Probabilistic Memetic Framework," IEEE Transactions on Evolutionary Computation, vol. 13, pp. 604-623, 2009.

[29] Y.S. Ong and A. Gupta, "Evolutionary Multitasking: A Computer Science View of Cognitive Multitasking," Cognitive Computation 8(2): 125-142, 2016.

[30] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Q. Zhang, "Transfer learning using computational intelligence: A survey," Knowledge-Based Systems, vol. 80, pp. 14-23, May 2015.

[31] A. Gupta, Y. S. Ong, L. Feng and K. C. Tan, "Multi-Objective Multifactorial Optimization in Evolutionary Multitasking," IEEE Transactions on Cybernetics, 2016.

[32] A. Gupta, J. Mańdziuk and Y.S. Ong, "Evolutionary multitasking in bi-level optimization," Complex and Intelligent Systems 1(1-4): 83-95, 2015.

[33] A. Gupta, Y. S. Ong and L. Feng, "Multifactorial evolution: towards evolutionary multitasking," IEEE Transactions on Evolutionary Computation, 2016.

[34] L. Feng, Y. S. Ong, A. H. Tan and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization + transfer learning for routing problems," Memetic Computing Journal, vol. 7, no. 3, pp. 159-180, 2015.

[35] Z. Zhu, Y. S. Ong and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 37, no. 1, pp. 70-76, Feb. 2007.

[36] Z. Zhu, S. Jia and Z. Ji, "Towards a memetic feature selection paradigm [application notes]," in IEEE Computational Intelligence Magazine, vol. 5, no. 2, pp. 41-53, May 2010.

[37] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," ORSA J. Comput., vol. 6, no. 2, pp. 154–160, 1994.