IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 23, 2017, accepted July 19, 2017, date of publication August 3, 2017, date of current version August 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2732738

Minimizing Makespan in Distributed Blocking Flowshops Using Hybrid Iterated Greedy Algorithms

KUO-CHING YING[1] AND SHIH-WEI LIN[2,3,4]

[1]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan
[2]Department of Information Management, Chang Gung University, Taoyuan 33302, Taiwan
[3]Department of Neurology, Linkuo Chang Gung Memorial Hospital, Taoyuan 33305, Taiwan
[4]Department of Industrial Engineering and Management, Min Chi University of Technology, Taipei 24301, Taiwan

Corresponding author: Shih-Wei Lin (swlin@mail.cgu.edu.tw)

The work K.-C. Ying was supported by the Ministry of Science and Technology of the Republic of China, Taiwan, under Grant MOST106-2221-E-027-085. The work S.-W. Lin was supported in part by the Ministry of Science and Technology of the Republic of China, Taiwan, under Grant MOST105-2410-H-182-009-MY2 and in part by the Linkou Chang Gung Memorial Hospital under Grant CMRPD3G0011.

ABSTRACT This paper studies the distributed blocking flow shop scheduling problem (DBFSP) using meta-heuristics. A mixed integer programming model for solving the problem is proposed, and then three versions of the hybrid iterated greedy algorithm (HIG_1, HIG_2, and HIG_3) are developed, combining the advantages of an iterated greedy algorithm with the operators of the variable Tabu list, the constant Tabu list, and the cooling schedule. A benchmark problem set is used to assess empirically the performance of the HIG_1, HIG_2, and HIG_3 algorithms. Computational results show that all the three versions of the proposed algorithm can efficiently and effectively minimize the maximum completion time among all factories of the DBFSP, and HIG_1 is the most effective.

INDEX TERMS Flowshop scheduling, hybrid meta-heuristic, distributed blocking flowshop.

I. INTRODUCTION

In today's globalized economy, many companies have turned from traditional single-factory production to multi-factory production to reduce the production risk and the cost of transportation. Consequently, distributed scheduling problems that concern the assignment of jobs to various factories and their subsequent sequence have been increasingly attracting the attention of researchers during the last decade. With respect to the various distributed scheduling problems, Ruiz and Naderi [1] were the first to present a distributed permutation flowshop scheduling problem (DPFSP), which was a novel generalization of the most popular permutation flowshop scheduling problem (PFSP). Ruiz and Naderi [1] also extended the well-known NEH heuristic [2] and two variable neighborhood descent search algorithms, referred to as VND_a and VND_b, to minimize the global makespan, which is the maximal completion time of the last job to be processed in any factory. These methods provided much better computational results than typical heuristic approaches, and the VND_a heuristic yielded the best solutions, on average, among the above methods.

Following the presentation of their pioneering conference paper in 2009, Naderi and Ruiz [3] subsequently presented a full journal paper that provided several MIP models and heuristics for solving the same problem. Thereafter, some effective and efficient improvement heuristics were presented to solve the DPFSP. Among these, the NEH-based heuristic algorithm [4], the genetic algorithm (GA) [5], [6], the Tabu search algorithm (TS) [7], the estimation of distribution algorithm (EDA) [8], the modified iterated greedy algorithm (MIG) [9], and the bounded-search iterated greedy algorithm (BSIG) [10] were increasingly better approximate methods for solving the DPFSP. In particular, the BSIG algorithm of Fernandez-Viagas and Framinan [10] stands out as the state-of-the-art heuristic for solving the DPFSP.

This work studies the distributed blocking flowshop scheduling problem (DBFSP) using meta-heuristics. Distributed blocking permutation flowshops are common in manufacturing, especially in the chemical, metal, pharmaceutical, electronic, plastic, food-processing, and service industries, for example [11]. In a blocking flowshop, a finite number or possibly zero intermediate buffers exist between

2169-3536 © 2017 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

each successive pair of machines. Consequently, a processing job that is completed on a machine will be blocked on that instrument until the next machine downstream becomes available. Reddi and Ramamoorthy [12] were the first to study the two-machine blocking flowshop scheduling problem (BFSP). Having reduced the problem to a special case of the travelling salesman problem, they applied the well-known Gilmore-Gomory algorithm [13] to solve it in polynomial time. Since the BFSP is NP-hard in the strong sense for a shop that has more than two machines [14], finding the optimal solution within an acceptable computation time using exact methods (such as those of Suhami and Mah [15], Ronconi and Armentano [16], and Ronconi [17] is very difficult even for problems of moderate size. Most studies of this highly complex problem have concentrated on developing constructive heuristics and improvement heuristics that can find high-quality – but not necessarily optimal – solutions in a short computation time.

The most famous constructive heuristics for tackling the BFSP include the profile fitting (PF) heuristic [18], the min-max heuristic (MM) [19], the combined MM and NEH heuristic (MME) [19], and the combined PF and NEH heuristic (PFE) [19]. Recently, Companys and Ribas [20] proposed some constructive heuristics to minimize the maximum completion time among all factories on a distributed blocking flowshop. The experimental results in the aforementioned works reveal that these constructive heuristics can rapidly find feasible solutions, and they are more useful than exact methods for solving complex BFSPs. Noteworthy improvement heuristics for solving the BFSP include the GA [21], Ronconi's algorithm (RON) [16], the fast TS algorithm [22], the improved TS algorithm (TS+M) [22], the hybrid discrete differential evolution algorithm (HDDE) [23], the iterated greedy algorithm (IG) [24], and the revised artificial immune system algorithm (RAIS) [25]. Researchers have confirmed that, among these heuristic algorithms, TS+M, RON, HDDE, IG, and RAIS are the best for solving the BFSP. A more detailed discussion of related methods and their applications to various BFSPs can be found in the comprehensive investigation of Hall and Sriskandarajah [14]. Although some studies have investigated different DPFSPs and BFSPs, to the best of the authors' knowledge, there is only one research [20] has been done on the DPFSP with the blocking constraint. Companys and Ribas [20] proposed some constructive heuristics to minimize the maximum completion time among the factories for the DPFSP with the blocking constraint. The computational results showed good performance of these constructive heuristics, which could be applied to obtain a fast solution or as the initial solution procedure in more sophisticated meta-heuristics for solving the DPFSP.

This work presents three versions of the hybrid iterated greedy (HIG) algorithm that combine the advantages of the IG algorithm with the operators of the variable Tabu list, the constant Tabu list, and the cooling schedule, to solve the DPFSP with the blocking constraint. Notably, this paper is the first study to provide effective and efficient IG-based algorithms for solving this problem. This paper is organized as follows. Section 2 defines the DBFSP and formulates it using a mixed integer linear (MIP) model. Section 3 describes in detail the three versions of the HIG algorithm. Section 4 presents results of simulations and statistical evaluations of the proposed algorithm, applied to a benchmark problem set of instances. Finally, Section 5 draws conclusions and makes recommendations for future studies.

II. DESCRIPTION AND FORMULATION OF PROBLEM

The DBFSP that is considered herein is described as follows. A set of n jobs is to be assigned to, and processed in, one of f identical factories, each with a flowshop production system that comprises the same set of m machines in series. Every factory can process all jobs, and each job can be assigned to and processed in any one of these factories. All jobs must be sequentially processed through the m machines of the assigned factory in an identical sequence without preemption. The processing time for each job may vary among machines, but does not change from factory to factory. All jobs are ready for processing at the beginning of the planning horizon, and all machines are available over the scheduling period. No intermediate buffer, which could store jobs until the next operation is performed, is assumed to exist between any pair of consecutive machines in any factory, so no upstream machine can release a completed job to the succeeding machine if the latter is busy. In such a case, the completed job must be blocked on the upstream machine until the preceding job has been completed on the succeeding machine.

The objective of scheduling is simultaneously to assign jobs to various factories and to determine their sequences to be processed in each factory to minimize the maximum completion time among the factories (global makespan). Notably, a schedule that minimizes the global makespan for a DBFSP also minimizes the sum of the job waiting times and the sum of the machine idle times. The DBFSP can be designated by a triplet $DF_m|block|C_{\max}$ using the well-known notation of Pinedo [26]. Since the $DF_m|block|C_{\max}$ problem with only one factory reduces to a general BFSP, which is NP-hard in the strong sense [14] when the number of machines exceeds two, the $DF_m|block|C_{\max}$ problem can be confidently concluded also to be NP-hard in the strong sense.

Based on the MIP model of the DPFSP that presented by Naderi and Ruiz [3], the $DF_m|block|C_{\max}$ problem can be formulated as the following MIP mathematical model. First, the following notation is defined to simplify the formulation.

Parameters:

n : Number of jobs
m : Number of machines
f : Number of factories
i : Index of machines, $i \in \{0, 1, 2, \ldots, m\}$, where $i = 0$ is a dummy machine
j : Index of jobs, $j \in \{1, 2, \ldots, n\}$

k : Index of job position in a given sequence, $k \in \{1, 2, \ldots, n\}$
l : Index of factories, $l \in \{1, 2, \ldots, f\}$
$p_{j,i}$: Processing time of job$j$on machine i

Decision variables:

$$X_{j,k,l} = \begin{cases} 1, & \text{if job } j \text{ occupies position } k \text{ in factory } l \\ 0, & \text{otherwise} \end{cases}$$

$d_{k,i,l}$ = Departure/Completion time of job in position k on machine i in factory l
$C_{\max}$ = Maximal completion time (global makespan) of the last job to be processed in any factory

The objective function of the MIP formulation is

Minimize $C_{\max}$

and the constraints are

$$\text{s.t. } d_{1,0,l} = 0,\ l = 1, \ldots, f;\ k = 1, \ldots, F, \tag{1}$$

$$d_{k,0,l} = d_{k-1,1,l}, \quad k = 2, \ldots, n;\ l = 1, \ldots, f, \tag{2}$$

$$d_{1,i,l} = d_{1,i-1,l} + \sum_{j=1}^{n} X_{j,1,l} \cdot p_{j,i},$$
$$i = 1, \ldots, m-1;\ l = 1, \ldots, f, \tag{3}$$

$$d_{k,i,l} \geq d_{k,i-1,l} + \sum_{j=1}^{n} X_{j,k,l} \cdot p_{j,i}, \quad k = 2, \ldots, n;$$
$$i = 1, \ldots, m-1;\ l = 1, \ldots, f, \tag{4}$$

$$d_{k,i,l} \geq d_{k-1,i+1,l}, \quad k = 2, \ldots, n;$$
$$i = 1, \ldots, m-1;\ l = 1, \ldots, f, \tag{5}$$

$$d_{k,m,l} = d_{k,m-1,l} + \sum_{j=1}^{n} X_{j,k,l} \cdot p_{j,m},$$
$$k = 1, \ldots, n;\ l = 1, \ldots, f, \tag{6}$$

$$\sum_{k=1}^{n} \sum_{l=1}^{f} X_{j,k,l} = 1, \quad j = 1, \ldots, n, \tag{7}$$

$$\sum_{j=1}^{n} X_{j,k,l} \leq 1, \quad k = 1, \ldots, n,\ l = 1, \ldots, f, \tag{8}$$

$$d_{k,i,l} \geq 0, \quad k = 1, \ldots, n;\ i = 1, \ldots, m;\ l = 1, \ldots, f, \tag{9}$$

$$C_{\max} \geq d_{k,m,l}, \quad k = 1, \ldots, n;\ l = 1, \ldots, f, \tag{10}$$

$$X_{j,k,l} \in \{0, 1\},\ j = 1, \ldots, n;\ k = 1, \ldots, n;\ l = 1, \ldots, f. \tag{11}$$

The goal is to minimize the maximal completion time (global makespan) of the last job to be processed in any factory. Constraint sets (1) and (2) define the starting time of the job in position k on the first machine in factory l. Constraint set (3) defines the departure time of the job in the first position on machine i in factory l. Constraint sets (4), (5) and (6) specify the relationship between departure times of each assigned job on two successive machines in a factory. Constraint set (7) ensures that each job is dispatched to exactly one factory and to exactly one job position in the assigned factory. Constraint set (8) ensures that only one job can be allocated to each job position at a factory. Constraint set (9) specifies the departure time of each job on each machine as non-negative. Constraint set (10) computes the maximum completion time among all factories. Finally, constraint set (11) defines all relevant binary variables.

Procedure *Iterated_Greedy*

Input: α
Output: ξ_{best}
1 **begin**
2 $\xi_0 \leftarrow$ Create_Initial _Solution
3 $\xi := \xi_0$; $\xi_{best} := \xi_0$
4 **while** termination conditions not meet **do**
5 $\xi_p \leftarrow$ Destruction (α, ξ)
6 $\xi_{new} \leftarrow$ Construction (ξ_p)
7 $\xi \leftarrow$ Acceptance_Criterion (ξ, ξ_{new})
8 $\xi_{best} \leftarrow$ Acceptance_Criterion (ξ_{best}, ξ_{new})
9 **return** ξ_{best}
10 **end**

FIGURE 1. An outline of the generic IG algorithm.

III. PROPOSED SELF-TUNING ITERATED GREEDY ALGORITHM

This work proposes three versions of the HIG algorithm, which combine the IG algorithm with the operators of the Tabu list and the cooling schedule. The IG algorithm (Fig. 1) is a simple but robust stochastic search algorithm that was developed by Jacobs and Brusco [27]. As can be seen in Fig. 1, after an initial solution ξ_0 is obtained, a generic IG algorithm improves the incumbent solution ξ and the best solution ξ_{best} through the iterative execution of two main phases, *destruction* and *construction*, until a specified termination condition (e.g., a maximum number of iterations, or a maximum computation time) is satisfied. In the *destruction* phase, a fixed number (α) of elements of ξ is eliminated, yielding a partial solution ξ_p. In the following *construction* phase, the eliminated elements are individually and sequentially reinserted into all possible positions of the current partial solution to yield a new solution ξ_{new}. After ξ_{new} has been yielded, some acceptance criteria are applied to determine whether it will replace ξ and ξ_{best}. Owing to their simplicity, flexibility, and high efficiency, IG-based algorithms have been successfully utilized to solve some classic scheduling problems, such as the single-machine [28], the parallel-machine [29]–[31], the permutation flowshop [32]–[34], the non-permutation flowshop [35], the multistage hybrid flowshop [36], and the distributed permutation flowshop [9] scheduling problems.

The three versions of the HIG algorithm that are proposed in this work (HIG_1, HIG_2, HIG_3) combine the Tabu list and the cooling schedule with the IG algorithm, respectively. The main difference between the HIG_1, HIG_2, and HIG_3 algorithms is they use the variable Tabu list, the constant Tabu list, and without Tabu list, respectively.

A highly effective speed-up method is used in the construction phase of all three versions of the HIG algorithm to reduce the computational burden. The following subsections further describe the coding scheme of the solutions, the detailed procedures of the three versions of the HIG algorithm and the speed-up method.

A. CODING SCHEME OF SOLUTIONS

To signify the assignment of jobs to different factories and their production sequences in each factory, a solution is coded using a numerical sequence of n integers, separated into segments by $f-1$ asterisks, where each segment corresponds to the sequences of the jobs in an assigned factory. Here, integers represent jobs and asterisks specify the partitioning of jobs in the factories. For instance, a solution that is encoded as $\{6, 5, 1, 9, 12, 7, *, 13, 10, 8, 11, 15, *, 3, 14, 2, 4, 16\}$ is a solution with 16 jobs in three factories, with production sequences in the first, second and third factories of $\{6, 5, 1, 9, 12, 7\}$, $\{13, 10, 8, 11, 15\}$, and $\{3, 14, 2, 4, 16\}$, respectively.

B. PROCEDURES OF PROPOSED HIG ALGORITHM

The main procedures of the three versions of the HIG algorithm for solving the $DF_m|block|C_{\max}$ problem are as follows.

Step 1: Generation of initial solution

Step 1.1 Apply the PW heuristic ([37], see Section 3.3) to yield a job list $J = (J_{[1]}, J_{[2]}, \ldots, J_{[n]})$.

Step 1.2 Apply the NEH_2 heuristic [3] to insert a job, in order from the first job in the job list J, into the current partial solution until all jobs have been inserted and an initial solution Π has thus been obtained. That is, each job is inserted into all possible positions in the current partial solution, and the one with the lowest partial global makespan, is subsequently utilized to replace the current partial solution before the insertion of the next job.

Step 1.3 Set Π as both the incumbent solution (Π^*) and the best solution (Π_{best}), and set the accessible Tabu list $TL := \phi$.

Step 2: Destruction phase

Step 2.1 Determine the accessible Tabu list (TL) for the removal of jobs based on the Tabu list tenure (TLT), which is obtained using the following formula;

$$TLT = TLT^{low} + u(TLT^{high} - TLT^{low})$$

where TLT^{high} and TLT^{low} are the maximal and the minimal Tabu list tenure, respectively u is set as a random number between 0 and 1 (meaning that a variable Tabu list is used) for the HIG_1 algorithm, u is set as 0 (meaning that a constant Tabu list is used) for the HIG_2 algorithm; TLT^{low} and TLT^{high} are set as 0 (meaning that no Tabu list is used) for the HIG_3 algorithm.

Step 2.2 Select one job that is not in TL from each factory that with the largest completion time. Move the α_1 selected jobs (which are assumed to be all of the selected jobs) from Π^* to Π_R and put them into TL, where Π_R is a list of the removed jobs, arranged in order of their selection. Concurrently, set as Π^*_{P1} the current partial sequence of Π^* with the α_1 removed jobs eliminated.

Step 2.3 Select one job that is not in TL from each factory that has the smallest completion time. Move the α_2 selected jobs (which are assumed to be all of the selected jobs) from Π^*_{P1} to Π_R in the order in which they were selected, and add them to TL. Concurrently, set as Π^*_{P2} the current partial sequence of Π^*_{P1} with the α_2 removed jobs eliminated.

Step 2.4 Randomly select $(\alpha - \alpha_1 - \alpha_2)$ distinct jobs that are not in TL from all of the factories, where $\alpha \in [\alpha_{\min}, \alpha_{\max}]$. Move the $(\alpha - \alpha_1 - \alpha_2)$ selected jobs from Π^*_{P2} to Π_R in the order in which they were selected and add them to TL.

Step 3: Construction phase

Sequentially reinsert the jobs in Π_R, from the first position to the last position, into Π^*_{P2}, until a new solution (Π^*_{new}) has been constructed. When reinserting a job, all possible positions in the current partial solution should be considered and the best one, and the best position, which is the one with the lowest partial global makespan, is subsequently utilized to replace the current partial solution before the insertion of the next job. To accelerate the insertion operation, the speed-up method that is described in Section 3.4 is used.

Step 4: Acceptance criterion

To improve the ability of the incumbent solution to escape from local minima, the following acceptance criterion and the cooling schedule are used in all three versions of the HIG algorithm to determine whether Π^* and Π_{best} will be updated by the newly obtained solution Π^*_{new}.

IF $C_{\max}(\Pi^*_{new}) \le C_{\max}(\Pi_{best})$ THEN
set $\Pi_{best} := \Pi^*_{new}$ and $\Pi^* := \Pi^*_{new}$;
ELSE_IF $C_{\max}(\Pi^*_{new}) \le C_{\max}(\Pi^*)$ THEN set $\Pi^* := \Pi^*_{new}$;
ELSE_IF $C_{\max}(\Pi^*_{new}) > C_{\max}(\Pi^*)$ TEHN
Generate $r \sim$ U (0,1);
IF $r < e^{([C_{\max}(\Pi^*) - C_{\max}(\Pi^*_{new})]/T)}$ set $\Pi^* := \Pi^*_{new}$
Otherwise, discard Π^*_{new}.

Here, $C_{\max}(\cdot)$ represents the global makespan of solution $(\cdot)$; $r \in [0, 1]$ is a pseudo-random number that is sampled from the standard uniform distribution U (0,1); and T denotes the current temperature with an initial temperature $T_0 = T_{Value} \cdot \sum_{i=1}^{m}\sum_{j=1}^{n} p_{j,i}$ and will be decreased from its preceding temperature, i.e., $T \leftarrow \lambda T$ ($\lambda \in [0, 1]$), after running a preset number of iterations (I_{iter}) at a particular temperature.

Step 5: Stopping criterion

To test fairly the three versions of the HIG algorithm, iterate Steps 2–4 for each algorithm until the computation time reaches a specified threshold ($T_{\max}$).

The PW heuristic was proposed by Pan and Wang [37] for the single blocking flowshop problem, and The NEH_2 heuristic was proposed by Naderi and Ruiz [3] for the distributed permutation flowshop scheduling problem. In this study, we combined the PW heuristic and the NEH_2 heuristic to generate an initial solution for the $DF_m|block|C_{\max}$ problem. To compare the solution quality on the same basis, the procedures for implementing the three versions of the HIG algorithm are the same, except for the use of different Tabu lists in Step 2.1: the HIG_1, HIG_2, and HIG_3 algorithms use a variable Tabu list, a constant Tabu list, and no Tabu list, respectively. Additionally, in Step 4, all the three versions of the HIG algorithm employ the decreasing temperature mechanism instead of the constant temperature mechanism to avoid searching processes that would be prone to stagnation. The process used is executed by generating a pseudo-random number $r \in [0, 1]$ and updating the incumbent solution Π^* by Π^*_{new} whenever $r < e^{([C_{\max}(\Pi^*) - C_{\max}(\Pi^*_{new})]/T)}$.

C. PW HEURISTIC

The PW heuristic is a simple construction heuristic that was proposed by Pan and Wang [37] for solving the blocking flowshop scheduling problem. The PW heuristic is applied as follows.

Step 1: Use the following equations to calculate the departure time, $d_{[k],i}$, of job j on machine i if it is at position k in the schedule.

$$d_{[1],0} = 0, \tag{12}$$
$$d_{[k],0} = d_{[k-1],1}, k = 2 \ldots, n, \tag{13}$$
$$d_{[k],i} = \max\{d_{[k],i-1} + p_{[k],i}, d_{[k-1],i+1}\}, \quad k = 2 \ldots, n; i = 1 \ldots, m-1, \tag{14}$$
$$d_{[k],m} = d_{[k],m-1} + p_{[k],m}, k = 1 \ldots, n. \tag{15}$$

Step 2: Use the following equations to calculate the slope index $f_{j,k}$ $(\forall j, k)$ of job j at position k.

$$f_{j,k} = (n-k-2)\delta_{j,k} + \chi_{j,k}, \tag{16}$$

Where

$$\delta_{j,0} = \sum_{i=1}^{m} \frac{m}{i + \frac{k(m-i)}{n-2}}(d_{[1],i} - p_{j,i}),$$
$$\delta_{j,k} = \sum_{i=1}^{m} \frac{m}{i + \frac{k(m-i)}{n-2}} \times (d_{[k+1],i} - d_{[k],i} - p_{j,i}), \quad \text{and}$$
$$\chi_{j,k} = \sum_{i=1}^{m} \frac{m}{i + \frac{k(m-i)}{n-2}}(d_{[k+2],i} - d_{[k+1],i} - \sum_{\substack{q \in U \\ q \neq j}} \frac{p_{q,i}}{(n-k-1)}).$$

Step 3: Set the job with the smallest $f_{j,0}$ value as the first job in current partial job list $J_P = (J_{[1]})$. In case of a tie, make the job has the smallest $\chi_{j,0}$ value as the first job. Let the unscheduled job set $U = J - \{J_{[1]}\}$.

Step 4: Repeat the following procedure to select a job for adding to the next position in the current partial job list until $U = \phi$.

Step 4.1 For each machine, use Eqs. 12-16 to calculate the departure time, $d_{[k],i}$ $(i = 1, \ldots, m)$, of the job at the last position, say position k, in the current partial job list.

Step 4.2 Use Eq. 17 to calculate the slope index $f_{j,k}$ of each job $j \in U$. Remove the job with the smallest $f_{j,k}$ value from the unscheduled job set, and add it at the next position in the current partial job list. In case of a tie, make the job has the smallest $\chi_{j,0}$ as the first job.

D. SPEED-UP METHOD

In the construction phase (Step 3) of the three versions of HIG algorithm, the jobs in Π_R are successively inserted at all possible positions in Π^*_{P2}, and the best one, which is the one with the lowest partial global makespan, is chosen. In such a construction phase, substantial time is taken to calculate the makespan of each possible (partial) solution of the insertion. To accelerate the evaluation of the best insertion position, a speed-up method that is revised the scheme of Wang et al. [23] for solving the blocking flowshop problem is proposed. Based on the assumption that, in the current partial solution, n_P jobs have been assigned to a factory, the following speed-up method is applied to evaluate $n_P + 1$ sequences that are generated by inserting a job, $J_{j'}$, at all possible positions in this current partial solution.

Step 1: Use Eqs. 11-15 to calculate the departure times $d_{[k],i}$ $(k = 1, \ldots, n_P; i = 1, \ldots, m)$ of the n_P jobs at the assigned factory in the current partial solution.

Step 2: Use the following equations to calculate the tails $f_{[j],i}$ $(j = 1, \ldots, n_P; i = 1, \ldots, m)$ of the

n_P jobs in the assigned factory in the current partial solution.

$$f_{[n_P],m+1} = 0, \tag{17}$$

$$f_{[n_P],i} = f_{[n_P],i+1} + p_{[n_P],i},\ i = m, \ldots, 1, \tag{18}$$

$$f_{[j],m+1} = f_{[j+1],m}, j = n_P - 1, \ldots, 1, \tag{19}$$

$$f_{[j],i} = \max\{f_{[j],i+1} + p_{[j],i}, f_{[j+1],i-1}\},\ j = n_P - 1, \ldots, 1; i = m, \ldots, 2, \tag{20}$$

$$f_{[j],1} = f_{[j],2} + p_{[j],1}, j = n_P - 1, \ldots, 1. \tag{21}$$

Step 3: Use Eqs. 12-16 to calculate the departure times $d_{[q],i}$ $(i = 1, \ldots, m)$ of job $J_{j'}$ when it were to be inserted at position q in the assigned factory in the current partial solution.

Step 4: Use the following equation to calculate the global makespan of the partial sequence $\Pi^*_{P_q}$ when job $J_{j'}$ were to be inserted at position q in the assigned factory in the current partial solution.

$$C_{\max}(\Pi^*_{P_q}) = \max_{i=1,\ldots,m} (d_{[q],i} + f_{[q],i}),\ q = 1, \ldots, n_P + 1. \tag{22}$$

Step 5: Choose the optimal insertion position, among the best ones across all factories, which minimizes the global makespan.

IV. COMPUTATIONAL RESULTS AND DISCUSSION

A. TEST PROBLEMS

To verify the effectiveness of the three versions of the HIG algorithm, the benchmark problem set that was presented by Naderi and Ruiz [3] for testing the DPFSP was used. The benchmark problem set was augmented using the 120 benchmark instances of Taillard [38], where the processing time $p_{j,i}(j = 1, \ldots, n; i = 1, \ldots, m)$ is an integer that is generated from the uniform distribution [1, 99]. Naderi and Ruiz [3] expanded these 120 test instances to 420 and 720 test instances in the small and large problem sets, respectively.

The instances in the small problem set featured the number of jobs $n = \{4, 6, 8, 10, 12, 14, 16\}$, the number of machines $m = \{2, 3, 4, 5\}$, and the number of factories $f = \{2, 3, 4\}$. The total number of combinations of distinct numbers of jobs, machines, and factories was 84, which was therefore the number of sub-problem sets. Five instances were generated for each sub-problem set, yielding a total of 420 (84×5) instances in the small problem set. The instances in the large problem set were those in 72 sub-problem sets, featuring the number of jobs $n = \{20, 50, 100, 200, 500\}$, the number of machines $m = \{5, 10, 20\}$, and the number of factories $f = \{2, 3, 4, 5, 6, 7\}$. Ten instances were generated for each sub-problem set, yielding a total of 720 (72×10) problem instances. The files of these test instances can be downloaded from http://soa.iti.es.

B. PARAMETER CALIBRATION

The proposed three versions of the HIG algorithm have seven parameters, which are T_{Value}, I_{iter}, λ, TLT_R^{low}, TLT_R^{high}, $\alpha_{\min}$, and $\alpha_{\max}$, where T_{Value} is used to determine the initial temperature ($T_0 = T_{Value} \cdot \sum_{i=1}^{m}\sum_{j=1}^{n} p_{j,i}$); I_{iter} denotes the number of iterations in the search at a particular temperature; λ is the coefficient that directs the cooling schedule; TLT_R^{low} and TLT_R^{high} determine the minimal and maximal Tabu list tenures ($TLT^{low} = n \cdot TLT_R^{low}$ and $TLT^{high} = n \cdot TLT_R^{high}$), respectively; $\alpha_{\min}$ and $\alpha_{\max}$ are minimal and maximal number of jobs to be eliminated in the destruction phase.

Since the parameter values affect the computational results of the three versions of HIG algorithm, two sets of test instances are used to calibrate the parameters. In the two calibration experiments, the maximum computation time ($T_{\max}$) to solve each selected instance was set to $10 \cdot n \cdot m$ (ms). The three versions of the HIG algorithm were executed in *C* language on a personal computer that had an Intel Core Quad CPU Q9400 @2.66 GHz processor and 20 GB of RAM. Each calibration experiment was conducted on 30 instances that are randomly produced using the same data generation

TABLE 1. Parameter values used in the two-parameter calibration experiments.

Parameter	First calibration
T_{Value}	0.02: **0.03***: 0.04
I_{iter}	2000: **3000**: 4000
λ	0.875: **0.900**: 0.925
$(TLT_R^{low}, TLT_R^{high})$	**(5%,10%)**: (5%,15%): (10%,15%)
$(\alpha_{\min}, \alpha_{\max})$	(2,5):(2,6):**(3,6)**:(3,7):(4,7):(4,8)
Parameter	**Second calibration**
T_{Value}	0.025: **0.03**: 0.035
I_{iter}	2500: 3000: **3500**
λ	0.885: 0.900: **0.915**
$(TLT_R^{low}, TLT_R^{high})$	(2.5%,10.0%): (5.0%,7.5%): **(5.0%,10.0%)**: (5%,12.5%): (7.5%,10.0%)
$(\alpha_{\min}, \alpha_{\max})$	(2,6): **(3,6)**: (3,7)

*: The best combination is bold face

TABLE 2. Performance comparisions on taillard's benchmark problems (Ave. RPD_{BKS}) for $t = 15$.

Problem size	HIG_1	HIG_2	HIG_3	RAIS	HDDE	IG
20|5	0.000	0.000	0.000	0.000	0.000	0.000
20|10	0.000	0.000	0.000	0.000	0.000	0.000
20|20	0.000	0.000	0.000	0.000	0.000	0.000
50|5	0.059	0.016	0.211	0.027	1.000	0.219
50|10	0.035	-0.040	0.123	0.072	0.805	0.272
50|20	-0.145	-0.075	-0.053	0.084	0.439	0.088
100|5	-0.093	0.010	-0.079	0.000	2.618	0.826
100|10	-0.215	-0.099	-0.163	0.000	1.912	0.890
100|20	-0.300	-0.171	-0.142	0.021	1.602	0.740
200|10	-0.419	-0.273	-0.315	0.000	2.833	0.331
200|20	-0.661	-0.463	-0.571	0.000	1.880	0.633
500|20	-1.161	-1.017	-1.038	0.076	1.631	0.091
Total average	-0.242	-0.176	-0.169	0.023	1.227	0.341

TABLE 3. Paried T-tests on AVE. RPDBKS for $t = 15$.

HIG_1 vs.	HIG_2	HIG_3	RAIS	HDDE	IG
Difference	-0.0657	-0.0727	-0.2648	-1.4682	-0.5824
dof	119	119	119	119	119
t-Value	-3.3981	-3.8810	-7.1686	-13.4746	-11.1155
One-tail significance	0.0005	0.0001	0.0000	0.0000	0.0000

TABLE 4. Ave. RPD_{LB} values obtained using the milp mathematical model, HIG_1, HIG_2, and HIG_3 algorithms (small problem set).

$n\|m$	MIP	CPU Time	HIG_1	HIG_2	HIG_3
4|2	0.0000	0.07	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
4|3	0.0000	0.09	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
4|4	0.0000	0.11	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
4|5	0.0000	0.13	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
6|2	0.0000	0.20	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
6|3	0.0000	0.30	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
6|4	0.0000	0.44	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
6|5	0.0000	0.97	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
8|2	0.0000	4.55	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
8|3	0.0000	2.90	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
8|4	0.0000	2.46	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
8|5	0.0000	6.17	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
10|2	0.0000	61.90	0.000/0.000/0.000	0.000/0.000/0.019	0.000/0.000/0.000
10|3	0.0000	88.97	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
10|4	0.0000	154.35	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
10|5	0.0000	83.17	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
12|2	0.4632	1192.45	0.463/0.463/0.463	0.463/0.463/0.463	0.463/0.463/0.463
12|3	0.0000	789.79	0.000/0.000/0.000	0.000/0.000/0.000	0.000/0.000/0.000
12|4	0.7487	1372.73	0.749/0.749/0.749	0.749/0.749/0.749	0.749/0.749/0.749
12|5	0.9822	1087.41	0.982/0.982/0.982	0.982/0.982/0.982	0.982/0.982/0.982
14|2	4.1460	2880.46	4.146/4.146/4.146	4.146/4.146/4.146	4.146/4.146/4.146
14|3	7.3083	2843.24	7.308/7.308/7.308	7.308/7.308/7.308	7.308/7.308/7.308
14|4	5.9173	2897.78	5.917/5.917/5.917	5.917/5.917/5.917	5.917/5.917/5.917
14|5	8.9057	2876.78	8.891/8.891/8.891	8.891/8.891/8.891	8.891/8.891/8.891
16|2	4.1152	3540.68	3.980/3.920/3.920	3.998/3.998/4.156	3.980/3.920/3.920
16|3	14.0446	3600.29	13.810/13.810/13.810	13.810/13.810/13.866	13.810/13.810/13.810
16|4	14.0071	3600.28	13.906/13.906/13.906	13.906/13.906/13.906	13.906/13.906/13.906
16|5	18.6180	3600.28	18.129/18.129/18.129	18.129/18.129/18.129	18.129/18.129/18.129
Total average	2.8306	1096.03	2.796/2.794/2.794	2.796/2.796/2.805	2.796/2.794/2.794

procedures as that of Naderi and Ruiz [3]. The test instances in the two sets featured the number of jobs $n = \{20, 50, 100\}$, the number of machines $m = \{5, 10, 20\}$, and the number of factories $f = \{2, 3, 4, 5, 6, 7\}$. For each combination p of parameter values, each test instance was solved five times, and the best known solution ($C^{p}_{\min\_i}$) among the five replications was recorded for each test instance i. Then, the best known solution for each test instance i($C^{BKS}_{\min\_i}$) was obtained among all $C^{p}_{\min\_i}$. For choosing the best value for each parameter, the combination with the smallest average relative percentage rate $RPD_p = (C^{p}_{\min} - C^{BKS}_{\min\_i})/C^{BKS}_{\min\_i} \times 100\%$ was chosen.

In the first calibration experiment, several values of the parameters T_{Value}, I_{iter}, λ, TLT^{low}_{R}, TLT^{high}_{R}, $\alpha_{\min}$ and $\alpha_{\max}$ (see Table 1) were tested for the calibration and tuning of parameter combinations. In the second calibration experiment, the results of the first experiment were refined by adding additional values, which are presented in Table 1. As shown in Table 1, the following values were obtained using the optimal parameter combinations; in the first calibration experiment, $T_{Value} = 0.03$, $I_{iter} = 3000$, $\lambda = 0.9$, $TLT^{low}_{R} = 5\%$, $TLT^{high}_{R} = 10\%$, $\alpha_{\min} = 3$ and $\alpha_{\max} = 6$, while in the second calibration experiment (after refinement), $T_{Value} = 0.03$, $I_{iter} = 3500$, $\lambda = 0.915$, $TLT^{low}_{R} = 5\%$, $TLT^{high}_{R} = 10.0\%$, $\alpha_{\min} = 3$ and $\alpha_{\max} = 6$. Three values of TLT^{low}_{R}, 5.0%, 7.5%, and 10.0%, are tested for use in HIG_2 algorithm. Because setting TLT^{low}_{R} to 7.5% yielded the best result, TLT^{low}_{R} was fixed at this value in the HIG_2 herein. Using the above parameter settings, the maximal computational time was set to $t \cdot n \cdot m$ (ms) for various problem sets, where t is a scale factor that was set to (5, 10, 15, 20, 25, 30) and (5, 10, 15) for the single-factory and multi-factory problem, respectively. Each problem was executed five runs, and the best solution from the five replications was recorded. The following subsections present and discuss the computational results.

TABLE 5. Ave. RPD_{LB} values obtained using HIG_1, HIG_2, and HIG_3 algorithms grouped according to n and m for $t = 5$, 10, and 15 (large problem set).

$n\|m$	HIG_1	HIG_2	HIG_3
20|5	0.020/0.004/0.000	0.014/0.014/0.007	0.010/0.007/0.007
20|10	0.012/0.009/0.009	0.016/0.016/0.013	0.000/0.000/0.000
20|20	0.000/0.000/0.000	0.005/0.005/0.000	0.006/0.000/0.000
50|5	0.560/0.328/0.257	0.598/0.598/0.309	0.606/0.390/0.329
50|10	0.565/0.281/0.219	0.577/0.577/0.237	0.596/0.343/0.264
50|20	0.595/0.198/0.159	0.588/0.588/0.185	0.666/0.248/0.186
100|5	0.850/0.399/0.267	0.824/0.824/0.258	0.873/0.453/0.308
100|10	0.969/0.385/0.213	0.920/0.920/0.216	1.001/0.398/0.233
100|20	1.228/0.382/0.206	1.282/1.282/0.219	1.277/0.375/0.169
200|10	1.347/0.497/0.181	1.381/1.381/0.255	1.406/0.523/0.195
200|20	1.598/0.480/0.177	1.651/1.651/0.164	1.606/0.477/0.156
500|10	1.626/0.388/0.065	1.806/1.806/0.209	1.688/0.417/0.103
Total average	0.781/0.279/0.146	0.805/0.805/0.173	0.811/0.303/0.163

C. RESULTS OBTAINED USING PROBLEM SET OF SINGLE-FACTORY

To confirm the effectiveness of the three versions of the HIG algorithm, their performance was compared with

TABLE 6. Ave. RPD_{LB} values obtained using HIG_1, HIG_2, and HIG_3 algorithms grouped according to n and m for $t = 5$, 10, and 15 (large problem set).

f	HIG_1	HIG_2	HIG_3
2	1.102/0.280/0.118	1.244/1.244/0.224	1.163/0.279/0.117
3	0.941/0.293/0.143	0.974/0.974/0.179	0.934/0.314/0.150
4	0.714/0.256/0.137	0.808/0.808/0.177	0.800/0.302/0.162
5	0.656/0.260/0.140	0.687/0.687/0.172	0.646/0.276/0.155
6	0.611/0.265/0.141	0.619/0.619/0.182	0.636/0.282/0.151
7	0.660/0.320/0.196	0.499/0.499/0.102	0.688/0.363/0.241
Total Average	0.781/0.279/0.146	0.805/0.805/0.173	0.811/0.303/0.163

that of leading algorithms using the 120 single-factory benchmark instances of Taillard [38]. These leading algorithms are HDDE [23], IG [24], and RAIS [25], all of which were developed to solve the single-factory BFSP. Notably, the HIG_3 algorithm proposed in this study is an improved version of the IG algorithm, which is a state-of-the-art algorithm, proposed by Lin et al. [9] for the distributed permutation flowshop scheduling problem. In the literature, HDDE is performed with ten replications, while IG and RAIS are performed with five replications. Therefore, the proposed HIG_1, HIG_2, and HIG_3 algorithms were conducted with five replications, and the relative percentage deviation (RPD_{BKS}) of makespan from the best solution that was calculated using the following formula was used to compare the performance of these algorithms with that of HDDE, IG, and RAIS.

$$RPD_{BKS} = (C_{\max}^{alg} - C_{\max}^{BKS})/C_{\max}^{BKS} \times 100\%$$

where $C_{\max}^{alg}$ is the makespan value in the best solution that was obtained using the algorithm of interest, and $C_{\max}^{BKS}$ is the makespan value in the best solution that was obtained using HDDE, IG, and RAIS.

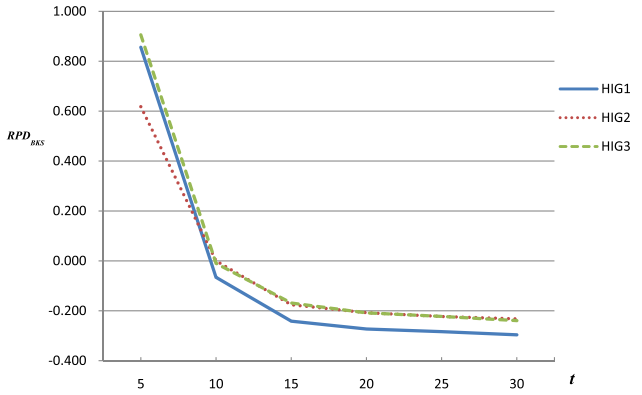

FIGURE 2. The average RPD_{BKS} values under different t vaules.

To reveal how the computational time affects quality of the solutions obtained using HIG_1, HIG_2, and HIG_3, various t values for these algorithms were tested. Figure 2 plots the total average RPD_{BKS} values for the solutions that were obtained using the HIG_1, HIG_2, and HIG_3 algorithms with various t values. Evidently, as can be seen in Fig. 2, solution quality increases with computation time. When t is equal to or larger than 15, the total average RPD_{BKS} values of the solutions that were obtained using of HIG_1, HIG_2, and HIG_3 algorithms are negative. Accordingly, HIG_1, HIG_2, and HIG_3 outperform HDDE, IG, and RAIS if t is equal to or greater than 15. Therefore, considering both of solution quality and computational efficiency, $t = 15$ is used in subsequent analysis of the 120 single-factory benchmark instances.

TABLE 7. Paired t-tests on Min. RPD_{BKS}, Mean RPD_{BKS}, and Max. RPD_{BKS}.

HIG_1 vs.	HIG_2	HIG_3
Test on Min. RPD_{BKS}		
Paired difference	-0.024/-0.027/-0.027	-0.031/-0.023/-0.017
t-value	-2.102/-2.580/-2.638	-2.898/-2.592/-1.844
Degree of freedom	719	719
P-value	0.018/0.005/0.004	0.002/0.005/0.033
Test on Mean RPD_{BKS}		
Paired difference	-0.020/-0.021/-0.024	-0.025/-0.018/-0.019
t-value	-2.627/-2.976/-2.875	-3.528/-2.833/-3.015
Degree of freedom	719	719
P-value	0.004/0.002/0.002	0.000/0.002/0.001
Test on Max. RPD_{BKS}		
Paired difference	-0.017/-0.010/-0.017	-0.022/-0.012/-0.015
t-value	-1.618/-0.879/-1.618	-2.098/-1.292/-1.620
Degree of freedom	719	719
P-value	0.053/0.190/0.053	0.018/0.098/0.053

Table 2 lists the average RPD_{BKS} (Ave. RPD_{BKS}) value for the solutions of each problem size that are obtained using HIG_1, HIG_2, HIG_3, RAIS, HDDE, and IG. Each average RPD_{BKS} value was taken over the 10 test instances for each problem size. Tables 8 and 9 present the best solutions that were obtained using these algorithms for each benchmark instance. As shown in Table 2, the total average RPD_{BKS} value of the solutions that were obtained using the HIG_1 algorithm was –0.242%. For the HIG_2, HIG_3, RAIS, HDDE, and IG algorithms, the corresponding values were –0.176%, –0.169%, 0.023%, 1.227%, and 0.341%, respectively. Evidently, the three versions of the HIG algorithm outperform the three leading algorithms in solving the traditional BFSP, while HIG_1 is the best of them. Notably, as reported by Lin and Ying [25], RAIS is better than HDDE and IG, with its maximal computational time set to $100 \cdot n \cdot m$ (ms), while the maximal computational times for the HIG_1, HIG_2, and HIG_3 algorithms are set to $15 \cdot n \cdot m$ (ms). Therefore, the three versions of the HIG algorithm take significantly less time to compute better solutions to the traditional BFSP than taken by the RAIS algorithm.

To confirm further the effectiveness of the proposed HIG_1 algorithm, paired t-tests were performed on the average RPD_{BKS} values obtained using HIG_1 and those obtained using HIG_2, HIG_3, RAIS, HDDE, and IG. The results in Table 3 that the proposed HIG_1 algorithm significantly outperforms the HIG_2, HIG_3, RAIS, HDDE, and IG algorithms at a confidence level $\alpha = 0.05$.

TABLE 8. Results for the instances with $n = 20$, 50 and 100 for $t = 15$.

$n\|m$	HIG_1	HIG_2	HIG_3	RAIS	HDDE	IG	$n\|m$	HIG_1	HIG_2	HIG_3	RAIS	HDDE	IG
20|5	**1374**	**1374**	**1374**	**1374**	**1374**	**1374**	50|5	2993	**2989**	3001	2995	3033	3002
	1408	**1408**	**1408**	**1408**	**1408**	**1408**		3199	**3182**	3200	3191	3226	3201
	1280	**1280**	1285	**1280**	**1280**	**1280**		**3001**	3011	3013	3007	3039	3011
	1448	**1448**	**1448**	**1448**	**1448**	**1448**		3126	**3118**	3128	3125	3147	3128
	1341	**1341**	**1341**	**1341**	**1341**	**1341**		3150	3154	3165	**3143**	3192	3166
	1363	**1363**	**1363**	**1363**	**1363**	**1363**		3173	3173	3179	**3169**	3183	**3169**
	1381	**1381**	**1381**	**1381**	**1381**	**1381**		3028	3024	3028	3021	3054	**3013**
	1379	**1379**	**1379**	**1379**	**1379**	**1379**		3059	**3053**	3054	3058	3081	3073
	1373	**1373**	**1373**	**1373**	**1373**	**1373**		**2900**	2906	2913	2908	2929	2908
	1283	**1283**	**1283**	**1283**	**1283**	**1283**		3113	3118	**3107**	3114	3146	3120
20|10	**1698**	**1698**	**1698**	**1698**	**1698**	**1698**	50|10	**3622**	3641	3629	3633	3667	3638
	1833	**1833**	**1833**	**1833**	**1833**	**1833**		3489	**3484**	3497	3487	3523	3507
	1659	**1659**	**1659**	**1659**	**1659**	**1659**		3481	**3475**	3485	3482	3515	3488
	1535	**1535**	**1535**	**1535**	**1535**	**1535**		3662	**3651**	3660	3666	3685	3656
	1617	**1617**	**1617**	**1617**	**1617**	**1617**		3630	**3627**	3638	3634	3650	3629
	1590	**1590**	**1590**	**1590**	**1590**	**1590**		3590	3591	3593	**3576**	3622	3621
	1622	**1622**	**1622**	**1622**	**1622**	**1622**		3694	**3674**	3686	3683	3704	3696
	1731	**1731**	**1731**	**1731**	**1731**	**1731**		**3562**	3567	3567	3574	3590	3572
	1747	**1747**	**1747**	**1747**	**1747**	**1747**		3530	**3517**	3534	3541	3556	3532
	1782	**1782**	**1782**	**1782**	**1782**	**1782**		3619	3625	3621	**3616**	3642	3624
20|20	**2436**	**2436**	**2436**	**2436**	**2436**	**2436**	50|20	4502	**4492**	4499	4504	4516	4500
	2234	**2234**	**2234**	**2234**	**2234**	**2234**		4275	4283	**4271**	4291	4296	4276
	2479	**2479**	**2479**	**2479**	**2479**	**2479**		**4261**	4271	4269	4279	4290	4289
	2348	**2348**	**2348**	**2348**	**2348**	**2348**		**4349**	4354	4353	4368	4393	4377
	2435	**2435**	**2435**	**2435**	**2435**	**2435**		**4262**	4277	4263	4275	4284	4268
	2383	**2383**	**2383**	**2383**	**2383**	**2383**		4289	**4276**	4297	4283	4308	4280
	2390	**2390**	**2390**	**2390**	**2390**	**2390**		**4302**	4313	4310	4315	4325	4308
	2328	**2328**	**2328**	**2328**	**2328**	**2328**		**4313**	4314	4318	4319	4337	4326
	2363	**2363**	**2363**	**2363**	**2363**	**2363**		4313	**4307**	4312	4313	4332	4316
	2333	**2333**	**2333**	**2333**	**2333**	**2333**		**4401**	4410	4415	4419	4439	4428

*Bold font means the best solution among various algorithms.

D. RESULTS OBTAINED USING SMALL PROBLEM SET OF MULTI-FACTORY

For the small problem set, the relative percentage deviation (RPD_{BKS}) of makespan from the lower bound that is calculated using the following formula, was used to compare HIG_1, HIG_2 and HIG_3 algorithms with the proposed MIP mathematical model, in terms of solution quality.

$$RPD_{LB} = (C_{\max}^{alg} - C_{\max}^{LB})/C_{\max}^{LB} \times 100\%$$

where $C_{\max}^{alg}$ is the makespan value of the best solution that is get using a given version of the HIG algorithm or by solving the proposed MIP mathematical model, and $C_{\max}^{LB}$ is the lower bound on the makespan value that is get by solving the proposed MIP mathematical model.

The MIP mathematical model was solved using a famous mathematical programming solver, Gurobi (Version 7.0), on a personal computer with an Intel Core Quad CPU Q9400 @ 2.66 GHz processor and 20 GB of RAM. The maximal computational time for each test instance was set to an elapsed CPU time of 3600 seconds. The final incumbent solution that was obtained by the Gurobi MIP solver was recorded as the feasible solution. The difference between the feasible solution and the lower bound is known as the gap; a gap of zero reveals that the solution is optimal.

Table 4 lists the statistical results concerning the average RPD_{LB} (Ave. RPD_{LB}) values that were obtained using the small problem set using the MIP mathematical model, and the HIG_1, HIG_2, and HIG_3 algorithms. The three t values are separated by a slash ($t = 5/10/15$), except in the result obtained using the MIP mathematical model, which includes only one Ave. RPD_{LB} value. It should be noted that, because the number of jobs is smaller in these problems, the maximal number of jobs to be removed is set to $n/2$. As revealed in Table 4, the total average RPD_{LB} values of the solutions that were obtained using HIG_1, HIG_2, and HIG_3 algorithms are smaller than those obtained using the MIP mathematical model. The MIP mathematical model found optimal solutions for all benchmark instances when the number of jobs did not exceed 10. When the number of jobs in benchmark instances was 12, 14, or 16, the MIP mathematical model obtained optimal solutions in 51, 18, and one out of 60 test instances, respectively. In total, the MIP mathematical model obtained optimal solutions in 310 out of 420 benchmark instances in the small problem set. Notably, all except one of the optimal solutions that were obtained using the MIP mathematical model were also obtained using HIG_1, HIG_2, and HIG_3 algorithms. Furthermore, the computational times required for HIG_1, HIG_2, and HIG_3 algorithms are much less than that of the MIP mathematical model. These analytical results confirm that the proposed HIG_1, HIG_2, and HIG_3 algorithms exhibit excellent convergence to optimal solutions.

TABLE 9. Results for the instances with $n = 100$, 200 and 500 for $t = 15$.

$n\|m$	HIG_1	HIG_2	HIG_3	RAIS	HDDE	IG	$n\|m$	HIG_1	HIG_2	HIG_3	RAIS	HDDE	IG
100|5	**6098**	6111	6100	6102	6291	6151	200|10	**13325**	13341	13340	13363	13756	13406
	5969	5976	**5968**	5979	6136	6022		13238	13285	**13234**	13285	13621	13313
	5858	5885	5891	5883	6063	5927		13370	13399	**13335**	13391	13741	13416
	5714	5711	**5691**	5702	5839	5772		13267	13286	**13256**	13330	13718	13344
	5918	5934	5922	5939	6065	5960		13265	**13243**	13265	13325	13721	13360
	5791	5805	5819	5806	5971	5852		13043	**13037**	13054	13064	13474	13192
	5945	5950	5955	**5936**	6095	6004		**13501**	13542	13525	13577	13925	13598
	5843	5860	**5825**	5835	5985	5915		**13392**	13423	13433	13493	13863	13504
	6098	6083	**6069**	6090	6234	6123		13207	**13204**	13238	13270	13659	13310
	6107	**6086**	6109	6125	6273	6159		**13274**	13318	13341	13345	13744	13439
100|10	**6969**	6982	6993	7004	7131	7042	200|20	**14658**	14714	14733	14766	15057	14912
	6708	**6690**	6720	6704	6816	6791		**14884**	14923	14886	14954	15284	15002
	6827	6856	6841	6847	6956	6936		**14970**	15041	14984	15109	15360	15186
	7099	7116	7116	7112	7261	7187		14957	14939	**14924**	14995	15276	15082
	6789	6767	**6751**	6771	6913	6810		**14777**	14814	14801	14889	15183	14970
	6582	6598	**6581**	6614	6739	6666		**14894**	14934	14958	14988	15223	15101
	6734	**6729**	6756	6733	6874	6801		14928	14914	**14912**	15038	15296	15099
	6787	6802	6804	6815	6940	6874		14928	14937	**14898**	15025	15310	15141
	7003	7018	**6986**	7022	7133	7055		14860	14864	**14836**	14942	15243	15034
	6872	6893	**6858**	6896	7065	6965		**14857**	14928	14914	14996	15284	15122
100|20	**7752**	7769	7756	7765	7891	7844	500|20	**36100**	36197	36142	36593	37172	36609
	7801	7796	**7790**	7834	7931	7894		**36441**	36513	36463	36915	37485	36927
	7759	7788	7800	7810	7935	7794		**36103**	36221	36209	36577	37209	36646
	7760	7796	7793	7836	7930	7899		36465	**36358**	36463	36880	37291	36641
	7789	7792	7793	7798	7944	7901		**36219**	36281	36264	36593	37232	36583
	7837	**7835**	7858	7871	7971	7888		**36451**	36521	36516	36946	37513	36917
	7914	7907	7931	**7900**	8051	7930		**36072**	36143	36110	36506	37121	36518
	7965	7948	**7937**	7950	8102	8022		**36293**	36368	36368	36804	37202	36837
	7842	7869	7848	7883	8007	7969		36141	36109	**36103**	36495	37116	36641
	7899	7919	7936	7922	8050	7993		**36307**	36410	36404	36820	37492	36866

*Bold font means the best solution among various algorithms.

E. RESULTS OBTAINED USING LARGE PROBLEM SET OF MULTI-FACTORY

Because of the complexity of the $DF_m|block|C_{\max}$ problem, a high-quality feasible solution to a large problem cannot be obtained using the proposed MIP mathematical model in a reasonable computation time. Therefore, for the test instances in the large problem set, the relative percentage deviation (RPD_{BKS}) of makespan from the best solution obtained using the HIG_1, HIG_2, and HIG_3 algorithms was used to compare the solutions obtained using the HIG_1, HIG_2, and HIG_3 algorithms in terms of quality.

Tables 5 and 6 present the statistical results concerning the average RPD_{BKS} values of 60 and 120 solutions, respectively, in the large problem set obtained using the three versions of the HIG algorithm; the results that were obtained using the three t values are separated by a slash ($t = 5/10/15$). The statistical results reveal that more jobs yield a larger average RPD_{BKS} (Table 5) and more factories are associated with a smaller average RPD_{BKS} (Table 6). Thus, the number of jobs and the number of factories in the benchmark instances affected the performance of the HIG_1, HIG_2, and HIG_3 algorithms, whereas the number of machines did not.

The best, mean, and worst makespan values of the solutions to each test problem, based on five trials, obtained using each version of the HIG algorithm, were used to compute RPD_{BKS} values, which were denoted as Min. RPD_{BKS}, Mean RPD_{BKS}, and Max. RPD_{BKS}. To determine whether the HIG_1 algorithm was better than the HIG_2 and HIG_3 algorithms, one-sided paired t-tests in terms of Min. RPD_{BKS}, Mean RPD_{BKS}, and Max. RPD_{BKS} were performed for the various t values ($t = 5/10/15$). The statistical results listed in Table 7 revealed that, at a confidence level of $\alpha = 0.05$, the proposed HIG_1 algorithm significantly outperformed the HIG_2 and HIG_3 algorithms in terms of Min. RPD_{BKS}, MeanRPD_{BKS}, and Max. RPD_{BKS}, for most of the t values. These statistical results confirm that adopting the variable Tabu list operator of TS and the cooling schedule operator of SA significantly improves the performance of IG in solving the $DF_m|block|C_{\max}$ problem.

V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

The $DF_m|block|C_{\max}$ problem is a variant of the DPFSP. Owing to the broad applications of distributed blocking flowshop systems, this work developed three versions of the HIG algorithm (HIG_1, HIG_2, and HIG_3) for solving the $DF_m|block|C_{\max}$ problem to bridge the gap between theoretical progress in DBFSP and the industrial implication of distributed blocking flowshop systems. A comprehensive benchmark problem set is used to test the effectiveness and efficiency of the HIG_1, HIG_2, and HIG_3 algorithms.

In summary, the computational results that are presented in this work are very encouraging for the application of the HIG1 algorithm to the $DF_m|block|C_{\max}$ problem. The proposed HIG_1 algorithm exploits the PW and NEH2 heuristics to develop an initial schedule and combines the IG algorithm with the operators of the variable Tabu list and the cooling schedule. A highly effective speed-up method for evaluating of the best insertion position is used to reduce the computational burden. In view of the current lack of meta-heuristics to solve the $DF_m|block|C_{\max}$ problem, this work provides an important basis to exploring this significant topic.

Many issues are worthy of further study in the area of this pioneering study. First, additional exact methods and meta-heuristic algorithms should be developed to solve effectively and efficiently the $DF_m|block|C_{\max}$ problem. Second, the proposed HIG algorithms could be modified to solve the distributed no-idle flowshop scheduling problem. Third, the distributed blocking flowshop scheduling problem that involves other sophisticated objectives is worthy of research. Fourth, the extension of the present consideration of the $DF_m|block|C_{\max}$ problem to consider the multi-objective distributed blocking flowshop scheduling problem would increase the application of scheduling theory in industry. Finally, the novel theoretical research should be expanded from the deterministic $DF_m|block|C_{\max}$ problem to stochastic problems.

APPENDIX

See Tables 8 and 9.

REFERENCES

[1] R. Ruiz and B. Naderi, "Variable neighborhood descent methods for the distributed permutation flowshop problem," in *Proc. 4th Multidiscipl. Int. Scheduling Conf., Theory Appl. (MISTA)*, Dublin, Ireland, Aug. 2009, pp. 834–836.

[2] M. Nawaz, E. E. Enscore, Jr., and I. Ham, "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

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

[4] J. Gao and R. Chen, "An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems," *Sci. Res. Essays*, vol. 6, no. 14, pp. 3094–3100, 2011.

[5] J. Gao and R. Chen, "A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Comput. Intell. Syst.*, vol. 4, pp. 497–508, Mar. 2011.

[6] C.-W. Chiou, W.-M. Chen, and M.-C. Wu, "A combined dispatching criteria approach to scheduling dual flow-shops," *Int. J. Prod. Res.*, vol. 51, no. 3, pp. 927–939, 2013.

[7] 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, 2013.

[8] 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, 2013.

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

[10] V. Fernandez-Viagas and J. M. Framinan, "A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1111–1123, 2015.

[11] H. Gong, L. Tang, and C. W. Duin, "A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times," *Comput. Oper. Res.*, vol. 37, no. 5, pp. 960–969, 2010.

[12] S. S. Reddi and C. V. Ramamoorthy, "On the flow-shop sequencing problem with no wait in process," *J. Oper. Res. Quart.*, vol. 23, no. 3, pp. 323–331, 1972.

[13] P. C. Gilmore and R. E. Gomory, "Sequencing a one state-variable machine: A solvable case of the traveling salesman problem," *Oper. Res.*, vol. 12, pp. 655–679, Oct. 1964.

[14] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Oper. Res.*, vol. 44, pp. 510–525, Jun. 1996.

[15] I. Suhami and R. S. H. Mah, "An implicit enumeration scheme for the flowshop problem with no intermediate storage," *Comput. Chem. Eng.*, vol. 5, no. 2, pp. 83–91, 1981.

[16] D. P. Ronconi and V. A. Armentano, "Lower bounding schemes for flowshops with blocking in-process," *J. Oper. Res. Soc.*, vol. 52, no. 11, pp. 1289–1297, 2001.

[17] D. P. Ronconi, "A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking," *Ann. Oper. Res.*, vol. 138, no. 1, pp. 53–65, 2005.

[18] S. T. McCormick, M. Pinedo, S. Shenker, and B. Wolf, "Sequencing in an assembly line with blocking to minimize cycle time," *Oper. Res.*, vol. 37, no. 6, pp. 925–935, 1989.

[19] D. P. Ronconi, "A note on constructive heuristics for the flowshop problem with blocking," *Int. J. Prod. Econ.*, vol. 87, no. 1, pp. 39–48, 2004.

[20] R. Companys and I. Ribas, "Efficient constructive procedures for the distributed blocking flow shop scheduling problem," in *Proc. 6th IESM*, Seville, Spain, Oct. 2015, pp. 92–98.

[21] V. Caraffa, S. Ianes, T. P. Bagchi, and C. Sriskandarajah, "Minimizing makespan in a blocking flowshop using genetic algorithms," *Int. J. Prod. Econ.*, vol. 70, no. 2, pp. 101–115, 2001.

[22] J. Grabowski and J. Pempera, "The permutation flow shop problem with blocking. A tabu search approach," *Comput. Oper. Res.*, vol. 35, no. 3, pp. 302–311, 2007.

[23] L. Wang, Q.-K. Pan, P. N. Suganthan, W.-H. Wang, and Y.-M. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 509–520, 2010.

[24] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking," *Omega*, vol. 39, no. 3, pp. 293–301, 2011.

[25] S.-W. Lin and K.-C. Ying, "Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm," *Omega*, vol. 41, no. 2, pp. 383–389, 2013.

[26] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer, 2012.

[27] L. W. Jacobs and M. J. Brusco, "A local-search heuristic for large set-covering problems," *Naval Res. Logistics*, vol. 42, no. 7, pp. 1129–1140, 1995.

[28] K.-C. Ying, S.-W. Lin, and C.-Y. Huang, "Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 7087–7092, 2009.

[29] K.-C. Ying and H.-M. Cheng, "Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 2848–2852, 2010.

[30] S.-W. Lin, Z.-J. Lee, K.-C. Ying, and C.-C. Lu, "Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates," *Comput. Oper. Res.*, vol. 38, no. 5, pp. 809–815, 2011.

[31] K.-C. Ying, "Scheduling identical wafer sorting parallel machines with sequence-dependent setup times using an iterated greedy heuristic," *Int. J. Prod. Res.*, vol. 50, no. 10, pp. 2710–2719, 2012.

[32] G. Minella, R. Ruiz, and M. Ciavotta, "Restarted iterated Pareto greedy algorithm for multi-objective flowshop scheduling problems," *Comput. Oper. Res.*, vol. 38, no. 11, pp. 1521–1533, 2011.

[33] M. Ciavotta, G. Minella, and R. Ruiz, "Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study," *Eur. J. Oper. Res.*, vol. 227, no. 2, pp. 301–313, 2013.

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

[35] K.-C. Ying, "Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic," *Int. J. Adv. Manuf. Technol.*, vol. 38, pp. 348–354, Aug. 2008.
[36] K.-C. Ying, "An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks," *J. Oper. Res. Soc.*, vol. 60, no. 6, pp. 810–817, 2009.
[37] Q.-K. Pan and L. Wang, "Effective heuristics for the blocking flowshop scheduling problem with makespan minimization," *Omega*, vol. 40, no. 2, pp. 218–229, 2012.
[38] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, 1993.

KUO-CHING YING joined the National Taipei University of Technology in 2009, and is currently a Distinguished Professor with the Department of Industrial Engineering and Management. His research is focused on the operations scheduling. He has authored or co-authored over 100 academic papers, some of which have been published in international journals, such as *Applied Intelligence*, *Applied Soft Computing*, *Computers and Operations Research*, *Computers and Industrial Engineering*, *European Journal of Industrial Engineering*, *European Journal of Operational Research*, *International Journal of Production Economics*, *International Journal of Advanced Manufacturing Technology*, *International Journal of Innovational Computing, Information and Control*, *International Journal of Production Research*, *Journal of the Operational Research Society*, *OMEGA-The International Journal of Management Sciences*, *Production Planning & Control*, *Transportation Research Part E: Logistics and Transport Review*, and so on.

SHIH-WEI LIN received the bachelor's, master's, and Ph.D. degrees in industrial management from the National Taiwan University of Science and Technology, Taiwan, in 1996, 1998, and 2000, respectively. He is currently a Professor with the Department of Information Management, Chang Gung University, Taoyuan, Taiwan. He is also with the Department of Neurology, Linkuo Chang Gung Memorial Hospital, Taoyuan, Taiwan and with the Department of Industrial Engineering and Management, Min Chi University of Technology, Taipei, Taiwan. His current research interests include meta-heuristics and data mining. His papers have appeared in *Computers & Operations Research*, *European Journal of Operational Research*, *Journal of the Operational Research Society*, *European Journal of Industrial Engineering*, *International Journal of Production Research*, *International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, *Applied Soft Computing*, *Applied Intelligence*, *Expert Systems with Applications*, and so on.