IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 13, 2020, accepted July 26, 2020, date of publication August 4, 2020, date of current version August 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014106

Minimizing Total Completion Time in Mixed-Blocking Permutation Flowshops

CHEN-YANG CHENG[1], SHIH-WEI LIN[2,3,4,*], POURYA POURHEJAZY[1], KUO-CHING YING[1], AND JIA-WEN ZHENG[1,5]

[1]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan
[2]Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[3]Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan 333, Taiwan
[4]Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei 243, Taiwan
[5]Siliconware Precision Industries Company Ltd. (Zhong Ke Facility), Taichung 428, Taiwan

Corresponding author: Kuo-Ching Ying (kcying@ntut.edu.tw)

*Shih-Wei Lin is co-first author.

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST108-2221-E-027-025/MOST109-2410-H-182-009-MY3, and in part by the Linkou Chang Gung Memorial Hospital under Grant CMRPD3G0011.

ABSTRACT Scheduling decisions are of certain cachet in production and operations management. Many extensions are developed to help address different scheduling industrial situations, one of which, the Mixed-Blocking Permutation Flowshop Scheduling Problem (MBPFSP), has gained recent recognition due to its wide industrial applications both in manufacturing and service sectors. Given the complexities of MBPFSPs, effective heuristics are needed to help generate dependable solutions for industrial-scale applications. This article proposes an Extended Simulated Annealing (ESA) algorithm to minimize total completion time (TCT) in MBPFSPs, which is the first study in the literature of MBPFSPs considering a measure that helps improve the system's average response time. Through extensive computational experiments, it is shown that the ESA outperforms the best existing solution algorithms proposed for solving the MBPFSPs. Given the freshness of the topic, this research facilitates MBPFSP's wide range of industrial applications to narrow the gap between scheduling theory and practical applications.

INDEX TERMS Scheduling, permutation flowshop, mixed-blocking, metaheuristics.

I. INTRODUCTION

Introduced by [1], flowshop scheduling is one of the well-established optimization problems in production and operations management. The problem helps determine the best ordering of a set of required jobs, to be processed on available machines, so that the desired performance measure can be obtained. In many industrial situations, there is the same pre-ordering of jobs in all machines, making the permutation flowshop scheduling problem (PFSP) one of the most frequently applied scheduling problems. There are different varieties to the well-known PFSPs, the majority of which sought to enhance the original model through addressing practical situations with different industrial-scale applications. Assuming infinite storage capacity before each machine is a prime example of the situation where the traditional PFSP does not fit in the industrial application. Limited or zero buffer capacity causes blockage of preceding machines before the availability of a machine to receive the work-in-progress (WIP) item [2]. The blocking criterion may vary from one machine/procedure to another. Mixed blocking situations are prevalent in large-scale production industries [3], however, they have been predominantly assumed identical in the scheduling literature. This short-coming was recently addressed in Mixed Blocking Permutation Flowshop Scheduling Problem (MBPFSP) through which heterogeneous blocking situations are considered [4]. This extension of the PFSP simultaneously considers different blocking constraints, relaxing the assumption of identical waiting criteria between successive machines in a zero-buffer (no-wait) production system.

Given the importance of blocking constraints in just-in-time production, and more particularly in Kanban systems [5], [6], MBPFSPs help narrow the gap between scheduling theory and its industrial applications. The complexities involved in MBPFSPs, nevertheless, exacerbate *NP*-hardness

The associate editor coordinating the review of this manuscript and approving it for publication was Ligang He.

of this category of scheduling problems, making it hard to put them in industrial-scale practice. A handful of heuristics are available to solve the simple-blocking PFSPs [7]–[12]. However, studies in MBPFSPs, and more particularly solution approaches, are quite limited. Of the existing studies, [4] were the first to propose a mathematical model, and a heuristic solution algorithm, to optimize the MBPFSP. Reference [13] applied an improved bee colony optimization approach that outperformed the solution method proposed by [4] concerning solution quality. The enhanced scatter search algorithm presented by [3] was the next effort to propose a solution approach to MBPFSPs that outperformed the two earlier algorithms. Finally, reference [14] developed the most recent MBPFSP's solution approach that overtook all the existing methods; in their research, a constraint-guided local search (CGLS) was integrated with an exhaustive neighborhood generation strategy and an extension of the well-known Nawaz, Encore, and Ham (NEH; [15]) heuristic as the initialization method. To the best of our knowledge, CGLS is the best-performing algorithm in MBPFSP literature when considering makespan as the objective function. Our review of the limited MBPFSPs studies suggests that there is room for further improving solutions quality to facilitate industrial-scale applications of this emerging scheduling problem, especially concerning the performance measure of total completion time (TCT).

In this paper, we propose an extension to the renowned simulated annealing (SA), as well as two Iterated Greedy (IG)-based algorithms, to minimize TCT in MBPFSPs. To the best of our knowledge, this paper is the first study in MBPFSP literature extending to minimize TCT, which is essential when considering the system's average response time as the target for improvement. Contributing to the limited literature of MBPFSPs, extensive numerical test instances are performed to evaluate the effectiveness and robustness of the proposed approaches. It is shown in our study that the developed algorithms outperform the existing best-performing approach, CGLS, yielding better solutions in all of the benchmark instances.

The remainder of our paper starts with a detailed description of the MBPFSP, accompanied by preliminaries and mathematical formulation. The proposed solution approaches are then elaborated in section 3. Section 4 provides exhaustive computational analysis, including parameter calibration and numerical results, and statistical investigation between the algorithms' performance. Finally, the study is concluded by offering directions for deeper research into modeling and managing PFSPs considering mixed-blocking constraints.

II. PROBLEM DESCRIPTION AND FORMULATION

This section presents the MBPFSP considered in this study. We first describe the mixed blocking constraint that is the main feature distinguishing the problem at hand from the well-known PFSPs. Preliminaries and mathematical formulation will follow to set the foundations for our study.

A. MIXED-BLOCKING CONSTRAINT

The majority of the PFSPs assume unlimited buffer capacity before each machine. In real-world scheduling situations, however, the WIP storage space is limited, or there is no buffer space. Either way, the current machine cannot release the processed job until the next machine is ready to receive it. This availability criterion can be different from one machine to another. The extant blocking PFSP literature assumed an identical availability situation while this is not always the case. The following blocking conditions, inspired by real-world industrial practices, are identified to extend blocking PFSP.

The most prevalent blocking situation, Release when Starting blocking (RSb), occurs when the job $k+1$ in the machine j can be processed only when the machine $j+1$ starts processing the job k. A second blocking situation arises when a new job, $k+1$, can be started in the machine j only after the machine $j+1$ releases job k for processing in the machine $j+2$. This blocking situation is named Release when Complete blocking (RCb). Alternatively, there can be RCb* situations when the machine j can start the job $k+1$ immediately after the machine $j+1$ releases job k, regardless of the job k's state. MBPFSP accounts for all the mentioned blocking norms, along with the situation where there is no blocking constraint (Wb). An illustrative example of the MBPFSP is presented in Figure 1.

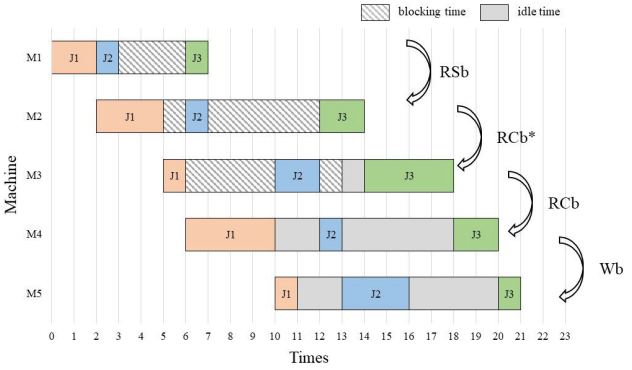

FIGURE 1. An illustrative example of the mixed-blocking situation.

The following assumptions are sought to solve MBPFSP. First of all, it is assumed that jobs are independent, and available when they are scheduled; the jobs will be processed without interruptions. Besides, the processing times of jobs are known before having them scheduled. On the other hand, we assumed that machines are always available; that there is no shutdown or break down is allowed, and each machine can only process one job at a given time.

B. PRELIMINARIES

Let assume n jobs should be processed in m machines with deterministic processing time, $p_{j,k}$; $\forall j \in \{1, 2, ..., m\}$, $k \in \{1, 2, ..., n\}$. Assuming that the machines are labeled in a fixed order, there is no buffer capacity between two

consecutive machines, j and $j+1$, hence, no room for keeping WIP items after each process until the next process can begin. A machine can process items one at a time and the next process can be started if and only if the current job is already completed; the same condition is true for the jobs, where each job can be processed in one machine at a given time.

Makespan and TCT are the most widely used measures in scheduling theory. Makespan – the maximum completion time among all jobs – is an indicator of the systems' throughput; minimizing makespan values results in the more efficient assignment of jobs to the resources, i.e. machines. Although maximizing throughput is an ultimate goal for many production systems, it increases the average response time of the system [16], making the makespan measure unfit for certain industrial applications. In this situation, TCT is of higher relevance to measuring systems performance. TCT – summation of the completion times of all jobs – is prevalent when stable utilization of resources, rapid turn-around of jobs, or minimization of WIP inventory costs is the operational improvement target [17]. Given the absence of MBPFSPs with TCT and the above justification, our study considers this performance measure to investigate this variety of scheduling problems in application areas where improving average response time is the major focus. Therefore, the problem is to find a permutation of n jobs, π, to be processed in m machines so that TCT of all machines is minimized.

The problem is denoted as $F_m\,|mixed, blk|\sum C_j$, the well-known three-filed $\alpha|\,\beta|\,\gamma$ notation suggested by [18], throughout the paper. The following notations including indices, parameters, and decision variables are considered before proceeding to the mathematical formulation;

Indices

j	Machine, $j \in \{1, 2, ..., m\}$
$job_{[k]}$	The job situated in the position $[k]$ of the sequence π

Parameters

m	Number of available machines
n	Number of jobs to be processed
$p_{j,[k]}$	The processing time of $job_{[k]}$ on the machine j
B_j	The blocking type of the machine j, where $B_j = \{RSb, RCb, RCb*, Wb\}$

Decisionvariables

$S_{j,[k]}$	Starting time of $job_{[k]}$ on machine j
$C_{j,[k]}$	Completion time of the $job_{[k]}$ on the machine j

C. MATHEMATICAL FORMULATION

Given the defined indices, parameters, and decision variables, the $F_m\,|mixed, blk|\sum C_j$ problem is formulated as a mixed-integer programming (MIP) model presented in the following. For the sake of brevity $job_{[k]}$ is replaced by $[k]$.

The MIP formulation of the problem is now presented.

$$Min\ Z = \sum_{j\in\{1,2,...,m\}} C_{j,[k]} \tag{1}$$

Subject to:

$$S_{i,[k]}=0,\ C_{i,[k]}=0,\ \ i<1\vee i>m\vee k<1\vee k>n \tag{2}$$

$$C_{j,[k]}=S_{j,[k]}+P_{j,[k]};\quad \forall j\in\{1,2,...,m\},$$
$$[k]\in\{1,2,...,n\} \tag{3}$$

$$S_{j,[1]} = C_{j-1,[1]};\quad \forall j\in\{2,...,m\} \tag{4}$$

$$S_{j,[k]}=\begin{cases} max\{C_{j-1,[k]}, S_{j+1,[k-1]}\}, & B_j=RSb \\ max\{C_{j-1,[k]}, S_{j+2,[k-1]}\}, & B_j=RCb \\ max\{C_{j-1,[k]}, C_{j+1,[k-1]}\}, & B_j=RCb^* \\ max\{C_{j-1,[k]}, C_{j,[k-1]}\}, & B_j=Wb \end{cases}$$
$$*\,\forall j\in\{1,2,...,m\},\quad k\in\{1,2,...,n\} \tag{5}$$

$$S_{j,[k]}, C_{j,[k]}\in\mathbb{Z}^+,$$
$$\forall j\in\{1,2,...,m\},\forall[k]\in\{1,2,...,n\} \tag{6}$$

The objective is to minimize TCT, shown in Equation (1); it comprises the duration for completing all jobs until the job in the last machine is done. Equation (2) sets one index out of range by assigning zero value to the respective decision variable. Equation (3) calculates the completion time of the $job_{[k]}$ on the machine j which is the summation of respective starting and processing times. According to Equation (4), the first job's start time on each machine is equal to the completion time on the previous machine; it obviously excludes the first machine. Start times are calculated using equation (5), considering different blocking situations. Finally, the last constraint allows the time decision variables to accept positive integer numbers.

III. SOLUTION METHODS

The literature of MBPFSP is relatively new and understudied. Of the existing literature, few have been devoted to developing effective solution algorithms, making it hard to put MBPFSPs in industrial-scale practice. As the main part of its contribution, this study put forward a solution approach, the ESA algorithm, to bridge the mentioned gap and help narrow the gap between scheduling theory and practical applications. Besides, two Iterated Greedy-based algorithms, one with constant destruction number (IGCD), and the other one with variable destruction number (IGVD) are included in numerical analysis to enrich our analysis.

This section continues with a brief introduction to SA and elaborates on the steps to solve the $F_m\,|mixed, blk|\sum C_j$ problem using this adapted algorithm. It is then followed by a brief explanation on IGCD and IGVD algorithms.

A. A BRIEF INTRODUCTION TO SIMULATED ANNEALING

Developed by [19], [20], SA is a stochastic local search algorithm best suited for solving large scale combinatorial problems. SA is applied by researchers in various fields, among which, [21] were the first to investigate SA applications in scheduling.

SA is inspired by the physical annealing process of metals. Slow cooling, from the annealing concept, has been interpreted as slowing down the probability of accepting weak individuals in a search procedure that seeks for finding low-energy, -objective value, solutions. Starting with a random initial solution, SA applies various neighborhood methods to generate new solutions in each iteration. The probability-based acceptance of the weaker solution enables SA to avoid being trapped in the local minima. In so doing, SA guides the search procedure to find the (near-)optima until certain conditions, stopping criterion, are met. Different extensions are proposed to the original SA among which acceptance criterion and the neighborhood search methods are more prevalent; these concepts are now explained.

1) ACCEPTANCE CRITERION

The desire for reaching out to a low-energy state is a fundamental rule of physics. Inspired by this rule, the move towards an optima state forms the optimization search algorithm [19]. $f(E_1)$ and $f(E_2)$ are the objective function, energy, values of the best current and new solutions. The difference between the objective function values, $\Delta E = f(E_2) - f(E_1)$, at any stage of the search procedure, triggers the move towards better solutions. A new solution will be accepted, and considered as the current solution, if $\Delta E < 0$. Otherwise, the Boltzmann probability function, $e^{(-\Delta E/kT)}$, helps make the acceptance decision. In this function, k is the Boltzmann constant, and T is the system temperature. Given a random number uniformly distrusted between 0 and 1, φ, a weaker solution can be accepted only if $\varphi < e^{(-\Delta E/kT)}$. This acceptance approach enables the search algorithm to explore less likely solution areas, and, consequently, reduces the prospect of being trapped at local optima [22]. The Boltzmann probability function approach works well when the algorithm is executed for a large number of iterations [19].

2) NEIGHBORHOOD SEARCH METHOD

There is a handful of neighborhood search procedures in the literature, each of which is developed for use in specific domains. Suggested by [23], *Swap, Insertion, Inversion,* and *Scramble* methods are suitable for the sort of PFSPs. A roulette wheel selection (RWS) mechanism could help select one of the above-mentioned basic procedures at each search step.

Using two randomly selected positions from the current solution, these methods apply different basic procedures to generate neighborhood solutions. The swap method exchanges the corresponding elements to x and y positions. The Insertion method inserts the x position element next to the y element. The inversion method reverses the sequence between x and y. Finally, the scramble method randomly displaces the elements that are located between x and y positions. The illustrative examples in Figure 2 show the mentioned procedures.

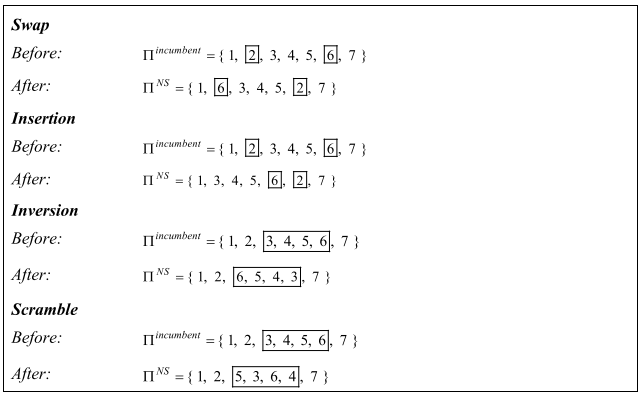

FIGURE 2. A generic local search procedure of the ESA.

To increase the chance of selecting the best basic procedure at each neighborhood search, one should refer to the sum of the fitness values recorder from the previous iteration. The procedure resulting in the highest improvement in the previous iteration should be applied in the current iteration. Given the fitness values fv_t, the sum of fitness values can be calculated by $Sfv = \sum_{t=1}^{4} fv_t$; the correlate fitness value of each neighborhood search is calculated using $Cfv_t = (\sum_{i=1}^{t} fv_i)/Sfv$ formula when $t = \{1, 2, 3, 4\}$. In this definition, Cfv_t refers to the likelihood of each basic procedure taking place in the next neighborhood search. To apply the RWS mechanism, a random number $prob_{NS}$ from [0, 1] interval will be generated at each stage. Given $prob_{NS}$ and Cfv_t values, the neighborhood search basic procedure can be selected. Once a procedure is applied, the corresponding fitness value must be added by one, if the new neighborhood solution is improved ($fv_t = fv_t + 1$), or subtracted by one ($fv_t = fv_t - 1$), otherwise. Replicating the same procedure may result in being trapped in one locality; to avoid it, the last fitness value should be set to a minimum value (ex., 10) once the same neighborhood search procedure is applied; this is to ensure that the underperforming neighborhood search procedures can also be tried with a small probability.

B. EXTENDED SIMULATED ANNEALING

This study proposes the Extended Simulated Annealing (ESA) algorithm to solve the $F_m\,|mixed, blk|\sum C_j$ problem. Figure 3 is a brief illustration of the solution procedure.

We now delve deeper into these steps and procedures.

1) SOLUTION INITIALIZATION

Similar to other heuristics, the first step to the ESA algorithm is to randomly generate the initial solution(s). For this purpose, a new NEH-based method, named NNEH, is applied to ensure that the initialization is in an acceptable quality. This method was initially proposed by [3], where two variants of NEH-based methods, NEH-Raj [24] and NEH-WPT [25], are combined to sort the initial list of NNEH. The procedure consists of the following steps:

ESA algorithm pseudocode

(1) Initialize T, λ, I_{iter}, I=0, and $fv = 25$;
(2) Generate the initial solution $\boldsymbol{\pi}^{initial}$;
(3) Let $\boldsymbol{\pi}^{best} =: \boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{initial}$ and $f(\boldsymbol{\pi}^{best}) = f(\boldsymbol{\pi}^{incumbent}) = f(\boldsymbol{\pi}^{initial})$;
(4) ***while*** (termination creiterion not satisfied) ***do***
(5) Calculate $Cfv_t = (\sum_{i=1}^{t} fv_i) / (\sum_{t=1}^{4} fv_t)$ of every NS;
(6) Generate a number $prob_{NS} \sim U(0,1)$;
(7) ***if*** $(Cfv_{t-1} < prob_{NS} \leq Cfv_t)$ ***then***
(8) Excute NS_t, obtain a new solution $\boldsymbol{\pi}^{new}$ with $f(\boldsymbol{\pi}^{new})$; I = I+1;
(9) ***if*** $f(\boldsymbol{\pi}^{new}) < f(\boldsymbol{\pi}^{best})$ ***then***
(10) $fv_t = fv_t + 1$;
(11) $\boldsymbol{\pi}^{best} =: \boldsymbol{\pi}^{new}$; $\boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{new}$;
(12) $f(\boldsymbol{\pi}^{best}) = f(\boldsymbol{\pi}^{new})$; $f(\boldsymbol{\pi}^{incumbent}) = f(\boldsymbol{\pi}^{new})$;
(13) ***if*** $f(\boldsymbol{\pi}^{best}) < f(\boldsymbol{\pi}^{new}) < f(\boldsymbol{\pi}^{incumbent})$ ***then***
(14) $fv_t = fv_t + 1$;
(15) $fv_t = fv_t + 1$;
(16) $\boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{new}$;
(17) $f(\boldsymbol{\pi}^{incumbent}) = f(\boldsymbol{\pi}^{new})$;
(18) ***elseif*** $(fv_t > 10)$ ***then***
(19) $fv_t = fv_t - 1$;
(20) ***if*** $(I = I_{iter})$ ***then***
(21) $T =: \lambda T$;
(22) $I = 0$;
(23) Calculate $e^{-\Delta E/T}$;
(24) Generate a number $\gamma \sim U(0,1)$;
(25) ***if*** $(\gamma < e^{-\Delta E/T})$ ***then***
(26) $\boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{new}$;
(27) $f(\boldsymbol{\pi}^{incumbent}) = f(\boldsymbol{\pi}^{new})$;
(28) ***goto*** (4);
(29) ***endwhile***
(30) ***return*** $\boldsymbol{\pi}^{best}$
(31) ***end***

FIGURE 3. A generic procedure of the ESA algorithm.

Step 1.1. Calculate $A[k]$ for each job, using Equation (7).

$$A[k] = \alpha \times \left(\sum_{j=1}^{m} (m - j + 1) \times p_{j,[k]}\right) + (1 - \alpha) \times \sum_{j=1}^{m} p_{j,[k]} \quad (7)$$

In this equation, $p_{j,[k]}$ is the processing time of the job $[k]$ on the machine j, and $\alpha \in (0, 1)$ is a parameter. As suggested by [3], a coefficient rate $\alpha = 0.1$ is used to calculate the $A[k]$ values in our experiments. It is worthwhile mentioning that the computational time increases when $p_{j,[k]}$ gets larger.

Step 1.2. Arrange the jobs in a non-decreasing order of $A[k]$ to form the initial list $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_n\}$.

Step 1.3. Given the first two jobs from $\boldsymbol{\pi}$, find the best partial sequence, β, which accounts for smaller TCT, and set k to 3.

Step 1.4. Select the k_{th} job form $\boldsymbol{\pi}$ and insert it into all possible positions in β without changing the relative position of the already assigned jobs. Then find the sequence with the best, smallest, TCT.

Step 1.5. Repeat Step 1.4. until $n \neq k$; otherwise, stop the initialization process. $\boldsymbol{\pi}^{initial}$ should be now attained considering the smallest TCT.

2) NEIGHBORHOODS SEARCH TO FIND NEW SOLUTIONS

Step 2.1. Set the best and incumbent solutions equal to the initialized solution obtained in Step 1 ($\boldsymbol{\pi}^{best}, \boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{initial}$).

Step 2.2. Calculate $f(\boldsymbol{\pi}^{incumbent})$, the objective function value associated with $\boldsymbol{\pi}^{incumbent}$.

Step 2.3. Apply the roulette wheel selection mechanism to randomly select a neighborhood search procedure among Swap, Insertion, Inversion, and Scramble methods mentioned in subsection 3.1.2.

Step 2.4. Calculate $f(\boldsymbol{\pi}^{new})$, the new solutions' objective function value.

3) ACCEPT/REJECT THE NEW SOLUTIONS

For each neighborhood solution, $\boldsymbol{\pi}^{NS}$, if its objective function value is better than that of the current best solution, $\boldsymbol{\pi}^{best}$, i.e., $f(\boldsymbol{\pi}^{NS}) < f(\boldsymbol{\pi}^{best})$, accept $\boldsymbol{\pi}^{NS}$ and let $\boldsymbol{\pi}^{best} =: \boldsymbol{\pi}^{NS}$ and $\boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{NS}$; if the associated objective function value is better than that of the incumbent solution, $\boldsymbol{\pi}^{incumbent}$, that is $f(\boldsymbol{\pi}^{best}) < f(\boldsymbol{\pi}^{NS}) < f(\boldsymbol{\pi}^{incumbent})$, accept $\boldsymbol{\pi}^{NS}$ and let $\boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{NS}$; Otherwise, use Equation (8), suggested by [22], for the accepting measure. Given that $\Delta E > 0$, if the randomly generated number is $\gamma \leq e^{-\Delta E/T}$, replace $\boldsymbol{\pi}^{incumbent}$ with $\boldsymbol{\pi}^{NS}$.

$$P = \begin{cases} 1, & if\, \gamma \leq e^{-\Delta E/T} \\ 0, & otherwise \end{cases} \quad (8)$$

In this probability function, γ is a random number uniformly generated between 0 to 1; $\Delta E = f(\boldsymbol{\pi}^{NS}) - f(\boldsymbol{\pi}^{incumbent})$ demonstrates the difference between objective function values of $\boldsymbol{\pi}^{NS}$ and $\boldsymbol{\pi}^{incumbent}$; and T is the current temperature. In our experiments, the initial value of T is calculated using $T = \sum_{i=1}^{m} \sum_{i=1}^{n} p_{ij}/n \times m \times 10$. Following Boltzmann annealing mechanism and to ensure the search for global optima, T is set to incline logarithmically, meaning that the probability of accepting weak solutions decreases by time. That is, T is decreased after running I_{iter} from the previous temperature decrease, according to the formula $T \leftarrow \lambda T$, where $0 < \lambda < 1$.

The above procedure continues until a prespecified termination criterion, maximum CPU time in this study, is met.

C. ITERATED GREEDY-BASED ALGORITHMS

Given the proven track record of IG-based algorithms in solving BPFSPs [9], two variants of this algorithm, named IGCD and IGVD, are considered to compare their performance with that of the ESA algorithm. Proposed by [26], the IG-based algorithms are stochastic metaheuristics that employ solution construction methods to iteratively modify incumbent individuals until an acceptance criterion allows for replacing the current best solution. IG-based algorithms are applied to solve a wide variety of problems,

including PFSPs, first of which were applied by [27]. The major elements of IG-based algorithms are now briefly explained.

1) INITIALIZATION

An initialization method similar to that of the ESA algorithm is used. Reminding from the last section, the initial job sequence $\boldsymbol{\pi}^{initial}$, with the objective function value $f(\boldsymbol{\pi}^{initial})$, is the result of the initialization process. Therefore, the first iteration starts with the following setting: $\boldsymbol{\pi}^{best}, \boldsymbol{\pi}^{incumbent} =: \boldsymbol{\pi}^{initial}$ and $f(\boldsymbol{\pi}^{best}) = f(\boldsymbol{\pi}^{incumbent}) = f(\boldsymbol{\pi}^{initial})$.

2) DESTRUCTION AND CONSTRUCTION PHASE

Destruction phase consists of random removal of d jobs from the $\boldsymbol{\pi}^{incumbent}$ list. The resulting partial sequence, comprising of $n - d$ remaining jobs, is defined as $\boldsymbol{\pi}^{remain}$, while the removed jobs from the $\boldsymbol{\pi}^{remove}$ sequence. In the construction phase, algorithm sequentially inserts the $job_{[k]}, k = 1, \ldots, d$, from the π^{remove} list, into every possible position in π^{remain} until a better partial sequence, π^{temp}, is resulted. This procedure will be continued until π^{remove} is empty. Eventually, the new job sequence, $\boldsymbol{\pi}^{new}$, is expected to be associated with better objective function value, $f(\boldsymbol{\pi}^{new})$.

It is worthwhile mentioning that the parameter d is the differentiating point between IGCD and IGVD, where the former approach assumes a constant d value, while the latter one considers a variable value. Using two variants of the IG-based algorithm helps improve numerical analysis by including more perturbation in the results.

3) ACCEPTANCE AND STOPPING CRITERIA

Given $\boldsymbol{\pi}^{new}$ from the construction phase, if the associated objective function value is better than that of the incumbent solution, $f(\boldsymbol{\pi}^{new}) < f(\boldsymbol{\pi}^{incumbent})$, the incumbent solution will be replaced by $\boldsymbol{\pi}^{new}$; otherwise, an acceptance probability function similar to that of ESA will help to make the acceptance/rejection decision.

To ensure a fair comparison in numerical analysis, including the comparison between IGCD and IGVD, maximum CPU time is considered as the stopping criterion for the IG-based algorithms.

IV. COMPUTATIONAL RESULTS AND DISCUSSION

The numerical results obtained in this study are now compared to that of the best-performing algorithm for solving MBPFSPs. Considering TCT as the objective function, performance measure, IGCD, IGVD, and the ESA algorithm along with the CGLS as the benchmark algorithm, are all coded and compiled by the authors on the C++ programming language. The experiments are run on a personal computer with the following specs: Intel®Core (TM) i7-7700 CPU, 32GB RAM, and Windows 7 operating system.

A. TEST PROBLEMS

A total of 150 test instances are considered for numerical analysis. The test instances were expanded from the benchmark datasets provided by [28], which is broadly accepted in the scheduling literature. The first 120 instances comprise 12 different combinations of jobs and machines, as following: {20, 50, 100}×{5, 10, 20}, {200}×{10, 20}, and {500}×{20}, where the former value indicates the number of jobs, and the latter one specifies the number of machines. Each of the mentioned combinations consists of 10 different instances. Three missing combinations, each of which consisting of 10 different instances, from the original dataset, {200}×{5} and {500}×{5, 10}, are compensated by the alternatives generated by [29].

The Relative Percentage Deviation (RPD), Equation (9), is used to measure the effectiveness of the algorithm both in parameter calibration, and the comparisons in numerical analysis.

$$RPD = \frac{f(\boldsymbol{\pi}) - f(\boldsymbol{\pi}^{best})}{f(\boldsymbol{\pi}^{best})} \times 100\% \tag{9}$$

In this equation, $f(\boldsymbol{\pi})$ refers to the objective function value for a given run using a specific test instance. In the calibration process, $f(\boldsymbol{\pi}^{best})$ indicates the best-found function value when different λ values in the ESA, and d values in the IG-based algorithms, are considered. In results analysis, $f(\boldsymbol{\pi}^{best})$ refers to the best-found solution when comparing the ESA with CGLS, IGCD, and IGVD algorithms using the optimum algorithm setting. In addition to RPD, average RPD is computed to compare the effectiveness of the algorithms overall numerical instances.

B. ALGORITHM PARAMETERS CALIBRATION

Before stepping into results analysis, the ESA algorithm settings, I_{iter} and λ, and that of the IG-based algorithms, d and T_0, need to be determined. Different levels for each of the above-mentioned parameters are considered to design the numerical experiments. On this basis, the calibration test configuration, consisting of the combinations shown in Tables 1 and 2, is conducted on nine randomly generated test instances.

Using five replications for each of the instances comprising $n \in \{35, 150, 250\}$ jobs $m \in \{5, 10, 20\}$ machines, and given the combinations in Table 1, a total of $9 \times 9 \times 5 = 405$ experiments are identified to find the best setting and tune the ESA algorithm. Similarly, 21 possible combinations of the IG-based algorithm parameters and five replications of each instance results in a total of $21 \times 9 \times 5 = 945$ runs in the destruction phase.

Calibration results are shown in Tables 3 and 4; considering the best values in these tables, the ESA key parameters are set to $I_{iter} = 200$ and d, while the IGCD and IGVD algorithms' parameters are fixed at $T = 0.5$ and $d = 3$ in IGCD, and $d \in [1, 6]$ in IGVD. Besides, the termination criterion for all of the experiments is considered equivalent and equal to

TABLE 1. Test parameters for I_{iter} and λ combinations.

Parameters	Combinations								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
I_{ter}	100	100	100	150	150	150	200	200	200
λ	0.90	0.93	0.95	0.90	0.93	0.95	0.90	0.93	0.95

TABLE 2. Test parameters for d and T_0 combinations.

Parameters	Combinations										
	1	2	3	4	5	6	7	8	9	10	11
d	1	1	1	2	2	2	3	3	3	4	4
T_0	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3
	12	13	14	15	16	17	18	19	20	21	
d	4	5	5	5	6	6	6	[1,6]	[1,6]	[1,6]	
T_0	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	

TABLE 3. Comparison results for I_{iter} and λ combinations (best in bold).

Jobs	Machines	Combinations								
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
35	5	0.122	0.000	0.043	0.176	0.469	0.556	0.084	0.208	0.191
	10	0.000	0.257	0.139	1.017	0.722	0.318	0.630	1.029	0.653
	20	0.000	0.545	0.287	0.456	0.835	1.018	0.209	0.536	0.822
150	5	0.869	0.596	0.450	0.163	0.000	0.312	0.076	0.062	0.426
	10	0.480	0.492	0.158	0.243	0.636	0.242	0.416	0.057	0.000
	20	0.485	0.053	0.343	0.278	0.366	0.647	0.369	0.621	0.000
250	5	0.444	1.330	1.521	0.644	1.268	0.509	0.353	0.560	0.000
	10	0.336	0.262	0.695	1.086	0.218	1.168	0.586	0.059	0.000
	20	0.022	0.027	0.750	0.909	0.706	0.155	0.132	0.243	0.000
Average RPD		0.307	0.396	0.487	0.552	0.580	0.547	0.317	0.375	**0.232**

TABLE 4. Comparison results for d and T combinations (best in bold).

job	machine	Combinations																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
35	5	6.905	5.997	7.492	3.668	3.218	2.771	3.693	2.073	2.934	2.953	3.639	2.545	4.261	2.843	2.533	3.094	3.388	4.124	3.058	3.265	3.563
	10	6.693	6.430	5.474	4.432	3.537	3.264	2.853	2.520	1.882	3.218	2.536	2.834	3.460	2.706	2.758	2.883	2.422	3.512	3.447	1.944	1.707
	20	4.374	3.082	3.497	3.228	2.791	1.092	2.094	1.494	1.404	2.113	2.230	1.839	2.078	1.604	1.816	2.294	1.768	2.121	2.256	1.821	1.820
150	5	3.642	3.189	2.911	1.082	0.540	1.663	1.236	0.662	0.793	0.594	0.762	0.554	1.068	1.134	0.952	1.403	1.890	1.022	1.135	1.347	1.731
	10	5.073	4.231	4.394	1.223	1.491	1.607	1.703	2.045	1.878	2.382	1.826	2.526	3.472	2.844	2.742	3.088	4.278	3.606	2.417	2.179	2.413
	20	3.516	3.541	3.184	1.126	1.138	0.898	1.606	1.488	0.785	1.966	1.917	2.067	2.760	3.125	2.013	2.542	2.667	3.352	1.553	1.540	1.194
250	5	3.561	3.352	3.796	0.950	1.417	1.960	1.816	1.894	1.515	2.403	2.940	2.455	2.935	2.837	2.862	3.828	3.871	3.694	1.562	1.260	1.721
	10	2.142	1.985	2.017	0.568	0.551	0.614	1.076	1.196	0.636	1.748	1.089	1.814	2.085	2.358	2.421	2.613	2.678	2.741	0.853	0.893	1.447
	20	2.462	2.044	2.118	1.185	1.070	0.808	1.344	1.605	1.615	1.641	1.780	2.245	2.664	2.467	2.792	3.137	2.915	3.340	1.550	1.370	1.403
Average RPD		4.263	3.761	3.876	1.940	1.750	1.631	1.936	1.664	**1.493**	2.113	2.080	2.098	2.754	2.435	2.321	2.765	2.875	3.057	1.981	1.736	1.889

$T_{\max} = 30 \times n \times m$milliseconds to ensure a fair comparison among the benchmark algorithms. Setting the mentioned parameter values, we now proceed to the final numerical experiments and results analysis.

TABLE 5. Average RPD of the numerical results in all 150 instances (best in bold).

Job	Machine	IG_{CD}		IG_{VD}		CGLS		ESA	
		Min	Ave.	Min	Ave.	Min	Ave.	Min	Ave.
20	5	1.250	1.843	1.077	1.810	1.110	1.397	0.000	0.343
	10	1.113	1.943	1.340	2.176	1.393	1.686	0.000	0.702
	20	0.794	1.257	0.976	1.447	0.856	0.943	0.030	0.678
50	5	4.280	5.423	4.714	5.841	1.910	2.187	0.000	0.614
	10	4.601	5.446	4.910	6.032	2.120	2.252	0.000	0.951
	20	3.431	4.374	3.695	4.603	1.270	1.382	0.000	0.677
100	5	4.928	5.717	5.759	6.396	1.611	1.648	0.000	0.452
	10	6.160	7.070	6.893	7.675	1.910	2.035	0.000	0.737
	20	5.725	6.575	6.421	7.152	2.209	2.302	0.000	0.584
200	5	9.184	9.794	9.551	10.252	2.395	2.441	0.000	0.286
	10	10.165	10.808	10.429	11.218	2.704	2.738	0.000	0.539
	20	8.592	9.467	9.068	9.802	2.547	2.576	0.000	0.672
500	5	14.151	14.723	14.532	15.127	2.757	2.757	0.000	0.188
	10	13.283	13.818	13.439	14.129	1.550	1.553	0.000	0.287
	20	11.002	11.521	11.108	11.701	1.249	1.252	0.001	0.304
Average RPD		6.577	7.319	6.928	7.691	1.839	1.943	**0.002**	0.534

TABLE 6. ANOVA analysis of the algorithm performances, considering 0.05 confidence interval.

Source	Degree of Freedom	Sum of Squares	Mean Square	F-value	p-value
Solution approach	3	5363.678	1787.893	180.117	0.000
Error	596	5916.078	9.926		
Total	599	11279.755			

TABLE 7. Statistical results for the algorithms pairwise comparison, considering 0.05 confidence level.

	Paired differences			t-value	Degree of Freedom	Sig. (single-tailed)
	Mean	Std. Deviation	Std. Error Mean			
IG_{CD} - SA	6.575	4.381	0.358	18.383	149	0.000
IG_{VD} - SA	6.926	4.418	0.361	19.197	149	0.000
CGLS - SA	1.837	1.026	0.084	21.937	149	0.000

C. NUMERICAL RESULTS

This section evaluates the performance of the ESA for solving the $F_m\,|mixed, blk|\sum C_j$ problem. For this purpose, a comparative analysis is conducted to compare the ESA's numerical outcomes with that of CGLS, IGCD, and IGVD. Similar to the calibration phase, and for the sake of fairness, a maximum CPU time of $T_{\max} = 30 \times n \times m$ milliseconds is considered to conduct all the numerical experiments.

Given TCT resulting after each run, an RPD value for every instance, and an average RPD value for ten instances in each combination are calculated. The computational results, considering the minimum and average of ARPD values, are shown in Table 5. The first column indicates the minimum RPD value found amongst all the 5 replications, and the second column is the mean value for 5 replications of each instance. The average RPD values yielded by the ESA, CGLS, IGCD, and IGVD algorithms are 0.002, 1.839, 6.577,

TABLE 8. Computational results, including the best-known solutions for all test instances.

Instance	BFS*	Instance	BFS	Instance	BFS	Instance	BFS	Instance	BFS
20-05		50-05		100-05		200-05		500-05	
1	22378	1	118278	1	498902	1	1878140	1	10583875
2	15381	2	76695	2	477403	2	1681404	2	10741643
3	19236	3	121152	3	458635	3	1826145	3	11587732
4	23086	4	123889	4	428885	4	1744602	4	10513884
5	19525	5	108996	5	414439	5	1853817	5	11368707
6	21144	6	119057	6	388698	6	1750995	6	11228023
7	22483	7	106884	7	482655	7	1736488	7	10180845
8	19534	8	105474	8	440144	8	1832597	8	11305784
9	21509	9	95310	9	423996	9	1721893	9	11395865
10	17967	10	124900	10	465027	10	1665977	10	10533874
20-10		50-10		100-10		200-10		500-10	
1	31360	1	152364	1	550070	1	1907793	1	12197824
2	30690	2	149596	2	496069	2	1751856	2	12257349
3	25648	3	133589	3	525048	3	1994177	3	12592189
4	27583	4	148383	4	619356	4	2067546	4	12654808
5	27723	5	127443	5	548540	5	1899404	5	12583819
6	25792	6	134087	6	555549	6	1895208	6	12717512
7	26630	7	113690	7	476869	7	2097912	7	11492742
8	27314	8	141199	8	555633	8	2126058	8	12454794
9	28353	9	149768	9	566578	9	2211999	9	11773115
10	29570	10	154257	10	584665	10	2024314	10	12501227
20-20		50-20		100-20		200-20		500-20	
1	39727	1	193493	1	606838	1	2396698	1	13666511
2	41516	2	177196	2	638753	2	2184037	2	13843184
3	42462	3	181835	3	646037	3	2449578	3	13934275
4	41783	4	189773	4	648467	4	2297215	4	13127729
5	41242	5	179978	5	576630	5	2333145	5	14304923
6	41358	6	187825	6	638489	6	2544550	6	15222423
7	42282	7	166945	7	634713	7	2245926	7	12551595
8	40364	8	183709	8	640119	8	2448139	8	12836705
9	43085	9	185342	9	580382	9	2522401	9	13868320
10	38617	10	186424	10	640399	10	2309391	10	13371978

BFS*: Best Found Solution

and 6.928, respectively. This initial finding suggests that the ESA algorithm performs better when compared to the other three solution approaches.

CGLS is the best-performing algorithm in MBPFSP literature when considering makespan as the objective function. However, the graphical demonstration of the experimental results, Figure 4, shows that the ESA developed our study outperforms CGLS in optimizing TCT. Besides, no significant difference is detected amongst IG-based algorithms and CGLS when considering small-size instances. However, and in contrast to CGLS that keeps a steady performance over larger instances, the average RPD values of the IG-based algorithms increases sharply when medium- and large-size instances are considered, as shown in Figure 5.

Next, analysis of variance, ANOVA, is carried out to support numerical results in confirming the superiority of the developed solution approach. Table 6 summarizes the statistical results from the performance difference test among the presented solution algorithms, at a $\alpha = 0.05$ confidence level. Given the negligible p-value, it can be concluded that there is

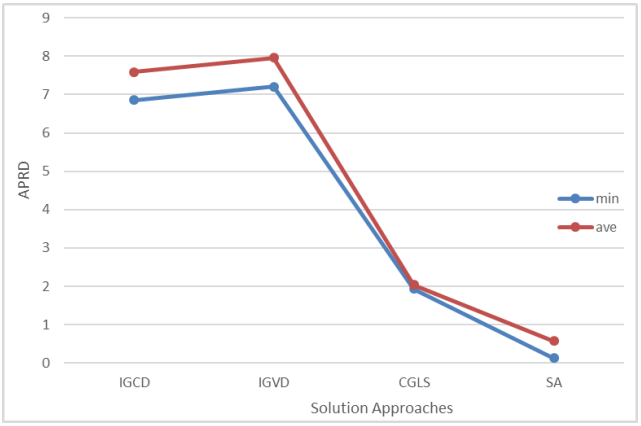

FIGURE 4. Average RPDs for the compared algorithms.

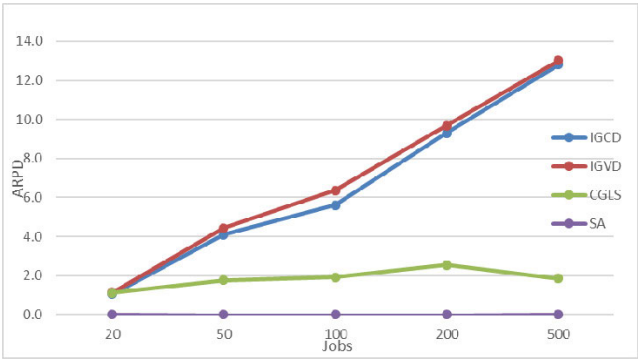

FIGURE 5. Average RPD of the algorithms considering the various number of jobs.

a significant difference between the ESA's performance and that of the other presented algorithms.

The above assertion is unidirectional; hence, it can be further evaluated using the one-tailed t-test. The t-test results of the pairwise comparisons, at a $\alpha = 0.05$ confidence level, are summarized in Table 7. Evidently, it can be confirmed that the ESA algorithm demonstrates a meaningfully better performance when compared to CGLS, IGCD, and IGVD.

Finally, the best-found solutions for each of the 150 benchmark test instances are shown in Table 8. Of the best-found solutions, 148 are obtained applying ESA; a 98 percent success rate that proves the effectiveness of the proposed approach when compared to the other compared algorithms. The high yield rate, along with RPD and ARPD comparison using ANOVA and t-test, provides sufficient evidence to support the hypothesis that the ESA outperforms the rest of the examined solution approaches. Besides, it is shown that the ESA demonstrates more robustness when solving medium- and large-size MBPFSPs, making it the best existing solution approach to solve industrial-scale problems.

V. CONCLUSION

Given the growing need for expansion of production activities and the limited space in manufacturing plants, it is not realistic to assume an unlimited buffer zone in product scheduling problems. Besides, cost-saving and just-in-time policies discourage the storage of WIP items. In this situation, various blocking situations can result. There is not only one type of blocking constraint, and this needs to be considered in the optimization of scheduling problems. This study proposed to improve the SA algorithm to solve the $F_m\,|mixed, blk|\sum C_j$ problem, which is rather cumbersome in their industrial-scale applications.

Comparing the proposed solution algorithm with the best performing algorithm in the literature, CGLS, as well as two IG-based algorithms, IGCD and IGVD, it is shown that the ESA outperforms in medium- and large-size instances when considering TCT. Overall, the proposed algorithms yield the best-known solution in all instances. The proposed ESA is particularly more robust and effective when the number of jobs increases. The study further analyzed the results using ANOVA, where the test results confirmed a significant difference between the algorithms considering the low p-value.

Given the limited body of the MBPFSPs' literature, the following research directions can help develop new extensions to the scheduling literature. First, applying mixed-blocking constraints on different operating environments is a worthwhile research direction to be pursued; job-shop and open-shop scheduling are only some examples of this possible research direction. Second, given the significance of setup times in the accuracy of the scheduling outcomes [30], MBPFSP with sequence-dependent setup times is another research direction to pursue. Finally, considering conflicting objectives, like makespan and TCT, within a multi-objective scheme is a promising room for extending MBPFSPs.

REFERENCES

[1] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," *Nav. Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, Mar. 1954.
[2] E. V. Levner, "Optimal planning of parts' machining on a number of machines," *Autom. Remote Control*, vol. 12, no. 12, pp. 1972–1978, 1969.
[3] V. Riahi, M. Khorramizadeh, M. A. H. Newton, and A. Sattar, "Scatter search for mixed blocking flowshop scheduling," *Expert Syst. Appl.*, vol. 79, pp. 20–32, Aug. 2017.
[4] W. Trabelsi, C. Sauvey, and N. Sauer, "Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems," *Comput. Oper. Res.*, vol. 39, no. 11, pp. 2520–2527, Nov. 2012.
[5] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Oper. Res.*, vol. 44, no. 3, pp. 510–525, Jun. 1996.
[6] S. Martinez, S. Dauzère-Pérès, C. Guéret, Y. Mati, and N. Sauer, "Complexity of flowshop scheduling problems with a new blocking constraint," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 855–864, Mar. 2006.
[7] D. P. Ronconi and L. R. S. Henriques, "Some heuristic algorithms for total tardiness minimization in a flowshop with blocking," *Omega*, vol. 37, no. 2, pp. 272–281, Apr. 2009.
[8] 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, Mar. 2010.
[9] 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, Jun. 2011.
[10] 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, Apr. 2013.
[11] M. F. Tasgetiren, D. Kizilay, Q.-K. Pan, and P. N. Suganthan, "Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion," *Comput. Oper. Res.*, vol. 77, pp. 111–126, Jan. 2017.

[12] Z. Shao, D. Pi, and W. Shao, "Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem," *Eng. Optim.*, vol. 50, no. 5, pp. 894–916, May 2018.
[13] M. Khorramizadeh and V. Riahi, "A bee colony optimization approach for mixed blocking constraints flow shop scheduling problems," *Math. Problems Eng.*, vol. 2015, pp. 1–10, Feb. 2015.
[14] V. Riahi, M. A. H. Newton, K. Su, and A. Sattar, "Constraint guided accelerated search for mixed blocking permutation flowshop scheduling," *Comput. Oper. Res.*, vol. 102, pp. 102–120, Feb. 2019.
[15] 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, Jan. 1983.
[16] M. C. Bhuvaneswari and G. Subashini, "Scheduling in Heterogeneous Distributed Systems," in *Application of Evolutionary Algorithms for Multi-Objective Optimization in VLSI and Embedded Systems*. New Delhi, India: Springer, 2015, pp. 147–169.
[17] J. M. Framinan and R. Leisten, "An efficient constructive heuristic for flowtime minimisation in permutation flow shops," *Omega*, vol. 31, no. 4, pp. 311–317, Aug. 2003.
[18] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Math.*, vol. 5, pp. 287–326, 1979.
[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953.
[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[21] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, Jan. 1989.
[22] S.-W. Lin, C.-J. Ting, and K.-C. Wu, "Simulated annealing with different vessel assignment strategies for the continuous berth allocation problem," *Flexible Services Manuf. J.*, vol. 30, no. 4, pp. 740–763, Dec. 2018.
[23] A. Sioud and C. Gagné, "Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 264, no. 1, pp. 66–73, Jan. 2018.
[24] C. Rajendran, "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime," *Int. J. Prod. Econ.*, vol. 29, no. 1, pp. 65–73, Feb. 1993.
[25] L. Wang, Q.-K. Pan, and M. F. Tasgetiren, "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem," *Comput. Ind. Eng.*, vol. 61, no. 1, pp. 76–83, Aug. 2011.
[26] L. W. Jacobs and M. J. Brusco, "Note: A local-search heuristic for large set-covering problems," *Nav. Res. Logistics*, vol. 42, no. 7, pp. 1129–1140, Oct. 1995.
[27] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, Mar. 2007.
[28] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993.
[29] Q.-K. Pan and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," *Eur. J. Oper. Res.*, vol. 222, no. 1, pp. 31–43, Oct. 2012.
[30] C.-Y. Cheng, P. Pourhejazy, K.-C. Ying, S.-F. Li, and C.-W. Chang, "Learning-based Metaheuristic for scheduling unrelated parallel machines with uncertain setup times," *IEEE Access*, vol. 8, pp. 74065–74082, 2020.

CHEN-YANG CHENG received the Ph.D. degree in industrial and manufacturing engineering from Penn State University. He is currently a Professor with the Department of Industrial Engineering and Management, National Taipei University of Technology. His research interests include computer integrated manufacturing, human–computer interaction, distributed systems and control, and intelligent systems.

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, Taiwan. He is also with the Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan, and with the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His current research interests include meta-heuristics and data mining. His articles have appeared in *Computers & Operations Research*, the *European Journal of Operational Research*, the *Journal of the Operational Research Society*, the *European Journal of Industrial Engineering*, the *International Journal of Production Research*, the *International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, *Applied Soft Computing*, *Applied Intelligence*, *Expert Systems with Applications*, and IEEE Access.

POURYA POURHEJAZY received the master's degree in industrial engineering from University Technology Malaysia, in 2014, and the Ph.D. degree in logistics engineering from INHA University, South Korea, in 2017. He is currently an Adjunct Assistant Professor and a Research Assistant Professor with the National Taipei University of Technology, Taiwan. His research interests include optimization and decision analysis in the supply chain, production, and transportation contexts.

KUO-CHING YING is currently a Distinguished Professor with the Department of Industrial Engineering and Management, National Taipei University of Technology. His research interests include operations scheduling and combinatorial optimization, in which he has published over 100 academic research articles in refereed international journals. He is the (senior) Editor of ten international journals.

JIA-WEN ZHENG received the master's degree in industrial engineering and management from the National Taipei University of Technology, Taiwan. She is currently a Production Planning Administrator at Siliconware Precision Industries Company Ltd. (Zhong Ke Facility). Her research interests include operations research and production scheduling.