IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 6, 2018, accepted October 29, 2018, date of publication November 5, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2879600

Multi-Objective Parallel Variable Neighborhood Search for Energy Consumption Scheduling in Blocking Flow Shops

FUCAI WANG[1], GUANLONG DENG[1], TIANHUA JIANG[2], AND SHUNING ZHANG[1]
[1]School of Information and Electrical Engineering, Ludong University, Yantai 264025, China
[2]School of Transportation, Ludong University, Yantai 264025, China

Corresponding author: Guanlong Deng (dglag@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61403180, in part by the National Natural Science Foundation of China under Grant 61603172, and in part by the Shandong Provincial Natural Science Foundation of China under Grant ZR2016GP02.

ABSTRACT Blocking flow shop scheduling problem has been extensively studied because of its widespread industrial applications. However, the existing research mostly aims at makespan or total flow time minimization and ignores the criterion for energy saving. This paper investigates the blocking flow shop scheduling problem with both makespan and energy consumption criteria. First, the multi-objective model of blocking flow shop scheduling is formulated in consideration of machine energy consumed in blocking and idle time. Then, a multi-objective parallel variable neighborhood search (MPVNS) algorithm is proposed to solve this problem. An improved Nawaz–Enscore–Ham-based heuristic is developed to generate initial solutions, and a variable neighborhood search is designed to explore these solutions in parallel. Furthermore, an insertion-based pareto local search method is embedded to enhance the exploitation of the algorithm. Finally, in order to validate its effectiveness, the MPVNS is compare with other two effective multi-objective metaheuristics by computational experiments based on well-known benchmark instances. The experimental results illustrate that the proposed algorithm is superior to non-dominated sorting genetic algorithm (II) and bi-objective multi-start simulated annealing algorithm in terms of set coverage and hypervolume measures.

INDEX TERMS Scheduling, energy consumption, heuristic algorithms, multi-objective, flow shop, variable neighborhood search.

I. INTRODUCTION

The permutation flow shop scheduling problem (PFSP) is an attractive research field in manufacturing. Owing to its non-deterministic polynomial time (NP)-hard characteristic, the PFSP is difficult to solve optimally even for a moderate size problem. Consequently, efficient methods or metaheuristics for the PFSP have been a topic of interest for researchers in the last few decades. The existing research on PFSP includes two categories. One aims at the optimization of a single objective, and the other takes simultaneously more than one objective into consideration, which is called multi-objective PFSP.

Because in many manufacturing situations, the decision maker is concerned with more than one objective [1], the multi-objective PFSP has wide applications in practice and attracts attention of many researchers in recent years. Loukil *et al.* [2] adopted the concept of non-dominated solutions and presented a bi-objective simulated annealing algorithm for the problem. Ishibuchi *et al.* [3] tackled the problem with three objectives, makespan, total flow time, and max tardiness. An enhanced bi-objective simulated annealing algorithm with multi-starts was proposed by Lin and Ying [4] to minimize makespan and total flow time. Armentano and Arroyo [5] presented a tabu search method with makespan and total flow time criteria. Minella *et al.* [6] developed a restarted pareto iterated greedy algorithm, solving not only the case of makespan and total flow time criteria, but also the case of makespan and total tardiness criteria. Frosolini *et al.* [7] proposed a modified harmony search algorithm for the multi-objective PFSP with due-dates. Besides, the problem was also treated by some other intelligent algorithms, such as genetic algorithm [8], multi-objective ant colony system algorithm [9], particle swarm optimization [10], hybrid differential evolution method [11], and

2169-3536 © 2018 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.

hybrid multi-objective shuffled frog-leaping algorithm [12]. For further comprehensive surveys on the research and development of the multi-objective PFSP, the reader is referred to Minella *et al.* [13] and Yenisey and Yagmahan [14].

The classical PFSP assumes that there are infinite intermediate buffers between two consecutive machines. However, the buffers are limited in some real-world manufacturing environments such as steel-making industry [15]. If the buffers are zero, a job that finishes a prior operation cannot always leave the incumbent machine immediately, because it has to stay on the incumbent machine until the posterior operation is ready to commence. In other words, a job tends to block itself on a machine, and the next operation is probably delayed. The problem with this constraint is called blocking flow shop scheduling problem (BFSP). The BFSP differs much from the PFSP, and it can be found in numerous industrial applications, such as batch plant, steal-making, petrochemical, and plastics molding processes [16], [17]. The BFSP has also attracted much attention in recently years. It was proved that the problem is NP-hard for more than two machines [18]. A branch and bound method was developed by Companys and Mateo [19], and the problem with small sizes was optimally solved. Bautista *et al.* [20] presented a bounded dynamic programming method, which was effective for small size instances but incapable of solving large size instances. Although researchers developed several effective heuristics for the problem [21]–[23], most of the prominent studies are focused on metaheuristics. In an earlier research report, Caraffa *et al.* [24] proposed a genetic algorithm for the makespan minimization. A tabu search method was later presented by Grabowski and Pempera [25]. The hybrid discrete differential evolution [26], iterated greedy [27], hybrid harmony search [28], and discrete artificial bee colony [29] algorithms were also developed for the problem. In two side papers, Han *et al.* [30] presented an improved artificial bee colony algorithm and later a discrete artificial bee colony incorporating differential evolution [31] for the BFSP with makespan criterion. Besides, they also considered the problem with total flow time criterion and proposed several effective hybrid discrete artificial bee colony algorithms [32]. Very recently, Han *et al.* [33] treated the blocking lot-streaming flow shop scheduling problem with machine breakdowns, and an effective evolutionary multi-objective algorithm was developed.

With regard to the optimization objective, the BFSP is usually treated with makespan or total flow time criterion whereas the energy related criterion is seldom taken into consideration. Nowadays, it is well-known that energy has become one of the most important resources in manufacturing, and the reduction of energy consumption has been a popular issue confronting the industrial sector. Because an optimized schedule can greatly reduce the energy consumption but require no financial investment, the energy consumption criterion in scheduling has drawn increasing attention. Shrouf *et al.* [34] proposed a mathematical model of energy consumption for a single machine scheduling problem. Liu [35] developed a non-dominated sorting genetic algorithm (NSGA-II) for batch scheduling with carbon emission criterion. Zhang and Chiong [36] considered the minimizations of total weighted tardiness and total energy consumption in a job shop scheduling problem and presented a multi-objective genetic algorithm with local search. With regard to the flow shop scheduling, Dai *et al.* [37] and Luo *et al.* [38] considered the flexible flow shop problem for energy efficiency. Ding *et al.* [39] investigated heuristics for the PFSP with carbon emission criterion. Lu *et al.* [40] tackled energy-efficient permutation flow shop scheduling problem, using a hybrid multi-objective backtracking search algorithm. For further reviews, the reader is referred to Gahm *et al.* [41]. Recently, Hansen *et al.* [42] developed an improved invasive weed optimization algorithm for reducing energy consumption in optimal chiller loading. Zheng and Li [43] considered setup energy consumption and presented an efficient multi-objective optimization algorithm for hybrid flow shop scheduling problem. However, to the best of our knowledge, the research on energy efficiency for BFSP is still in shortage. Furthermore, research on the multi-objective BFSP remains insufficient, although the multi-objective PFSP has been extensively studied by numerous researchers.

This study considers the energy consumption as well as makespan criteria in BFSP and develops a multi-objective variable neighborhood search algorithm for the multi-objective model. The variable neighborhood search (VNS) is a metaheuristic for solving combinatorial and global optimization problems. Its basic idea consists in a systematic change of neighborhood combined with a local search. Li *et al.* [44] provided a survey on VNS and pointed out that the algorithm was effective in numerous applications. Owing to its structural simplicity and easy implementation, the VNS algorithm has been successfully applied to the field of scheduling, including single machine [45], parallel machine [46], flow shop [47] and job shop scheduling [48].

With respect to multi-objective optimization algorithms, pareto-dominance-based method and decomposition-based method are recognized as two major types of approaches for approximating pareto front. The former type optimizes two or more objectives based on the dominance relation of solutions and maintains a non-dominated solution set during the algorithm process. The latter type decomposes a multi-objective optimization problem into a number of subproblems by linear or nonlinear aggregation functions and solves them simultaneously. The pareto-dominance-based method is easy to implement and usually employed for problems with no more than three objectives [13], [14], whereas the decomposition-based method needs delicate design for aggregation functions and performs well for many-objective optimization problems in which number of objectives may exceed three [49]–[51]. For this reason, this study is devoted to presenting a pareto-dominance-based method for the considered BFSP with two objectives. The purpose of multi-objective scheduling is usually to find a non-dominated solution set rather than a single solution; therefore, it is

significant that an algorithm for multi-objective scheduling needs to keep diversity to escape from local optima. In order to achieve diversification, the VNS is designed with multi-starts, and a multi-objective parallel variable neighborhood search algorithm (MPVNS) is proposed. In addition, an insertion-based pareto local search method is hybridized with the algorithm to enhance its exploitation.

The remainder of the paper is organized as follows. Section 2 provides the formulation of multi-objective BFSP. Section 3 discusses the details of the proposed MPVNS algorithm. In section 4, the algorithm is calibrated, and the comparison with other metaheuristics is discussed. Finally, Section 5 summarizes the conclusions and directions of our future work.

II. FORMULATION OF MULTI-OBJECTIVE BFSP

In a blocking flow shop scheduling problem (BFSP), there are n jobs (job j, $j = 1, 2, \ldots, n$) and m machines (machine M_i, $i = 1, 2, \ldots, m$). Each job has to be processed firstly on machine M_1, then on machine M_2, …, and lastly on machine M_m. The processing time of job j on machine M_i is known as p_{ji}. The blocking constraint exists in the production process, which means that there is no buffer between any two consecutive machines. To be specific, the following assumptions are used: (1) at any time, a machine is able to process at most one job, and a job is able to be processed on at most one machine; (2) no job splitting is allowed; (3) all the jobs and machines are available at time zero; (4) the set-up, release, and transfer time is omitted.

A feasible solution for the multi-objective BFSP is represented as a job permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, where π_j denotes a job to be processed. Let $d_{\pi(j),i}$ denote the departure time of job $\pi(j)$ from machine M_i, the departure time is obtained as follows.

$$d_{\pi(1),0} = 0 \tag{1}$$

$$d_{\pi(1),i} = d_{\pi(1),i-1} + p_{\pi(1),i}, \quad i = 1, \ldots, m-1 \tag{2}$$

$$d_{\pi(j),0} = d_{\pi(j-1),1}, \quad j = 2, \ldots, n \tag{3}$$

$$d_{\pi(j),i} = \max\{d_{\pi(j),i-1} + p_{\pi(j),i}, d_{\pi(j-1),i+1}\}, \quad j = 2, \ldots, n, \quad i = 1, \ldots, m-1 \tag{4}$$

$$d_{\pi(j),m} = d_{\pi(j),m-1} + p_{\pi(j),m}, \quad j = 1, \ldots, n \tag{5}$$

where $d_{\pi(j),0}$ denotes the start time of $\pi(j)$ on machine M_1.

Let $f_1(\pi)$ and $f_2(\pi)$ denote the makespan ($C_{\max}$) and energy consumption for permutation π, respectively. Then the makespan is directly obtained as

$$f_1(\pi) = C_{\max}(\pi) = d_{\pi(n),m} \tag{6}$$

Generally, the total energy consumption in a flow shop consists of energy consumption for the setup stage, transportation phase, machine idle stage, processing phase, and public use [40]. In this study, the energy consumption for setup stage, transportation phase, and public use is omitted for simplicity. The energy consumption for processing phase is proportional to the fixed processing time and unable to be reduced by scheduling the sequence of jobs. Therefore, the energy consumption for machine idle stage is considered. Besides, blocking time exists for BFSP when a job is blocked in a machine. Considering that the machine workload of blocking time is probably different from that of idle time, the energy consumption for blocking time is also taken into consideration. The overall energy consumption is comprised of energy consumption E_I for idle time T_I and energy consumption E_B for blocking time T_B. A natural assumption is that that energy consumption is proportional to time, so we have $E_I = wT_I$, where w is the energy consumption for each idle time unit. Let λ denote the ratio of blocking time energy consumption to idle time energy consumption for each time unit. Then we have $E_B = w\lambda T_B$. Generally, each blocking time unit causes no less energy consumption than each idle time unit, so the ratio is not lower than one ($\lambda \geq 1$). Accordingly, the second objective is obtained as

$$f_2(\pi) = E_I(\pi) + E_B(\pi) = wT_I(\pi) + w\lambda T_B(\pi) \tag{7}$$

It should be noted that blocking time can be removed by postponing some operations, which can be discussed in three cases. Case A: for the blocking situation on the first machine M_1, the blocking time is removed by postponing the corresponding operations on M_1. In other words, we can remove blocking time by adding no-wait constraint for two consecutive operations on M_1 and M_2 of a job. In this case, the increase in idle time is equal to the amount of removed blocking time, and the makespan does not change. Case B: for the blocking situation of the last job $\pi(n)$ on machine M_i ($1 < i < m$), the blocking time is removed by treating $\pi(n)$ as a job with no-wait constraint. In this case, the increase in idle time is greater than the amount of removed blocking time, and the makespan does not change. Case C: for the blocking situation of job $\pi(j)$ ($1 < j < n$) on machine M_i ($1 < i < m$), the blocking time is removed by treating $\pi(j)$ as a job with no-wait constraint. In this case, the increase in idle time is greater than the amount of removed blocking time, and the makespan is possible to increase.

When energy consumption is taken as a scheduling objective, it is necessary to consider the above three cases of removing blocking time. For cases B and C, it is difficult to determine in advance whether it is worthwhile to remove blocking time, because it is unknown whether $f_2(\pi)$ will increase before it is calculated, and, particularly for case C, it is unknown whether $f_1(\pi)$ will increase before it is calculated. For case A, the removal of blocking time on M_1 can consistently achieve a relatively lower value of energy consumption if λ is greater that one.

One way to handle the issue of removing blocking time is to consider all the three cases and evaluate all the schedules. The other way is to just consider case A and ignore cases B and C. For simplicity, we calculate the objectives of π using the latter way in this study. Accordingly, the blocking time

and idle time are calculated as

$$T_B(\pi) = \sum_{j=2}^{n} \sum_{i=2}^{m-1} \max\{d_{\pi(j-1),i+1} - (d_{\pi(j),i-1} + p_{\pi(j),i}), 0\} \tag{8}$$

$$T_I(\pi) = \sum_{i=1}^{m} d_{\pi(n),i} - \sum_{j=1}^{n} \sum_{i=1}^{m} p_{j,i} - T_B(\pi) \tag{9}$$

For a given job permutation π, the two objectives can be computed in $O(mn)$ time. Let Π denote the set of all the job permutations. Then the multi-objective BFSP is formulated as

$$\text{Minimize } \{f_1(\pi), f_2(\pi)\} \quad \text{for all } \pi \in \Pi \tag{10}$$

The practical energy consumption is related to the energy consumption for each idle time unit (w) and the ratio of blocking time energy consumption to idle time energy consumption for each time unit (λ). Considering that w and λ are probably different for different blocking flow shop environments in manufacturing industry, this study assumes that we have $w = 1$ and $\lambda = 2$ for simplicity.

Take the following instance as an example to illustrate the removal of blocking time and computation of the objectives. There are four jobs and three machines. The processing times are given as

$$(p_{j,i})_{4\times 3} = \begin{bmatrix} 1 & 4 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 3 \\ 1 & 2 & 1 \end{bmatrix},$$

and the permutation is $\pi = (1, 2, 3, 4)$.

The departure time, blocking time, and idle time are computed as follows.

$$\begin{aligned}
d_{\pi(1),0} &= 0, \\
d_{\pi(1),1} &= p_{\pi(1),1} = 1, \\
d_{\pi(1),2} &= d_{\pi(1),1} + p_{\pi(1),2} = 5, \\
d_{\pi(1),3} &= d_{\pi(1),2} + p_{\pi(1),3} = 7, \\
d_{\pi(2),0} &= d_{\pi(1),1} = 1, \\
d_{\pi(2),1} &= \max\{d_{\pi(2),0} + p_{\pi(2),1}, d_{\pi(1),2}\} = 5, \\
d_{\pi(2),2} &= \max\{d_{\pi(2),1} + p_{\pi(2),2}, d_{\pi(1),3}\} = 7, \\
d_{\pi(2),3} &= d_{\pi(2),2} + p_{\pi(2),3} = 10, \\
d_{\pi(3),0} &= d_{\pi(2),1} = 5, \\
d_{\pi(3),1} &= \max\{d_{\pi(3),0} + p_{\pi(3),1}, d_{\pi(2),2}\} = 8, \\
d_{\pi(3),2} &= \max\{d_{\pi(3),1} + p_{\pi(3),2}, d_{\pi(2),3}\} = 10, \\
d_{\pi(3),3} &= d_{\pi(3),2} + p_{\pi(3),3} = 13, \\
d_{\pi(4),0} &= d_{\pi(3),1} = 8, \\
d_{\pi(4),1} &= \max\{d_{\pi(4),0} + p_{\pi(4),1}, d_{\pi(3),2}\} = 10, \\
d_{\pi(4),2} &= \max\{d_{\pi(4),1} + p_{\pi(4),2}, d_{\pi(3),3}\} = 13, \\
d_{\pi(4),3} &= d_{\pi(4),2} + p_{\pi(4),3} = 14, \\
T_B(\pi) &= \max\{d_{\pi(1),3} - (d_{\pi(2),1} + p_{\pi(2),2}), 0\} \\
&\quad + \max\{d_{\pi(2),3} - (d_{\pi(3),1} + p_{\pi(3),2}), 0\} \\
&\quad + \max\{d_{\pi(3),3} - (d_{\pi(4),1} + p_{\pi(4),2}), 0\} = 3, \\
T_I(\pi) &= \sum_{i=1}^{3} d_{\pi(n),i} - \sum_{j=1}^{4} \sum_{i=1}^{3} p_{j,i} - T_B(\pi) = 10.
\end{aligned}$$

The objective values are obtained as follows.

$$\begin{aligned}
f_1(\pi) &= d_{\pi(4),3} = 14, \\
f_2(\pi) &= T_I(\pi) + 2T_B(\pi) = 16.
\end{aligned}$$

The Gantt chart is shown in Fig. 1, where blocking time on M_1 is removed (case A). Figure 2 shows the schedule when all jobs are processed as early as possible. When all the blocking time on M_1 and blocking time of $\pi(4)$ are removed (cases A and B), we have the schedule shown in Fig. 3. Figure 4 shows the schedule with all the blocking time removed (cases A, B, and C).

It is worth mentioning that there is a conflict between the makespan and energy consumption criteria. Assuming that a schedule π_a has a lower makespan than another schedule $\pi_b(f_1(\pi_a) < f_1(\pi_b))$. It is possible that the blocking time in

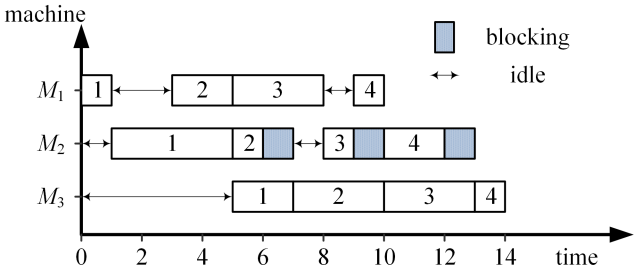

FIGURE 1. Gantt chart for $\pi = (1, 2, 3, 4)$.

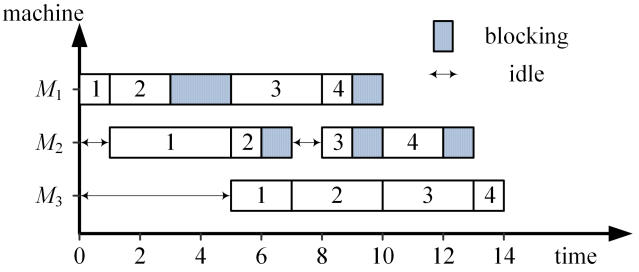

FIGURE 2. Gantt chart with jobs scheduled as early as possible ($T_B(\pi) = 6$, $T_I(\pi) = 7$, $C_{\max}(\pi) = 14$).

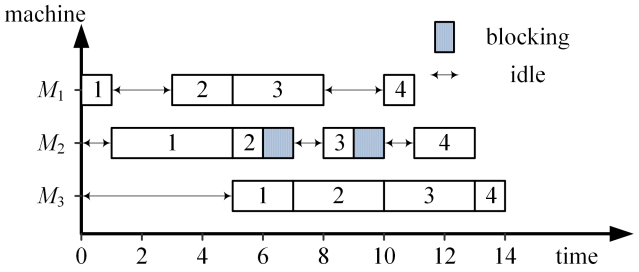

FIGURE 3. Gantt chart with blocking time on M_1 and of $\pi(4)$ removed ($T_B(\pi) = 2$, $T_I(\pi) = 12$, $C_{\max}(\pi) = 14$).

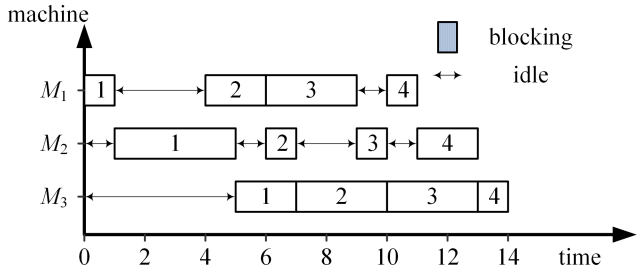

FIGURE 4. Gantt chart with all blocking time removed ($T_B(\pi) = 0$, $T_I(\pi) = 14$, $C_{max}(\pi) = 14$).

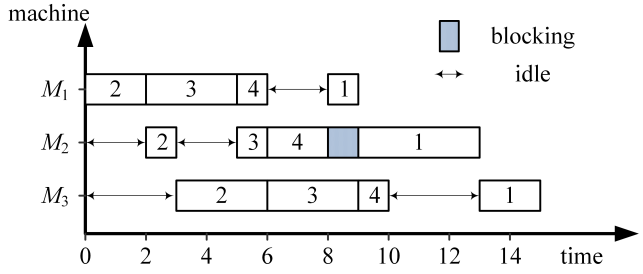

FIGURE 5. Gantt chart for $\pi' = (2, 3, 4, 1)$.

schedule π_a is more than that in schedule π_b, and hence the energy consumption for schedule π_a is greater than that for schedule π_b, namely $f_2(\pi_a) > f_2(\pi_b)$. An example can be found when $\pi' = (2, 3, 4, 1)$ for the aforementioned instance. Using the same decoding method, we have $f_1(\pi') = 15$, $T_I(\pi') = 12$, $T_B(\pi') = 1$, and $f_2(\pi') = 14$. The Gantt chart is illustrated in Fig. 5.

III. MULTI-OBJECTIVE PARALLEL VARIABLE NEIGHBORHOOD SEARCH ALGORITHM

The selection of neighborhood structures and design of local search are crucial to the performance of variable neighborhood search (VNS). Two well-known neighborhood structures, insert move and swap move, are used, and a multi-objective variable neighborhood search process that systematically changes the neighborhood is designed. To maintain a diverse evolution population, the VNS begins with *ps* multi-starts and evolves in parallel. In addition, an insertion-based pareto local search method is designed to enhance its search ability for pareto front. The initialization, variable neighborhood search process, and pareto local search method are explained as follows.

A. INITIALIZATION

The individual is represented as a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. The well-known Nawaz-Enscore-Ham (NEH) heuristic [52] is an effective heuristic designed originally for the makespan criterion. It has basically three steps. Firstly, all jobs are sorted in non-increasing order of the total processing time, and a job priority order is obtained. Secondly, the first two jobs in the order are scheduled to achieve a minimum partial objective value, and a partial sequence is obtained. Thirdly, the p-th job ($p = 3, \ldots, n$) in the order is inserted into the best position of the partial sequence, and a partial sequence with p jobs is obtained. The final permutation with all jobs scheduled is the NEH solution. In order to generate an initial population with both diversity and quality, the NEH heuristic is modified as follows. The job priority order in the first step is obtained randomly, and the objective value $f_3(\pi)$ used in the second and third steps is designed as a weighted sum of $f_1(\pi)$ and $f_2(\pi)$.

$$f_3(\pi) = \frac{k}{ps-1} f_1(\pi) + \frac{ps-k-1}{ps-1} f_2(\pi), \quad k = 0, 1, \ldots, ps-1 \tag{11}$$

The modified NEH heuristic is performed with different k values as shown in (11), and *ps* initial solutions are obtained. This initialization scheme distributes the solutions on the two-dimensional plane of $f_1(\pi)$ and $f_2(\pi)$ so as to generate them with advantages of good diversity as well as quality.

In order to facilitate the multi-objective search of the algorithm, a non-dominated solution set $NDS = \{S_1, S_2, \ldots, S_{nb}\}$ (*nb* denotes the size of the incumbent *NDS*) is stored throughout the algorithm process, and each solution in the *NDS* is marked with a flag "searched" or "unsearched". After initialization, all solutions are aggregated to generate the initial *NDS*, and all solutions in the *NDS* are marked with "unsearched".

B. VARIABLE NEIGHBORHOOD SEARCH

Two widely-used kinds of neighborhood structures, insert and swap neighborhoods, are applied in the process of variable neighborhood search (VNS). For a given job permutation, the insert neighborhood consists of solutions obtained by inserting a job into another position, while the swap neighborhood is composed of solutions obtained by swapping any two jobs. Clearly, the sizes of insert and swap neighborhoods are $(n-1)^2$ and $n(n-1)/2$, respectively.

Since there are two objectives in the considered multi-objective problem, it is necessary to determine an objective as descending direction of the VNS. Here, the descending direction (denoted by function $f(x)$) is selected randomly as $f_1(x)$ or $f_2(x)$. The VNS firstly makes d random insert moves on the solution and then searches the insert neighborhood. If a solution with a lower objective is found, then the incumbent solution is updated. The process goes on until a local optimum is found. Thereafter, the VNS searches the swap neighborhood in the same way. The VNS terminates when no better solution is found with respect to both insert and swap neighborhoods. The VNS procedure performed on solution x is shown as follows.

Step 1: select randomly $f_1(x)$ or $f_2(x)$ to be the descending direction $f(x)$. Perform d random insert moves on x. Set flag $= 0$.

Step 2: if flag $= 2$, stop the procedure; otherwise, go to step 3.

Step 3: evaluate the insert neighborhood $insertNb(x)$ and update the *NDS*. If there exists $x' \in insertNb(x)$ and

$f(x') < f(x)$, then $x = x'$, flag $= 0$, and go to step 2; otherwise, flag $=$ flag $+ 1$, go to step 4.

Step 4: if flag $= 2$, stop the procedure; otherwise, go to step 5.

Step 5: evaluate the swap neighborhood swap$Nb(x)$ and update the *NDS*. If there exists $x' \in swapNb(x)$ and $f(x') < f(x)$, then $x = x'$, flag $= 0$, and go to step 4; otherwise, flag $=$ flag $+ 1$, go to step 2.

The time of executing the VNS is greatly affected by steps 3 and 5. Since there are randomized processes in the VNS, it is difficult to determine exactly the number of times of executing steps 3 or 5. However, the best-case time complexity can be analysed for the VNS. The best case happens when steps 3 and 5 are both run once, which means that x is not updated in steps 3 and 5. According to the sizes of insert and swap neighbourhoods and time complexity of evaluating a solution, step 3 or 5 requires $O(n^2) \cdot O(mn)$ time in this case, resulting in the best-case time complexity $O(mn^3)$ for the VNS.

It is worth noting that once a solution is added to the *NDS*, it is marked with ''unsearched''. The VNS procedure searches solutions with lower objective values. Meanwhile, the non-dominated solution set is updated.

C. INSERTION-BASED PARETO LOCAL SEARCH

An insertion-based pareto local search (IPLS) is designed to enhance the algorithm's searching ability for pareto fronts. The IPLS is based on the idea of pareto domination. For a given solution x, the IPLS considers all the insert moves of a job. If a solution dominating x is found, x is updated and the insert moves of another job is considered. The IPLS terminates when no domination solution is found for all jobs. The IPLS procedure is shown as follows.

Step 1: randomly generate a job permutation $\pi_R = (\pi_R(1), \pi_R(2), \ldots, \pi_R(n))$. Let $i = 0, j = 1$.

Step 2: find the position of job $\pi_R(j)$ in x, insert $\pi_R(j)$ into all other positions in x, and obtain $n - 1$ permutations. Evaluate all the $n - 1$ permutations, aggregate them, and obtain a local non-dominated solution set *LNDS*. If there exists a solution $x' \in LNDS$ that satisfies $x' \prec x$ (x' dominates x), then $LNDS = LNDS \backslash x'$, $x = x'$, $i = 1$; otherwise, $i = i + 1$.

Step 3: update the *NDS* using the *LNDS*. If a solution is added to the *NDS*, then it is marked with ''unsearched''.

Step 4: $j = (j+1)\ \%\ n$. If $i < n$, then go to step 2; otherwise, update the *NDS* using x. If x is added to the *NDS*, then it is marked with ''searched''. Stop the procedure.

Similar to the VNS procedure, the above procedure includes randomized processes, which make it difficult to determine the number of times of executing step 2. When the best case happens, step 2 is run for n times and the procedure terminates thereafter. Therefore, the best-case time complexity of the IPLS is also obtained as $O(mn^3)$.

The IPLS is performed on a solution that belongs to the *NDS* in each iteration of the algorithm. Specifically, if there exists a solution in the *NDS* that is ''unsearched'', the IPLS is performed on this solution; otherwise, a perturbation solution is firstly obtained by d random insert moves on a randomly selected solution from the *NDS*, and then the IPLS is performed on the perturbation solution. The procedure of applying the IPLS is shown as follows.

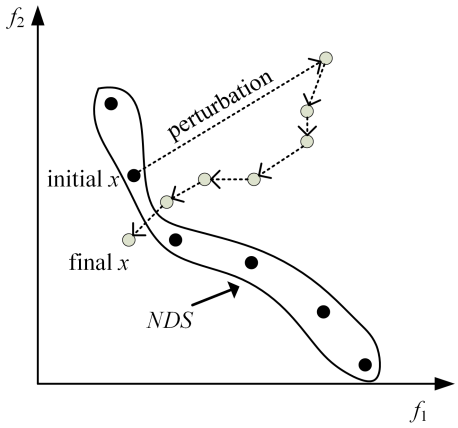

FIGURE 6. The process of applying the IPLS.

Step 1: if there exists a solution S_k in the *NDS* that is unsearched, then let $x = S_k$, and go to step 2; otherwise, randomly select a solution S_h in the *NDS*, let $x = S_h$, and go to step 3.

Step 2: perform the IPLS on x. If x returned by the IPLS is not changed, then mark the corresponding S_k in the *NDS* with ''searched''. Stop the procedure.

Step 3: make d random insert moves on x, and perform the IPLS on x. Stop the procedure.

The process of applying the IPLS is illustrated in Fig. 6.

D. PROCEDURE OF THE MPVNS

Based on the initialization scheme, process of the VNS, and strategy of applying the IPLS, the procedure of the MPVNS is shown as Fig. 7. Firstly, the modified NEH heuristic is applied to generate the initial multi-start solutions, and the non-dominated solution set *NDS* is initialized by aggregating the initial population. Then, the variable neighborhood search is performed on each solution in the population. Furthermore, the IPLS is performed on a solution selected from the *NDS* to enhance the search ability for pareto front solutions. Only two parameters, population size ps and perturbation size d, need to be calibrated in the MPVNS.

IV. COMPUTATIONAL RESULTS

This section first introduces the performance measures for evaluating the obtained non-dominated solutions. Then, the parameters of the MPVNS is tuned. Finally, the algorithm is compared with other effective metaheuristics for multi-objective scheduling. In the computational experiments, the well-known Taillard benchmark instances are used. The instances treated by the proposed algorithm are the ones with 20 to 100 jobs and 5 to 20 machines, namely Ta01-Ta90. All algorithms involved in this study are programmed in C++

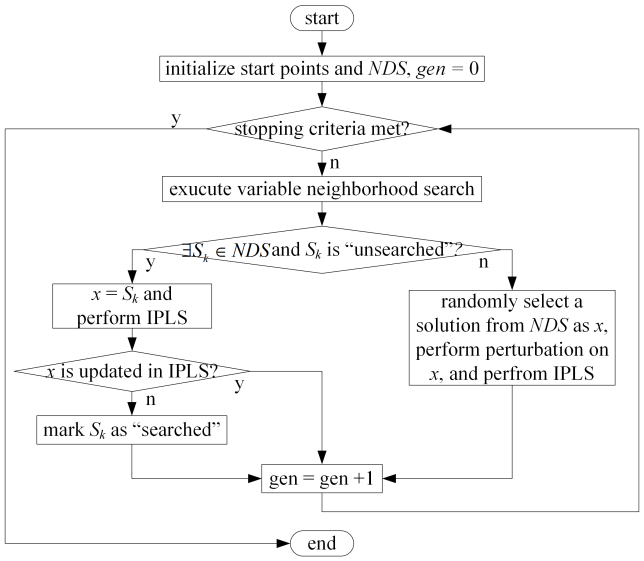

FIGURE 7. The flowchart of the MPVNS.

language, and the running environment is a PC with Intel Core (TM) i7-6700 3.4-GHz processor.

A. PERFORMANCE MEASURES

For a single-objective optimization problem, the result obtained by an algorithm is a single value. Therefore, it is relatively easy to compare the results of different algorithms. For a multi-objective optimization problem, the comparison of different algorithms is, however, more complicated, because the result of an algorithm is actually a non-dominated solution set which contains plenty of solutions. There are various performance measures applied in multi-objective optimization to compare the results of different algorithms. In this study, two measures, the set coverage [53] and hypervolume [54], are used. These measures are explained as follows.

(1) Set Coverage: let A and B be two non-dominated solution sets. The set coverage $C(A, B)$ represents the percentage of solutions in B that are dominated by at least one solution in A, which is computed as

$$C(A, B) = \frac{|\{x \in B | \exists y \in A : y \prec x\}|}{|B|} \tag{12}$$

If $C(A, B)$ is relatively great and $C(B, A)$ is relatively small, then A is superior to B to some extent.

(2) Hypervolume: suppose that A is a non-dominated solution set and $Ref = (ref_1, ref_2, \ldots, ref_r)$ is a reference point. The hypervolume of A is the volume of the hypercubes defined by all solutions in A and the reference point, which is computed as

$$Hv(A) = Leb(\bigcup_{X \in A} [f_1(X), ref_1] \times \ldots \times [f_r(X), ref_r]) \tag{13}$$

where the right side of the equation is Lebesgue measure of the hypercubes.

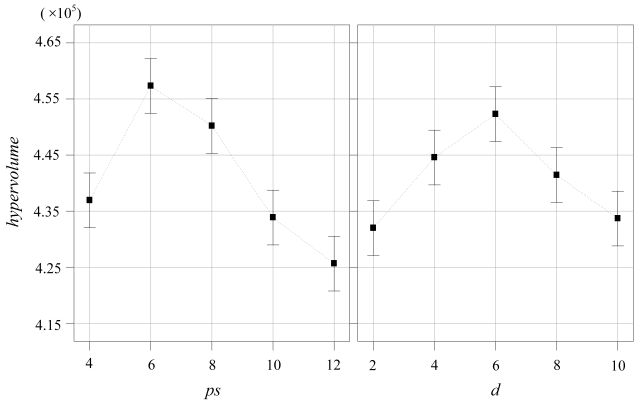

FIGURE 8. Means and 95.0% LSD intervals for tested factors.

B. ALGORITHM CALIBRATION

Before comparing the MPVNS with other algorithms from the literature, a calibration experiment was carried out. The design of experiments (DOE) technique was employed to tune the parameters of MPVNS. A full factorial experiment was conducted based on the two factors: (1) population size ps, tested at five levels: 4, 6, 8, 10, 12. (2) perturbation size d, tested at five levels: 2, 4, 6, 8 and 10. The stopping criterion of the algorithm was the elapsed CPU time not less than $50mn$ milliseconds. Three instances with different problem sizes, the instances Ta21, Ta51, and Ta81, were selected for the calibration experiment. The MPVNS was run for ten independent replicates on each instance for each combination of tested factors, and all the solution sets of ten replicates were gathered to form a non-dominated solution set. Thereafter, the hypervolume was calculated as a response variable. The reference point used herein for an instance was formed by the maximum objective values in all the non-dominated solution sets. The obtained hypervolume values were analyzed by multi-factor analysis of variance (ANOVA) technique. The ANOVA results demonstrate that the factors are statistically significant. The mean plots, together with least significant difference (LSD) with 95% confidence intervals, are illustrated in Fig. 8. Recall that if intervals for two means do not overlap, then the difference between the two means is statistically significant.

Figure 8 suggests that the population size and the perturbator size are both statistically significant. For the population size ps, values 6 and 8 are statistically better than the other values. However, the difference is small between 6 and 8. Similar results can be obtained for the parameter d. The value 6 is statistically better than the values 2, 8, and 10, but no statistical difference is found between the values 6, and 4. Based on the ANOVA results, the parameters are chosen as $ps = 6$ and $d = 6$. It is worth noting that such parameter setting is not necessarily the best choice, owing to the restricted instances and performance measure we employed in the calibration experiment.

TABLE 1. Set coverage yielded by the algorithms for instance groups with different sizes.

instance size	MPVNS (A) vs BMSA (B)		MPVNS (A) vs NSGA-II (C)	
	$C(A, B)$	$C(B, A)$	$C(A, C)$	$C(C, A)$
20× 5	**0.00**	**0.00**	**0.01**	0.00
20× 10	**0.00**	**0.00**	**0.00**	**0.00**
20× 20	0.01	**0.02**	**0.01**	**0.01**
50× 5	**0.71**	0.22	**0.89**	0.05
50× 10	**0.69**	0.18	**0.87**	0.11
50× 20	**0.58**	0.29	**0.93**	0.07
50× 5	**0.75**	0.16	**1.00**	0.00
50× 10	**0.62**	0.31	**1.00**	0.00
50× 20	**0.75**	0.21	**0.96**	0.02
average	**0.46**	0.15	**0.63**	0.03

TABLE 2. Hypervolume yielded by the algorithms for instance groups with different sizes.

instance size	hypervolume		
	MPVNS	BMSA	NSGA-II
20× 5	**10035.0**	**10035.0**	10034.8
20× 10	**31665.7**	**31665.7**	31664.5
20× 20	**153359.3**	153342.8	153219.0
50× 5	**15085.9**	12644.3	9890.9
50× 10	**79085.5**	68867.1	56602.0
50× 20	**332954.9**	313605.6	265788.1
100× 5	**32443.5**	24769.2	5841.8
100× 10	**181727.1**	164692.2	70363.6
100× 20	**787582.6**	598916.2	528266.3
average	**180437.7**	153170.9	125741.2

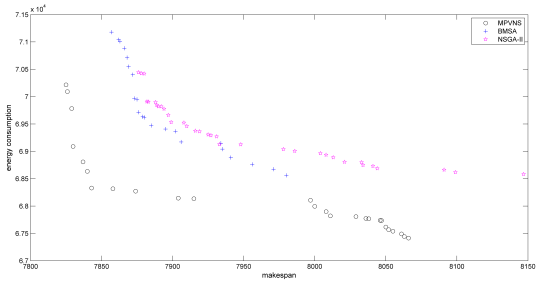

FIGURE 9. The non-dominated solutions obtained by the compared algorithms for Ta81.

C. COMPARISON WITH OTHER METAHEURISTICS

To test the performance of the proposed MPVNS, two other existing powerful meta-heuristics, non-dominated sorting genetic algorithm (NSGA-II) [55] and bi-objective multi-start simulated annealing algorithm (BMSA) [4], were re-implemented for solving the proposed model of the multi-objective blocking flow shop scheduling problem (BFSP). The NSGA-II is a well-known multi-objective optimization algorithm that has been applied to not only continuous function optimization but also various kinds of scheduling problems. The BMSA is an efficient algorithm recently developed for multi-objective scheduling in permutation flow shops, and its superiority has been validated by comparisons with several existing benchmark algorithms. In the computational experiments, all the algorithms were applied to solve each of the 90 instances for ten replicates, and the stopping criterion were all set as $50mn$ milliseconds. For the convenience of calculating performance measures of a considered instance, all the solution sets of ten replicates were gathered for each algorithm, and three instance-related non-dominated solution sets, A_1, A_2, and A_3, were obtained for the three algorithms. Thereafter, the set coverage and hypervolume values were computed based on A_1, A_2, and A_3. Note that a reference point is needed to compute hypervolume. The reference point used herein for an instance was formed by the maximum objective values in A_1, A_2, and A_3. The calculated set coverage and hypervolume values are shown in Tables 1 and 2 respectively, where the results are averaged and grouped by different instance sizes. Besides, the net non-dominated front which is formed by gathering A_1, A_2, and A_3 is listed in Appendix. Since the actual pareto front is not known, the net non-dominated front can possible serve as benchmark for future research attempts.

Table 1 compares the MPVNS with the BMSA as well as NSGA-II in terms of set coverage. It is observed that on one hand, the MPVNS yielded better set coverage values than both the BMSA and NSGA-II algorithms for the instance groups with 50 and 100 jobs. On the other hand, the differences between the MPVNS and the BMSA (or NSGA-II) is small for the instance groups with 20 jobs. Specifically, when the MPVNS (A) is compared with the BMSA (B), $C(A, B)$ is equal to $C(B, A)$ for the 20×5 and 20×10 instance groups, and $C(A, B)$ is slightly lower than $C(B, A)$ for the 20×20 instance group. When the MPVNS (A) is compared with the NSGA-II (C), $C(A, C)$ is equal to $C(C, A)$ for the 20×10 and 20×20 instance groups, and $C(A, C)$ is slightly greater than $C(C, A)$ for the 20×5 instance group. Overall, although the differences among them are small for instances with 20 jobs, the MPVNS is superior to the other two algorithms for instances with 50 and 100 jobs.

Table 2 compares the three algorithms in terms of hypervolume. It can be observed that on average the MPVNS outperforms the BMSA, and the BMSA outperforms the NSGA-II. In accordance with the results given in Table 1, the MPVNS yields better values than the BMSA and NSGA-II algorithms for the instance groups with 50 and 100 jobs. However, for the instance groups with 20 jobs, the results are more interesting. For the 20×5 and 20×10 instance groups, the hypervolume values of MPVNS (10035.0, 31665.7) are equal to those of BMSA but greater than those of NSGA-II (10034.8, 31664.5), which indicates

TABLE 3. Net non-dominated fronts found by all the algorithms for instance size 20 × 5.

Ta01		Ta02		Ta03		Ta04		Ta05	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1,374	1,815	1,408	2,086	1,280	2,237	1,448	1,723	1,341	1,977
1,377	1,790	1,411	2,082	1,281	2,234	1,450	1,705	1,344	1,890
1,379	1,787	1,417	2,054	1,286	2,233	1,456	1,696	1,345	1,874
1,380	1,738	1,424	2,050	1,290	2,194	1,457	1,685	1,348	1,873
1,385	1,651	1,427	2,015	1,292	2,182	1,459	1,672	1,351	1,872
1,427	1,645	1,435	2,013	1,293	2,110			1,353	1,864
1,442	1,636	1,436	2,003	1,295	2,102			1,354	1,860
		1,438	1,986	1,296	2,012			1,357	1,848
		1,441	1,979	1,330	1,998			1,359	1,843
		1,442	1,959	1,359	1,961			1,360	1,781
		1,448	1,938					1,384	1,756
		1,455	1,929						
		1,469	1,892						
		1,486	1,875						
		1,489	1,862						
		1,491	1,785						
Ta06		Ta07		Ta08		Ta09		Ta10	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1,363	2,269	1,381	2,256	1,379	1,723	1,373	1,805	1,283	1,800
1,367	2,252	1,383	2,126	1,381	1,626	1,378	1,771	1,286	1,798
1,368	2,233	1,385	2,083			1,380	1,729	1,289	1,784
1,369	2,123	1,408	2,065			1,383	1,713	1,291	1,770
1,373	2,092	1,410	2,063			1,411	1,711	1,292	1,748
1,377	2,070	1,413	2,061			1,412	1,701	1,293	1,746
1,382	2,049	1,422	2,052			1,416	1,681	1,296	1,734
1,391	2,035	1,427	2,013			1,418	1,649	1,301	1,710
1,406	2,011	1,428	2,002			1,482	1,620	1,333	1,707
1,407	1,942	1,429	1,969						
1,414	1,935	1,430	1,959						
1,415	1,922	1,432	1,940						
1,417	1,851	1,447	1,886						
1,418	1,841	1,451	1,873						
		1,488	1,843						

TABLE 4. Net non-dominated fronts found by all the algorithms for instance size 20 × 10.

Ta11		Ta12		Ta13		Ta14		Ta15	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1,698	6,968	1,833	7,209	1,659	6,327	1,535	6,835	1,617	7,037
1,702	6,824	1,834	7,182	1,672	6,322	1,537	6,760	1,626	6,606
1,718	6,815	1,835	7,156	1,677	6,056	1,539	6,726	1,634	6,600
1,723	6,771	1,861	7,146			1,541	6,700	1,640	6,485
1,726	6,765	1,870	7,115			1,542	6,410	1,674	6,464
1,813	6,717	1,874	7,113			1,630	6,364	1,681	6,378
		1,885	7,102			1,649	6,275	1,685	6,376
		1,895	7,031			1,653	6,222	1,696	6,344
		1,905	7,000			1,677	6,136	1,701	6,163
		1,928	6,975					1,714	6,154
Ta16		Ta17		Ta18		Ta19		Ta20	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1,590	6,753	1,622	6,567	1,731	7,195	1,747	7,252	1,782	7,597
1,596	6,712	1,630	6,418	1,740	7,120	1,751	7,203	1,786	7,556
1,597	6,600	1,632	6,388	1,741	6,838	1,752	7,182	1,788	7,530
1,601	6,558	1,670	6,378	1,755	6,804	1,755	6,994	1,790	7,404
1,604	6,520	1,673	6,288	1,779	6,758	1,758	6,966	1,796	6,919
1,619	6,507	1,682	6,272	1,794	6,714	1,759	6,955	1,797	6,806
1,632	6,473	1,710	6,151			1,765	6,889	1,799	6,767
1,640	6,461	1,725	6,086			1,766	6,848	1,822	6,660
1,642	6,457					1,779	6,846	1,840	6,650
1,646	6,429					1,785	6,746		
1,722	6,407					1,789	6,701		
						1,807	6,682		
						1,811	6,583		
						1,817	6,580		
						1,824	6,528		
						1,829	6,440		
						1,838	6,366		

that the MPVNS is equivalent to the BMSA but superior to the NSGA-II. For the 20×20 instance group, the MPVNS obtains a better hypervolume value than the BMSA as well as the NSGA-II.

To show the differences among the algorithms more clearly for the large size instances, we select instance Ta 81 and draw the non-dominated solution set of each algorithm in Fig. 9. Note that the non-dominated solution set of each algorithm is the aggregation of the results in 10 replicates. It can be seen from Fig. 9 that the non-dominated solutions obtained by the MPVNS dominate a large number of solutions obtained by the other algorithms. In other words, the MPVNS can find a non-dominated solution set that is more approximate to actual pareto front.

D. PERFORMANCE ANALYSIS

The results of computational experiments demonstrate that on average, the proposed MPVNS is superior to the BMSA, and the BMSA is superior to the NSGA-II. For small-size instances with 20 jobs, the advantages of the MPVNS are not obvious. The reasons may lie in two aspects. One is that the instances are less difficult to solve because of its small sizes. The other is that the given CPU time (50mn milliseconds) is sufficient for the algorithms to find a non-dominated solution set that is extremely approximate to the actual pareto front.

For the other instance groups, the superiority of the proposed algorithm is clearly evident. The efficiency of the proposed algorithm is due to design of the VNS and IPLS, as well as the collaborative paradigm of the algorithm. The VNS utilizes both insert and swap neighborhoods and reaches a local optimum with respect to these two neighborhoods. Meanwhile, the searching direction is randomly chosen as makespan or energy consumption. Based on the above processes, the VNS can cover a large solution space for exploration. Furthermore, the IPLS is designed to enhance the algorithm's searching ability for pareto fronts. Since it is performed on a non-dominated solution, the IPLS consistently searches the surroundings of the current non-dominated solution set, achieving a high pressure for exploitation. Last but

TABLE 5. Net non-dominated fronts found by all the algorithms for instance size 20 × 20.

Ta21		Ta22		Ta23		Ta24		Ta25	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
2,436	24,226	2,234	22,210	2,479	24,015	2,348	22,961	2,435	23,387
2,439	24,093	2,238	22,193	2,483	23,774	2,353	22,700	2,436	23,241
2,441	24,055	2,242	22,173	2,484	23,531	2,357	22,508	2,448	23,221
2,442	23,995	2,247	22,102	2,488	23,530	2,358	22,335	2,453	23,136
2,445	23,847	2,248	21,999	2,491	23,415	2,366	22,306	2,454	23,129
2,446	23,776	2,253	21,918	2,496	23,400	2,367	22,223	2,458	22,972
2,447	23,592	2,259	21,816	2,497	23,187	2,368	22,076	2,485	22,967
2,451	23,560	2,284	21,774	2,510	22,982	2,369	21,886	2,501	22,956
2,455	23,365	2,326	21,762	2,511	22,905	2,372	21,611	2,598	22,928
2,467	23,345	2,327	21,740	2,519	22,824	2,378	21,597	2,600	22,815
2,468	23,338	2,331	21,644	2,524	22,658	2,379	21,535	2,613	22,744
2,474	23,337	2,350	21,581	2,544	22,647	2,388	21,532		
2,476	23,306	2,356	21,396	2,550	22,632	2,389	21,436		
2,477	23,279	2,453	21,373	2,554	22,624				
2,479	23,256	2,465	21,272	2,557	22,614				
2,487	23,169	2,470	21,249	2,561	22,424				
2,490	23,145	2,471	21,228	2,578	22,387				
2,498	23,029	2,493	21,226	2,588	22,328				
2,508	23,005			2,609	22,297				
2,513	22,996			2,612	22,236				
2,522	22,901			2,621	22,235				
2,527	22,794			2,628	22,157				
2,531	22,786			2,631	22,145				
2,534	22,719			2,632	22,086				
2,554	22,709			2,648	21,951				
2,593	22,671			2,690	21,901				
2,595	22,522								
Ta26		Ta27		Ta28		Ta29		Ta30	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
2,383	23,505	2,390	22,437	2,328	22,753	2,363	23,035	2,323	22,188
2,388	23,029	2,396	22,323	2,344	22,736	2,372	22,903	2,324	21,809
2,398	23,007	2,398	21,824	2,345	22,521	2,379	22,782	2,327	21,468
2,399	22,404	2,404	21,784	2,352	22,486	2,395	22,710	2,328	21,411
2,407	22,206	2,405	21,779	2,354	22,349	2,398	22,696	2,329	21,346
2,415	22,183	2,410	21,710	2,356	22,273	2,400	22,689	2,338	21,287
2,419	21,963	2,413	21,675	2,366	22,038	2,406	22,679	2,345	21,208
2,440	21,830	2,417	21,647	2,368	22,007	2,409	22,620	2,360	21,178
2,441	21,730	2,420	21,477	2,380	21,967	2,410	22,518	2,373	21,049
2,443	21,670	2,431	21,457	2,383	21,927	2,414	22,502	2,378	21,018
2,452	21,667	2,437	21,310	2,389	21,904	2,415	22,497	2,413	20,988
2,453	21,542	2,452	21,295	2,409	21,853	2,419	22,489	2,423	20,812
2,467	21,539	2,461	21,243	2,412	21,665	2,425	22,451	2,430	20,798
2,477	21,529	2,482	21,205	2,424	21,619	2,431	22,283		
2,507	21,437	2,523	21,163	2,452	21,514	2,476	22,256		
2,511	21,361	2,531	21,128	2,487	21,498				
				2,489	21,356				
				2,505	21,282				
				2,507	21,254				
				2,512	21,079				
				2,555	21,050				

not least, the organic combination of the VNS and IPLS in the paradigm of the MPVNS is also beneficial to the efficiency of the proposed algorithm.

TABLE 6. Net non-dominated fronts found by all the algorithms for instance size 50 × 5.

Ta31		Ta32		Ta33		Ta34		Ta35	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
2,994	3,299	3,194	3,511	3,004	3,666	3,123	3,411	3,150	3,685
2,995	3,285	3,199	3,478	3,005	3,611	3,124	3,357	3,156	3,680
2,997	3,277	3,200	3,473	3,006	3,557	3,126	3,345	3,159	3,653
2,998	3,257	3,205	3,418	3,007	3,524	3,128	3,342	3,164	3,634
3,010	3,238	3,207	3,415	3,009	3,505	3,129	3,337	3,175	3,606
3,013	3,231	3,211	3,398	3,010	3,498	3,131	3,299		
3,017	3,207	3,215	3,388	3,012	3,479	3,136	3,270		
				3,013	3,469	3,138	3,250		
				3,016	3,465	3,164	3,247		
				3,018	3,464	3,167	3,246		
				3,020	3,446	3,178	3,241		
				3,022	3,427	3,181	3,240		
Ta36		Ta37		Ta38		Ta39		Ta40	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
3,172	3,430	3,027	3,548	3,053	3,383	2,897	3,281	3,111	3,409
3,175	3,426	3,028	3,498	3,054	3,317	2,912	3,277	3,114	3,388
3,176	3,232	3,035	3,438	3,056	3,315	2,914	3,273	3,115	3,359
3,177	3,213	3,036	3,420	3,058	3,298				
3,190	3,203	3,040	3,416	3,063	3,288				
3,191	3,184	3,042	3,406	3,068	3,282				
3,241	3,183	3,047	3,402	3,069	3,263				
		3,048	3,391	3,081	3,259				
		3,061	3,328	3,085	3,249				
		3,086	3,318						
		3,090	3,306						
		3,092	3,288						
		3,095	3,276						
		3,096	3,264						
		3,097	3,240						
		3,098	3,216						
		3,102	3,208						
		3,104	3,205						
		3,105	3,188						
		3,120	3,185						

V. CONCLUSION

This study considers the blocking flow shop scheduling problem (BFSP) for minimizing both the makespan and energy consumption objectives. Most of the existing research on the BFSP has focused on other criteria, but little has been done for the energy consumption minimization. Therefore, a multi-objective BFSP model is formulated in consideration of both makespan and machine energy consumed in blocking and idle time. In order to solve the proposed multi-objective model, a multi-objective parallel variable neighborhood search (MPVNS) algorithm is presented. The Nawaz–Enscore–Ham-based heuristic is modified and used to generate initial solutions with both diversity and quality. In each iteration of the proposed algorithm, the variable neighborhood search with respect to both insert and swap neighborhoods is designed for exploring the solutions in the population. Furthermore, an insertion-based pareto

TABLE 7. Net non-dominated fronts found by all the algorithms for instance size 50 × 10.

Ta41		Ta42		Ta43		Ta44		Ta45	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
3,645	13,350	3,491	12,523	3,466	12,901	3,665	13,366	3,632	13,089
3,647	13,088	3,498	12,458	3,472	12,760	3,666	13,244	3,636	12,618
3,653	13,053	3,504	12,449	3,475	12,737	3,672	13,179	3,638	12,577
3,655	13,037	3,508	12,335	3,479	12,731	3,675	13,149	3,639	12,526
3,660	13,025	3,531	12,316	3,481	12,713	3,678	13,019	3,640	12,504
3,661	12,989	3,537	12,261	3,487	12,632	3,680	13,007	3,642	12,483
3,663	12,975	3,545	12,253	3,493	12,544	3,682	12,962	3,644	12,456
3,667	12,942	3,569	12,224	3,494	12,415	3,687	12,933	3,645	12,380
3,668	12,923	3,607	12,080	3,504	12,327	3,693	12,928	3,650	12,339
3,670	12,915			3,525	12,284	3,696	12,859	3,652	12,276
3,674	12,861					3,697	12,812	3,655	12,248
3,676	12,854							3,656	12,223
3,684	12,829							3,657	12,190
3,687	12,782							3,658	12,164
3,709	12,758							3,661	12,163
3,711	12,698							3,665	12,132
3,725	12,663							3,671	12,070
3,726	12,657							3,689	12,059
3,763	12,495								
Ta46		Ta47		Ta48		Ta49		Ta50	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
3,598	12,890	3,698	13,609	3,573	13,241	3,524	12,874	3,630	13,065
3,599	12,488	3,701	13,499	3,583	13,078	3,528	12,869	3,633	13,060
3,639	12,297	3,703	13,483	3,586	12,812	3,538	12,819	3,635	12,820
3,677	12,294	3,712	13,434	3,602	12,726	3,542	12,814	3,637	12,808
3,712	12,259	3,714	13,418	3,604	12,585	3,543	12,806	3,638	12,731
		3,717	13,338	3,610	12,447	3,545	12,777	3,639	12,568
		3,718	13,329	3,629	12,433	3,553	12,736	3,641	12,562
		3,721	13,316	3,646	12,428	3,555	12,669	3,643	12,460
		3,727	13,307	3,659	12,414	3,569	12,596	3,644	12,407
		3,733	13,296	3,661	12,413	3,579	12,588	3,646	12,405
		3,742	13,282	3,676	12,412	3,581	12,567	3,649	12,334
		3,748	13,173			3,592	12,561	3,652	12,204
		3,789	13,098			3,624	12,522	3,655	12,151
		3,802	13,069					3,666	12,118
		3,807	13,013					3,674	12,100
		3,820	12,984					3,693	12,099
								3,700	12,085
								3,701	12,042
								3,703	12,022
								3,711	12,004

local search method is developed to enhance the exploitation of the algorithm. A large computational campaign is conducted based on well-known benchmark instances. Firstly, the MPVNS is calibrated based on statistical analysis. Thereafter, the non-dominated sorting genetic algorithm (NSGA-II) and bi-objective multi-start simulated annealing algorithm (BMSA) are employed for comparison with the proposed algorithm. The set coverage and hypervolume are adopted as two performance measures for the comparison of

TABLE 8. Net non-dominated fronts found by all the algorithms for instance size 50 × 20.

Ta51		Ta52		Ta53		Ta54		Ta55	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
4,489	39,952	4,281	38,283	4,279	38,890	4,397	39,246	4,274	40,229
4,490	39,948	4,286	38,228	4,280	38,542	4,398	39,069	4,280	40,220
4,493	39,789	4,288	38,168	4,284	38,519	4,399	38,766	4,281	40,007
4,494	39,785	4,289	38,139	4,286	38,322	4,400	38,690	4,283	39,513
4,501	39,719	4,294	37,650	4,292	38,299	4,405	38,650	4,292	39,395
4,507	39,670	4,321	37,625	4,293	38,224	4,416	38,563	4,303	39,360
4,508	39,589	4,363	37,535	4,296	38,150	4,438	38,461	4,304	39,249
4,512	39,584	4,445	37,502	4,299	38,066	4,448	38,407	4,318	39,143
4,513	39,000	4,457	37,435	4,300	38,048			4,323	38,793
4,514	38,920	4,467	37,420	4,301	38,035			4,332	38,646
4,535	38,853	4,472	37,412	4,304	37,943			4,370	38,551
4,537	38,845	4,480	37,389	4,308	37,843			4,376	38,538
4,539	38,821			4,312	37,829			4,377	38,461
4,578	38,819			4,363	37,821			4,380	38,404
4,584	38,796			4,364	37,754			4,398	38,183
4,585	38,723			4,370	37,726			4,414	38,157
4,610	38,666			4,372	37,607			4,420	38,105
4,772	38,600							4,457	38,027
Ta56		Ta57		Ta58		Ta59		Ta60	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
4,291	39,813	4,323	40,626	4,322	41,105	4,317	39,348	4,431	41,160
4,293	39,757	4,324	39,858	4,326	40,682	4,329	39,284	4,433	41,145
4,297	39,619	4,340	39,749	4,328	40,569	4,331	39,279	4,436	40,960
4,306	39,510	4,341	39,710	4,330	40,501	4,341	39,267	4,441	40,892
4,313	39,351	4,342	39,686	4,335	40,457	4,355	39,243	4,442	40,786
4,315	39,204	4,343	39,579	4,336	40,078	4,366	39,176	4,446	40,675
4,320	39,153	4,345	39,101	4,349	40,072	4,379	39,103	4,447	40,115
4,350	38,728	4,354	39,064	4,351	39,740	4,389	39,050	4,454	40,037
4,360	38,604	4,360	39,050	4,366	39,719	4,393	39,010	4,456	39,500
4,376	38,596	4,361	39,009	4,373	39,669	4,418	39,000	4,463	39,356
4,377	38,593	4,362	38,895	4,418	39,648	4,419	38,888	4,466	39,240
4,380	38,544	4,373	38,839	4,440	39,560	4,429	38,836	4,473	38,930
4,385	38,531	4,388	38,815			4,432	38,810	4,480	38,920
		4,391	38,555			4,433	38,802	4,491	38,839
		4,394	38,553			4,441	38,740	4,508	38,812
						4,469	38,737	4,519	38,780
						4,480	38,692	4,529	38,736
						4,489	38,648	4,541	38,708
						4,507	38,641	4,557	38,647
						4,522	38,600	4,566	38,509
						4,525	38,597	4,584	38,429
						4,528	38,592		
						4,529	38,537		
						4,531	38,377		
						4,536	38,367		
						4,542	38,345		
						4,543	38,336		
						4,545	38,298		
						4,546	38,269		
						4,549	38,240		
						4,553	38,210		
						4,559	38,119		
						4,566	38,016		

TABLE 9. Net non-dominated fronts found by all the algorithms for instance size 100 × 5.

Ta61		Ta62		Ta63		Ta64		Ta65	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
6,109	5,773	5,982	5,636	5,888	5,828	5,737	5,602	5,943	5,657
6,110	5,757	6,029	5,632	5,889	5,789	5,739	5,549	5,945	5,651
6,111	5,705	6,059	5,629	5,890	5,778	5,742	5,537	5,947	5,649
6,112	5,703	6,069	5,599	5,892	5,737	5,744	5,484	5,949	5,621
6,115	5,701	6,071	5,587	5,893	5,698	5,754	5,470	5,951	5,597
6,116	5,699	6,077	5,579	5,894	5,687	5,757	5,447	5,958	5,571
6,117	5,693	6,087	5,576	5,919	5,665	5,760	5,411	5,959	5,528
6,118	5,689	6,093	5,568	5,922	5,647	5,763	5,372	5,964	5,516
6,119	5,685					5,767	5,358	5,966	5,506
6,120	5,649					5,768	5,354	5,967	5,440
6,121	5,617					5,775	5,348	5,973	5,429
6,123	5,609					5,783	5,344	5,975	5,428
6,136	5,601							5,977	5,413
Ta66		Ta67		Ta68		Ta69		Ta70	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
5,786	5,537	5,951	6,123	5,843	6,001	6,083	5,706	6,113	6,316
5,787	5,535	5,952	6,112	5,844	5,967	6,084	5,698	6,115	6,315
5,789	5,499	5,955	6,107	5,850	5,958	6,087	5,671	6,121	6,092
5,795	5,481	5,961	6,102	5,851	5,935	6,088	5,657	6,124	6,066
5,806	5,471	5,962	6,072	5,852	5,928	6,091	5,655	6,127	6,017
5,812	5,453	5,967	6,059	5,853	5,917			6,130	6,009
5,836	5,444	5,969	6,024	5,854	5,884			6,133	6,003
5,841	5,441			5,855	5,877			6,137	5,997
5,842	5,423			5,856	5,866			6,145	5,987
5,847	5,420			5,861	5,846				
				5,864	5,844				
				5,867	5,814				

TABLE 10. Net non-dominated fronts found by all the algorithms for instance size 100 × 10.

Ta71		Ta72		Ta73		Ta74		Ta75	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
7,005	23,194	6,705	23,914	6,860	24,017	7,108	23,178	6,804	23,269
7,008	22,865	6,706	23,794	6,861	23,908	7,110	23,168	6,807	23,079
7,012	22,777	6,709	23,773	6,868	23,891	7,118	23,037	6,808	23,003
7,023	22,733	6,710	23,754	6,869	23,874	7,123	22,956	6,810	22,944
7,024	22,729	6,711	23,634	6,872	23,808	7,125	22,954	6,815	22,633
7,026	22,720	6,714	23,416	6,874	23,800	7,126	22,910	6,816	22,566
7,029	22,652	6,715	23,303	6,875	23,768	7,133	22,851	6,820	22,528
7,031	22,621	6,720	23,218	6,877	23,734	7,136	22,850	6,821	22,508
7,033	22,559	6,725	23,212	6,879	23,703	7,138	22,770	6,822	22,450
7,037	22,483	6,737	23,196	6,880	23,694			6,846	22,372
7,041	22,438	6,739	23,100	6,882	23,609			6,853	22,314
7,044	22,404	6,742	23,092	6,884	23,601			6,855	22,302
7,051	22,355	6,767	23,091	6,887	23,582			6,863	22,297
7,058	22,319	6,773	23,046	6,890	23,510			6,866	22,263
7,094	22,248	6,775	23,042	6,891	23,413			6,867	22,242
7,101	22,199	6,776	23,036	6,894	23,358				
		6,778	23,032	6,899	23,319				
		6,779	23,025	6,903	23,300				
				6,906	23,262				
				6,916	23,210				
				6,941	23,199				
				6,953	23,166				
				6,964	23,145				
				6,971	23,105				
				6,988	23,087				
				6,991	23,075				
				6,992	23,029				
				7,006	23,002				
				7,013	22,962				
				7,030	22,951				
				7,033	22,933				
				7,034	22,887				
Ta76		Ta77		Ta78		Ta79		Ta80	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
6,655	25,055	6,757	23,028	6,843	24,005	7,041	23,919	6,917	24,015
6,657	24,402	6,758	22,939	6,846	23,913	7,042	23,591	6,918	23,991
6,704	24,341	6,768	22,920	6,848	23,197	7,044	23,535	6,919	23,985
6,715	24,336	6,772	22,829	6,852	23,163	7,045	23,479	6,922	23,963
6,716	24,323	6,807	22,812	6,859	23,068	7,059	23,472	6,924	23,952
6,719	24,301	6,816	22,761	6,865	23,037	7,073	23,442	6,926	23,838
6,727	24,239			6,866	22,973	7,080	23,437	6,927	23,805
6,739	24,231			6,869	22,932	7,084	23,422	6,930	23,743
6,759	24,206			6,870	22,871	7,085	23,375	6,931	23,719
6,771	24,198			6,876	22,867	7,088	23,335	6,939	23,682
				6,879	22,836	7,093	23,262	6,953	23,593
				6,881	22,686	7,097	23,251	6,959	23,564
				6,886	22,609	7,098	23,196	6,960	23,549
				6,890	22,603	7,100	23,193	6,963	23,431
				6,892	22,537	7,105	23,174	6,973	23,316
				6,895	22,526	7,109	23,005	6,986	23,074
				6,906	22,507	7,179	22,989	6,992	23,055
				6,909	22,496			6,995	23,020
								7,001	23,001
								7,006	22,993
								7,012	22,974

the considered multi-objective algorithms. The experimental results show that although the advantage of the proposed algorithm is not obvious for small size instances, the proposed algorithm clearly outperforms the BMSA as well as NSGA-II algorithms in terms of the set coverage and hypervolume measures.

It is worth mentioning that the multi-objective BFSP model is a preliminary attempt for energy consumption. When applied to practical BFSP environments, it has to be adjusted to meet the real situation. Owing to its structural simplicity and good performance, the proposed MPVNS is promising for other multi-objective flow shop scheduling problems. In the future, we will focus on developing the multi-objective model and adapting the MPVNS for other flow shop environments, such as the multi-objective BFSP with more than two objectives, and the multi-objective scheduling in flexible flow shops.

APPENDIX

The net non-dominated fronts found by all the algorithms for all the instances are listed in Table 3–11.

TABLE 11. Net non-dominated fronts found by all the algorithms for instance size 100 × 20.

Ta81		Ta82		Ta83		Ta84		Ta85	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
7,825	70,216	7,860	70,747	7,879	70,902	7,841	73,028	7,864	72,091
7,826	70,092	7,862	70,664	7,881	70,783	7,844	72,962	7,865	69,231
7,829	69,783	7,864	70,361	7,882	70,364	7,845	72,650	7,878	69,224
7,830	69,090	7,868	70,171	7,885	70,352	7,849	72,356	7,883	69,193
7,837	68,808	7,876	70,089	7,886	70,261	7,850	72,218	7,885	69,085
7,840	68,634	7,880	69,818	7,888	70,206	7,852	72,125	7,901	69,018
7,843	68,330	7,884	69,789	7,892	69,744	7,857	72,047	7,902	68,865
7,858	68,316	7,888	69,747	7,908	69,641	7,858	71,909	7,906	68,748
7,874	68,272	7,891	69,726	7,911	69,635	7,861	71,502	7,913	68,680
7,904	68,145	7,895	69,707	7,913	69,541	7,862	70,800	7,915	68,556
7,915	68,135	7,896	69,560	7,928	69,390	7,867	70,702	7,918	68,501
7,997	68,106	7,923	69,489			7,874	70,618	7,920	68,477
8,000	67,995	7,936	69,382			7,877	70,555	7,921	68,395
8,008	67,898	7,951	69,332			7,885	70,498	7,922	68,343
8,011	67,820	7,957	69,331			7,886	70,100	7,923	68,229
8,029	67,807	7,981	69,282			7,893	70,045	7,928	68,161
8,036	67,774	8,012	69,259			7,895	69,997		
8,038	67,768	8,013	69,142			7,902	69,914		
8,046	67,739	8,014	69,126			7,914	69,861		
8,047	67,736	8,016	69,111			7,925	69,816		
8,050	67,617	8,017	69,100			7,931	69,802		
8,052	67,569	8,021	68,887						
8,055	67,540	8,024	68,882						
8,061	67,491	8,030	68,849						
8,063	67,443								
8,066	67,414								

Ta86		Ta87		Ta88		Ta89		Ta90	
f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
7,867	70,953	7,966	73,963	8,011	73,103	7,923	71,422	7,954	72,582
7,870	70,691	7,974	73,877	8,012	72,783	7,924	71,176	7,955	72,457
7,872	70,559	7,975	73,842	8,013	72,550	7,925	70,334	7,957	71,948
7,874	70,394	7,976	73,834	8,017	72,450	7,927	69,876	7,960	71,825
7,883	70,289	7,977	73,277	8,018	72,204	7,934	69,873	7,964	71,745
7,887	70,219	7,983	73,228	8,020	71,924	7,936	69,658	7,966	71,203
7,890	69,908	7,984	72,964	8,024	71,922	7,941	69,475	7,969	70,921
7,894	69,642	7,986	72,947	8,028	71,469	7,943	69,419	7,980	70,809
7,897	69,431	7,992	72,748	8,031	71,337	7,944	69,411	7,981	70,779
7,902	69,424	7,998	72,746	8,033	71,136	7,945	69,297	7,982	70,601
7,904	69,281	8,003	72,710	8,043	70,896	7,946	69,216	7,987	70,555
7,909	69,274	8,011	72,544	8,050	70,878	7,949	69,113	7,988	70,036
7,981	69,117	8,015	72,281	8,057	70,820	7,950	69,028	7,989	69,986
8,002	68,995	8,016	72,270	8,062	70,779	7,952	68,833	7,991	69,897
8,039	68,631	8,020	72,167	8,064	70,757	7,953	68,765	7,993	69,806
8,047	68,460	8,024	71,945	8,065	70,675	7,957	68,693	7,994	69,703
8,054	68,373	8,029	71,932	8,070	70,634	7,959	68,682	7,997	69,652
8,087	68,370	8,049	71,897	8,072	70,612	7,961	68,665	8,001	69,525
8,123	68,358	8,050	71,863	8,073	70,495	7,962	68,612	8,006	69,513
8,125	68,320	8,059	71,855	8,078	70,324	7,968	68,600	8,021	69,471
		8,113	71,835	8,080	70,272	7,969	68,326		
				8,085	70,197	7,990	68,233		
				8,092	69,935	7,993	68,164		
				8,102	69,860	7,995	68,128		
				8,112	69,765	8,003	68,050		
				8,115	69,743	8,005	67,995		
				8,126	69,690	8,031	67,985		
				8,128	69,584	8,058	67,971		
				8,138	69,580	8,138	67,932		
				8,146	69,533	8,148	67,818		
				8,158	69,528				
				8,160	69,500				
				8,164	69,490				
				8,169	69,317				
				8,170	69,282				
				8,184	69,267				
				8,189	69,218				
				8,204	69,178				
				8,221	69,143				
				8,226	69,127				
				8,229	69,033				

REFERENCES

[1] H. Hoogeveen, "Multicriteria scheduling," *Eur. J. Oper. Res.*, vol. 167, no. 3, pp. 592–623, 2005.

[2] T. Loukil, J. Teghem, and D. Tuyttens, "Solving multi-objective production scheduling problems using metaheuristics," *Eur. J. Oper. Res.*, vol. 161, no. 1, pp. 42–61, 2005.

[3] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

[4] S. W. Lin and K. C. Ying, "Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm," *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1625–1647, 2013.

[5] V. A. Armentano and J. C. Arroyo, "An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem," *J. Heuristics*, vol. 10, no. 5, pp. 463–481, 2004.

[6] 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.

[7] M. Frosolini, M. Braglia, and F. A. Zammori, "A modified harmony search algorithm for the multi-objective flow shop scheduling problem with due dates," *Int. J. Prod. Res.*, vol. 49, no. 20, pp. 5957–5985, 2011.

[8] J. M. Framinan, "A fitness-based weighting mechanism for multicriteria flow-shop scheduling using genetic algorithms," *Int. J. Adv. Manuf. Tech.*, vol. 43, no. 9, pp. 939–948, 2009.

[9] B. Yagmahan and M. M. Yenisey, "A multi-objective ant colony system algorithm for flow shop scheduling problem," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1361–1368, 2010.

[10] D. Y. Sha and H. H. Lin, "A particle swarm optimization for multi-objective flowshop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 45, nos. 7–8, pp. 749–758, 2009.

[11] B. Qian, L. Wang, R. Hu, W. L. Wang, D. X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *Int. J. Adv. Manuf. Tech.*, vol. 38, no. 7, pp. 757–777, 2008.

[12] A. R. Rahimi-Vahed, M. Dangchi, H. Rafiei, and E. Salimi, "A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem," *Int. J. Adv. Manuf. Tech.*, vol. 41, no. 11, pp. 1227–1239, 2009.

[13] G. Minella, R. Ruiz, and M. Ciavotta, "A review and evaluation of multi-objective algorithms for the flowshop scheduling problem," *INFORMS J. Comput.*, vol. 20, no. 3, pp. 451–471, 2008.

[14] M. M. Yenisey and B. Yagmahan, "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends," *OMEGA*, vol. 45, no. 2, pp. 119–135, 2014.

[15] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.

[16] S. Martinez, S. D. Peres, C. Gueret, 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, 2006.

[17] 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.

[18] 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, 1996.

[19] R. Companys and M. Mateo, "Different behaviour of a double branch-and-bound algorithm on Fm/prmu/C_{max} and Fm/block/C_{max} problems," *Comput. Oper. Res.*, vol. 34, no. 4, pp. 938–953, 2007.

[20] J. Bautista, A. Cano, R. Companys, and I. Ribas, "Solving the $Fm/block/C_{max}$ problem using bounded dynamic programming," *Eng. Appl. Artif. Intel.*, vol. 25, no. 6, pp. 1235–1245, 2012.

[21] S. T. McCormick, M. L. Pinedo, S. Shenker, and B. Wolf, "Sequencing in an assembly line with blocking to minimize cycle time," *Oper. Res.*, vol. 37, no. 6, pp. 925–936, 1989.

[22] R. Leisten, "Flowshop sequencing problems with limited buffer storage," *Int. J. Prod. Res.*, vol. 28, no. 11, pp. 2085–2100, 1990.

[23] 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.

[24] 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.

[25] J. Grabowski and J. Pempera, "The permutation flowshop problem with blocking: A tabu search approach," *OMEGA*, vol. 35, no. 3, pp. 302–311, 2007.
[26] 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.
[27] 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.
[28] 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, 2011.
[29] G. Deng, Z. Xu, and X. Gu, "A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling," *Chin. J. Chem. Eng.*, vol. 20, no. 6, pp. 1067–1073, 2012.
[30] Y. Han, Q. Pan, J. Li, and H. Sang, "An improved artificial bee colony algorithm for the blocking flow shop scheduling problem," *Int. J. Adv. Manuf. Tech.*, vol. 60, nos. 9–12, pp. 1149–1159, 2012.
[31] Y. Y. Han, D. Gong, and X. Sun, "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking," *Eng. Optim.*, vol. 47, no. 7, pp. 927–946, 2015.
[32] Y. Han, J. Liang, Q. Pan, and J. Li, "Effective hybrid discrete artificial bee colony algorithms for the total flow time minimization in the blocking flow shop problem," *Int. J. Adv. Manuf. Tech.*, vol. 67, nos. 1–4, pp. 397–414, 2013.
[33] Y. Han, D. Gong, Y. Jin, and Q. Pan, "Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2771213.
[34] F. Shrouf, J. Ordieres-Meré, and A. García-Sánchez, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *J. Cleaner Prod.*, vol. 67, pp. 197–207, Mar. 2014.
[35] C. H. Liu, "Approximate trade-off between minimisation of total weighted tardiness and minimisation of carbon dioxide (CO_2) emissions in bi-criteria batch scheduling problem," *Int. J. Comput. Integr. Manuf.*, vol. 27, no. 8, pp. 759–771, 2014.
[36] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *J. Cleaner Prod.*, vol. 112, pp. 3361–3375, Jan. 2016.
[37] M. Dai, D. B. Tang, A. Giret, M. A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 5, pp. 418–429, 2013.
[38] H. Luo, B. Du, G. Q. Huang, H. Chen, and X. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," *Int. J. Prod. Econ.*, vol. 146, no. 2, pp. 423–439, 2013.
[39] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 758–771, 2016.
[40] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *J. Cleaner Prod.*, vol. 144, pp. 228–238, Feb. 2017.
[41] C. Gahm, F. Denz, M. Dirr, and A. Tuma, "Energy-efficient scheduling in manufacturing companies: A review and research framework," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 744–757, 2016.
[42] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: Methods and applications," *Ann. Oper. Res.*, vol. 175, no. 1, pp. 367–407, 2010.
[43] Z. Zheng and J. Li, "Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption," *Energy Buildings*, vol. 161, pp. 80–88, Feb. 2018.
[44] J. Li, H. Sang, Y. Han, C. Wang, and K. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Clean. Prod.*, vol. 181, pp. 584–598, Apr. 2018.
[45] X. Wang and L. Tang, "A population-based variable neighborhood search for the single machine total weighted tardiness problem," *Comput. Oper. Res.*, vol. 36, no. 6, pp. 2105–2110, 2009.
[46] J. Behnamian, M. Zandieh, and S. M. T. F. Ghomi, "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm," *Expert Syst. Appl.*, vol. 36, no. 6, pp. 9637–9644, 2009.
[47] J. Behnamian, S. M. T. F. Ghomi, and M. Zandieh, "A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic," *Expert Syst. Appl.*, vol. 36, no. 8, pp. 11057–11069, 2009.
[48] V. Roshanaei, B. Naderi, F. Jolai, and M. Khalili, "A variable neighborhood search for job shop scheduling with set-up times to minimize makespan," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 654–661, 2009.
[49] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[50] X. Cai, Z. Mei, and Z. Fan, "A decomposition-based many-objective evolutionary algorithm with two types of adjustments for direction vectors," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 1–14, Aug. 2007.
[51] J. Bai and H. Liu, "Multi-objective artificial bee algorithm based on decomposition by PBI method," *Appl. Intell.*, vol. 45, no. 4, pp. 976–991, 2016.
[52] M. Nawaz, E. E. Enscore, 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.
[53] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[54] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. D. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.
[55] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

FUCAI WANG was born in Yantai, China, in 1966. He received the B.S. degree in automation from Tsinghua University, Beijing, China, in 1988. Since 2005, he has been an Associate Professor with the School of Information and Electrical Engineering, Ludong University, Yantai. His research interests include production planning and scheduling, modeling and optimization for industry processes, and industrial automation.

GUANLONG DENG was born in Chenzhou, China, in 1985. He received the B.S. degree in automation and the Ph.D. degree in control theory and control engineering from the East China University of Science and Technology, Shanghai, China, in 2008 and 2012, respectively.

Since 2018, he has been an Associate Professor with the School of Information and Electrical Engineering, Ludong University, Yantai, China. His recent publications have appeared in some peer-reviewed journals, such as *Computers and Operations Research*, the *International Journal of Systems Science*, the *International Journal of Computer Integrated Manufacturing*, the *Chinese Journal of Chemical Engineering*, and *Mathematical Problems in Engineering*. His research interests include production scheduling and intelligent algorithms.

TIANHUA JIANG was born in Weihai, China, in 1983. He received the B.S. degree in automation from Jinan University, Jinan, China, in 2007, the M.S. degree in control science and engineering from the Sichuan University of Science and Engineering, Zigong, China, in 2010, and the Ph.D. degree with the Key Laboratory of Measurement and Control of Complex Systems of Engineering, School of Automation, Ministry of Education, Southeast University, China, in 2015.

Since 2015, he has been a Lecturer with the School of Transportation, Ludong University, Yantai, China. His recent publications have appeared in some peer-reviewed journals, such as the *Journal of Intelligent and Fuzzy Systems*, the *International Journal of Industrial Engineering: Theory, Applications and Practice*, *Proceedings of the Institution of Mechanical Engineers, Part B*: *Journal of Engineering Manufacture*, and the IEEE Access. His research interests include production scheduling and intelligence algorithm.

SHUNING ZHANG was born in Jining, China, in 1983. He received the B.S. degree in marine engineering from Yantai University, Yantai, China, in 2006, and the M.S. and the Ph.D. degrees in control theory and control engineering from Northeastern University, Shenyang, China, in 2009 and 2014, respectively.

Since 2014, he has been a Lecturer with the School of Information and Electrical Engineering, Ludong University, Yantai, China. His recent publications have appeared in some peer-reviewed journals, such as *Mathematical Problems in Engineering*, *Advances in Mechanical Engineering*, and *Applied Thermal Engineering*. His research interests include modeling, and control and optimization in complex industry process.