IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 29, 2020, accepted October 17, 2020, date of publication October 20, 2020, date of current version October 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032548

MILP Modeling and Optimization of Energy-Efficient Distributed Flexible Job Shop Scheduling Problem

LEILEI MENG[1], YAPING REN[2], BIAO ZHANG[1], JUN-QING LI[1], HONGYAN SANG[1], AND CHAOYONG ZHANG[3], (Member, IEEE)

[1]School of Computer Science, Liaocheng University, Liaocheng 252059, China
[2]School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China
[3]State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Leilei Meng (mengleilei@lcu-cs.com)

This work was supported in part by the Funds for the Basic and Applied Basic Research Foundation of Guangdong Province of China under Grant 2019A1515110399, in part by the National Natural Science Foundation of China under Grant 51575211 and Grant 51705263, and in part by the Project of International Cooperation and Exchanges NSFC under Grant 51861165202.

ABSTRACT With the global warming problem and increasing energy cost, manufacturing firms are paying more and more attention to reducing energy consumption. This paper addresses the distributed flexible job shop scheduling problem (DFJSP) with minimizing energy consumption. To solve the problem, firstly, a novel mixed integer linear programming (MILP) model is developed to solve small-scaled problems to optimality. Due to the NP-hardness of DFJSP, we then propose an efficient hybrid shuffled frog-leaping algorithm (HSFLA) for solving DFJSP, particularly for large-sized problems. HSFLA combines the shuffled frog-leaping algorithm (SFLA) with powerful global search ability and variable neighborhood search (VNS) with good local search ability. Moreover, in HSFLA, the encoding method, the decoding method, the initialization method and the memetic evolution process are specifically designed. Finally, numerical experiments are conducted to evaluate the performance of the proposed MILP model and HFSLA.

INDEX TERMS Distributed flexible job shop scheduling problem, mixed integer linear programming, shuffled frog-leaping algorithm, variable neighborhood search, energy consumption.

I. INTRODUCTION

Recent years, with the shortage of energy resources and serious global warming, energy-efficient manufacturing is attracting more and more attention from manufactures from all around the world [1]–[6]. For reducing energy consumption, it is a straightforward way to develop energy-efficient equipments. However, it is of long cycle and needs huge capital investment. Statistics show that in real production, 80% of the energy is consumed when a machine tool stays in idle state. Therefore, scheduling is suitable for reducing idle time. Moreover, energy-efficient scheduling is without additional investment and any research cycle. Up to now, energy-efficient scheduling has proved to be a good way to reduce energy consumption, and it is attracting more and more attention from researchers [2], [3], [7]–[10].

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

Flexible job shop scheduling problem (FJSP) is a typical scheduling problem, and it is with a wide application background. Moreover, it has been proved to be a NP-hard problem. With the development of the cooperative production, the distributed scheduling is attracting more and more attention from manufactures and researchers. Moreover, distributed scheduling can help to improve production efficiency and reduce production cost [11], [12]. With regard to distributed flexible job shop scheduling problem (DFJSP), it represents for a manufacturing system that comprises several factories, and each factory is a FJSP environment. Therefore, compared with FJSP, DFJSP considers factory flexibility and is more difficult to solve. For DFJSP, three sub-problems namely factory selection, machine selection and operations sequencing should be determined [13].

Although DFJSP has attracted a large number of concerns, the existing research mainly focuses on minimizing makespan [14], [15], and there is no published work about

FJSP with considering energy consumption in multi-factory production environment. In order to make up this gap, this paper addresses the energy-efficient DFJSP (EE-DFJSP) with minimizing total energy consumption. In this paper, we firstly propose a novel mixed integer linear programming (MILP) model for solving optimal solutions of small-scale instances. Due to inherent NP hardness of DFJSP, we then propose an efficient hybrid shuffled frog-leaping algorithm (HSFLA) to find the near optimal solutions of EE-DFJSP, particularly for the large-scaled instances. Compared with the existing research, the contributions of this work can be concluded as four points:

(1) According to author, this paper is the first work that considers DFJSP with minimizing energy consumption in multi-factory production environment.

(2) A novel MILP model is formulated for solving small-scale instances to obtain optimal solutions, and it is set the comparison standard to evaluate meta-heuristic algorithms.

(3) An efficient hybrid shuffled frog-leaping algorithm (HSFLA) is developed to obtain near optimal solutions for large-sized problems. Specifically, HSFLA is designed by combining SFLA and VNS. Notably, SFLA has never been applied to solve DFJSP.

(4) An energy-efficient active decoding is specifically designed for EE-DFJSP. In the decoding method, Turn Off/On and postponing strategies are designed specifically to reduce idle energy consumption.

The remainder of this paper is organized as follows: In Section II, literature review of DFJSP is provided. In Section III, a novel MILP model is developed for EE-DFJSP. In Section IV, an efficient HSFLA is designed. Moreover, an energy-efficient decoding method that considers postponing strategy and Turn Off/On is developed. In Section V, experiments are conducted to evaluate the proposed MILP model, HSFLA and energy-efficient decoding method. Lastly, Section VI provides conclusion and future study.

II. LITERATURE REVIEW

Up to now, a lot of research has been done to solve DFJSP, and they are mainly focused on minimizing makespan. For example, in order to minimize makespan, Jia *et al.* [16] designed a modified genetic algorithm (MGA) to solve distributed job shop scheduling problem (DJSP) without considering machine flexibility. For the same problem, two mixed integer linear programming (MILP) models and six heuristics were designed by Naderi and Azab [17]. With regard to DFJSP with minimizing makespan, Chan *et al.* [18] firstly developed a GA with dominant genes (GADG) to solve it. Then, GADG was extended to solve DFJSP with machine maintenance. Moreover, the encoding scheme of GADG considered all the three sub-problems. As to DFJSP, an improved GA was developed by Giovanni and Pezzella [13]. Moreover, an incomplete decoding representation that considered only factory selection and operations sequencing was designed, and the machine selection was determined in the decoding process by using heuristic rule. By using GA, Lu *et al.* [19] also proposed an incomplete decoding representation with only a job sequence, and all the three sub-problems were decided in the decoding process by heuristic rules. Then, Wu *et al.* [15] proposed another incomplete encoding representation that only took operations sequence into consideration. Moreover, a novel GA with this encoding scheme was designed [15]. Except for GA, Ziaee [20] designed an efficient heuristic; Marzouki *et al.* [21] proposed a chemical reaction optimization (CRO) algorithm, and Wu *et al.* [22] proposed an improved differential evolution (IDE) algorithm for solving DFJSP with minimizing makespan or earliness/tardiness. Meng *et al.* [14] proposed four MILP and a constraint programming models for solving DFJSP, and several new improved solutions were obtained. Most recently, Jiang *et al.* [23] studied the energy-efficient DJSP with minimizing both makespan and energy consumption, and proposed an effective modified multi-objective evolutionary algorithm with decomposition (MMOEA/D).

Above all, as to methods for solving DFJSP problem, there are mainly two categories namely exact method and approximation method. The MILP model belongs to the category of exact method, and it could obtain optimal solutions of small-sized instances. Optimal solutions are the comparison standards for designing approximation methods [14]. Therefore, MILP model is very important and meaningful for a scheduling problem, particularly for the new one. However, MILP model is not suitable for large-sized problems due to its solution pressure [24], [25]. The heuristics and meta-heuristic algorithms belong to the category of approximation method. Heuristics are often designed according to characteristics of scheduling problems and efficient. However, they cannot guarantee the quality of the obtained solution. Now, meta-heuristic algorithms are the most popular algorithms for scheduling problems, particularly for large-sized problems [26], [27]. Its performance is decided by many elements such as the workflow of the algorithm, the encoding scheme, the decoding scheme, and the operators for generating new solutions.

Therefore, for a new scheduling problem, for example the DFJSP with minimizing energy consumption in this paper, the design of both MILP models and meta-heuristic algorithms is important and meaningful. The MILP model solves optimal solutions for small-sized instances and the meta-heuristic algorithm is used for obtain near-optimal solutions for large-sized instances.

III. MILP MODEL FOR ENERGY-EFFICIENT DFJSP

A. PARAMETERS DEFINITION

The parameters definition is given as follows:

i, i'	indices for jobs
n	total number of jobs
I	set for jobs and $I = \{1, 2, \cdots, n\}$
j, j'	indices for operations of jobs

n_i	number of operations of job i
$n_{\max}$	maximum number of operations of all jobs
J_i	set for the operations of job i and $J_i = \{1, 2, \cdots, n_i\}$
k, k'	indices for machines
$O_{i,j}$	the j-th operation of job i
f	index for factory
n_f	number of factory
F	set of factories and $F = \{1, \ldots, n_f\}$
m_f	number of machines in factory f
$m_{i,j,f}$	number of machines in factory f that can process operation $O_{i,j}$
K_f	set of machines in factory f and $K_f = \{1, 2, \cdots, m_f\}$
$K_{i,j,f}$	set of machines in factory f that can process operation $O_{i,j}$
$x_{i,j,f,k}$	binary constant that takes 1if operation $O_{i,j}$ can be processed on machine k in factory f; otherwise it takes 0
$p^O_{i,j,f,k}$	processing time for operation $O_{i,j}$ processed by machine k in factory f
M	a large positive number
t	index for position
$p_{f,k}$	number of positions of machine k in factory f and $p_{f,k} = \sum_{i\in I}\sum_{j\in J_i} x_{i,j,f,k}$
$P_{f,k}$	set of positions of machine k in factory f and $P_{f,k} = \{1, 2, \cdots, p_{f,k}\}$
$P'_{f,k}$	set of top $p_{f,k} - 1$ positions for machine k in factory f and $P'_{f,k} = \{1, 2, \cdots, p_{f,k} - 1\}$
P^C_f	common power of factory f
$P^{idle}_{f,k}$	idle power of machine k in factory f
$P^O_{i,j,f,k}$	processing power of operation $O_{i,j}$ processed by machine k in factory f
$T^{off}_{f,k}$	time for Turn Off/On strategy of machine k in factory f
$T^{B,off}_{f,k}$	break-even time of machine k in factory f
$E^{off}_{f,k}$	energy consumption for Turn Off/On strategy of machine k in factory f
$N^{off}_{f,k}$	allowable times of Turn Off/On strategy for machine k in factory f
$I^E_{f,k,t}$	idle energy consumption between position t and $t+1$ of machine k in factory f
T^{PE}	total processing energy consumption
T^{IE}	total idle energy consumption
T^{CE}	total common energy consumption
T^E	total energy consumption

B. PROBLEM DEFINITION

As to EE-DFJSP, we formulate it as follows: there are a number of jobs to be processed in a number of factories. Each factory stands for a FJSP environment. Each job includes several operations with its own processing route and can be processed in only one factory. For each operation, it can only be processed by one machine in one factory. Besides, all the operations of the same job must be processed in the same factory. In order to reduce idle energy consumption of machine tools, Turn Off/On strategy is considered. In all, four sub-problems must be solved in EE-DFJSP: (1) determining the factory for each job (factory selection problem), (2) determining the machine selection of each job (machine selection problem), (3) determining the operations sequence on each machine (operations sequencing problem), and (4) determining whether the Turn Off/On strategy is implemented in an idle period of a machine or not (Turn Off/On strategy decision problem).

Moreover, the following assumptions are considered:

- All the factories, jobs, and machines are available at time zero.
- For each machine, it can at most process one operation at the same time. For each operation, interruption is not allowed.
- All the processing data such as processing time and processing power is deterministic.
- Each operation can only be assigned to one machine.
- Transportation time, setup time, energy consumption of transporters and setup energy consumption are not considered.

The objective in this paper is to minimize total energy consumption by designing methods to solve four sub-problems, namely, factory selection, machine selection, operations sequencing and Turn Off/On strategy decision.

C. ENERGY CONSUMPTION OF THE WORKSHOP

Total energy consumption of the workshop mainly includes three parts. Thereinto, the first part is processing energy consumption, which is the energy consumed by the machine when it is in processing state. The second part is idle energy consumption, and it is the energy consumed by the machine in idle state. The last part is common energy consumption, and it is the energy consumed by auxiliary equipments such as lighting and air conditioning in the workshop. The following sections will describe and model these three parts in detail.

1) PROCESSING ENERGY CONSUMPTION

Total processing energy consumption can be calculated as below,

$$T^{PE} = \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{t\in P_{f,k}} P^O_{i,j,f,k} p^O_{i,j,f,k} X_{i,j,f,k,t} \quad (1)$$

where, $X_{i,j,f,k,t}$ is binary decision variable, and it takes 1 if operation $O_{i,j}$ is assigned to position t of machine k in factory f; otherwise, it takes 0. $X_{i,j,f,k,t}$ is used to determine both machine selection problem and operations sequencing problem. Moreover, $X_{i,j,f,k,t}$ is based on Wagner's modeling idea [26], [27]. This modeling idea defines the "position" of machine. In detail, each machine is divided into several positions according to time sequence, and each position can process one operation at the same time.

2) IDLE ENERGY CONSUMPTION

Total idle energy consumption can be computed according to Eq.(2),

$$T^{IE} = = \sum_{f\in F}\sum_{k\in K_f}\sum_{t\in P'_{f,k}} I^E_{f,k,t} = \sum_{f\in F}\sum_{k\in K_f}\sum_{t\in P'_{f,k}} P^{idle}_{f,k}(S_{f,k,t+1} - F_{f,k,t}) \quad (2)$$

where, $S_{f,k,t}$ is continuous decision variable and defines the starting time of position t of machine k in factory f. $F_{f,k,t}$ represents for the completion time of position t of machine k in factory f.

In real production, a machine may run in idle state for a long time, and a lot of energy will be wasted. This may be because operations are blocked in the upstream machine tools, and the downstream machines will be in starving (idle) state for waiting the jobs' arrival. If the machine is allowed to be turned off for some time and then be switched on, a lot of energy will be saved. The energy consumed by the machine in the idle period will be reduced to the energy consumption for a Turn Off/On operation. Figure 1 shows the power curve of a machine with and without Turn Off/On operation. To implement the Turn Off/On strategy to machine k of factory f, two conditions needed to be satisfied: (1) The length of the idle period is no shorter than $T^{off}_{f,k}$. (2) Energy consumed in the idle period is no less than the energy consumption for Turn Off/On operation $E^{off}_{f,k}$. Therefore, we define constant break-even time $T^{B-off}_{f,k}$. If the idle period is longer than $T^{B-off}_{f,k}$, Turn Off/On is economically justifiable instead of letting the machine run in idle state. $T^{B-off}_{f,k}$ is computed according to Eq.(3),

$$T^{B-off}_{f,k} = \max\{T^{off}_{f,k}, E^{off}_{f,k}/P^{idle}_{f,k}\}, \quad \forall f \in F, k \in K_f \quad (3)$$

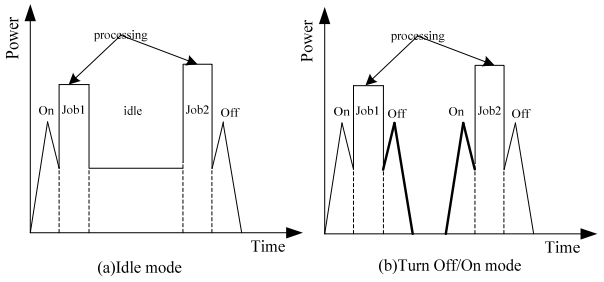

FIGURE 1. The power curve of a machine tool.

In order to determine whether Turn Off/On strategy is implemented or not (Turn Off/On strategy decision), binary decision variable $Z_{f,k,t}$ is introduced. To be specific, if a Turn off/ On strategy is applied between position t and $t+1$ of machine k in factory f, $Z_{f,k,t} = 1$; otherwise, $Z_{f,k,t} = 0$. Therefore, when the Turn Off/On is considered, $I^E_{f,k,t}$ and T^{IE} can be computed according to Eq.(4) and Eq.(5) respectively,

$$I^E_{f,k,t} = (1 - Z_{f,k,t})(S_{f,k,t+1} - F_{f,k,t})P^{idle}_{f,k} + Z_{f,k,t}E^{off}_{f,k} \quad \forall f \in F, k \in K_f, t \in P'_{f,k} \quad (4)$$

$$T^{IE} = \sum_{f\in F}\sum_{k\in K_f}\sum_{t\in P'_{f,k}} ((1 - Z_{f,k,t})(S_{f,k,t+1} - F_{f,k,t})P^{idle}_{f,k} + Z_{f,k,t}E^{off}_{f,k}) \quad (5)$$

3) COMMON ENERGY CONSUMPTION

Total common energy consumption is determined by the makespan and common power of each factory, and it can be calculated by Eq.(6),

$$T^{CE} = \sum_{f\in F} P^C_f C_{f,\max} \quad (6)$$

4) TOTAL ENERGY CONSUMPTION

Above all, the total energy consumption that considers Turn off/On strategy can be computed by Eq.(7),

$$\begin{aligned} T^E &= T^{PE} + T^{IE} + T^{CE} \\ &= \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{t\in P_{f,k}} P^O_{i,j,f,k}p^O_{i,j,f,k}X_{i,j,f,k,t} \\ &\quad + \sum_{f\in F}\sum_{k\in K_f}\sum_{t\in P'_{f,k}} ((1 - Z_{f,k,t})(S_{f,k,t+1} - F_{f,k,t})P^{idle}_{f,k} \\ &\quad + Z_{f,k,t}E^{off}_{f,k}) + \sum_{f\in F} P^C_f C_{f,\max} \end{aligned} \quad (7)$$

D. MILP MODELING FOR EE-DFJSP

Obviously, in Eq.(7), there is one non-linear term of $(1 - Z_{f,k,t})(S_{f,k,t+1} - F_{f,k,t})$. Non-linear models are much more difficult to solve than linear ones. Owing to the existing of many local optical solutions in the feasible region of the non-convex models, it is NP-hard to solve them to optimality. Therefore, Eq.(7) should be linearized. On the basis of our previous research[1], we introduce a continuous decision variable $E_{f,k,t}$ to represent for $(1-Z_{f,k,t})(S_{f,k,t+1} - F_{f,k,t})P^{idle}_{f,k} + Z_{f,k,t}E^{off}_{f,k}$. Then, linear objective function is obtained as Eq.(8),

$$\min T^E = \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{t\in P_{f,k}} P^O_{i,j,f,k}p^O_{i,j,f,k}X_{i,j,f,k,t} + \sum_{f\in F}\sum_{k\in K_f}\sum_{t\in P'_{f,k}} E_{f,k,t} + \sum_{f\in F} P^C_f C_{f,\max} \quad (8)$$

In this function, the first to the third terms of Eq.(8) represent for total processing energy consumption, total idle energy consumption and total common energy consumption respectively.

1) DECISION VARIABLE

As seen from Eq.(9), $F_{f,k,t}$ is aggravation of $S_{f,k,t}$ and $X_{i,j,f,k,t}$, therefore, it is removed to get a more simple model.

$$F_{f,k,t} = S_{f,k,t} + \sum_{i\in I}\sum_{j\in J_i}(p^O_{i,j,f,k}X_{i,j,f,k,t}), \quad \forall f \in F, k \in K_f, t \in P_{f,k} \quad (9)$$

In all, eight decision variables, namely $X_{i,j,f,k,t}$, $Y_{i,f}$, $Z_{f,k,t}$, $B_{i,j,f}$, $S_{f,k,t}$, $C_{f,\max}$, $C_{\max}$ and $E_{f,k,t}$ are needed for the MILP

model. $Y_{i,f}$ is a binary decision, and it is used to determine factory selection problem. To be more specific, if job i is assigned to factory f, $Y_{i,f} = 1$; otherwise, $Y_{i,f} = 0$. $B_{i,j,f}$ is a continuous decision variable that defines the starting time of operation $O_{i,j}$ in factory f. $C_{f,\max}$ is a continuous decision variable that defines the makespan of factory f. $C_{\max}$ is a continuous decision variable that represents for maximum makespan of all factories, $C_{\max} = \max_{f \in F} C_{f,\max}$.

2) CONSTRAINTS

For the MILP model, the following constraint sets (10)-(25) are needed.

$$\sum_{f \in F} Y_{i,f} = 1, \quad \forall i \in I \tag{10}$$

$$Y_{i,f} = \sum_{k \in K_{i,j,f}} \sum_{t \in P_{f,k}} X_{i,j,f,k,t}, \forall i \in I, j \in J_i, f \in F \tag{11}$$

$$B_{i,j,f} + \sum_{k \in K_{i,j,f}} \sum_{t \in P_{f,k}} (p^O_{i,j,f,k} X_{i,j,f,k,t}) \le B_{i,j+1,f}, \quad \forall i \in I, j \in \{1, \ldots, n_i - 1\}, f \in F \tag{12}$$

$$\sum_{i \in I} \sum_{j \in J_i} X_{i,j,f,k,t} \ge \sum_{i' \in I} \sum_{j' \in J_{i'}} X_{i',j',f,k,t+1}, \quad \forall f \in F, k \in K_f, t \in P'_{f,k} \tag{13}$$

$$\sum_{i \in I} \sum_{j \in J_i} X_{i,j,f,k,t} \le 1, \quad \forall f \in F, k \in K_f, t \in P_{f,k} \tag{14}$$

$$S_{f,k,t+1} \ge S_{f,k,t} + \sum_{i \in I} \sum_{j \in J_i} (p^O_{i,j,f,k} X_{i,j,f,k,t}) + T^{B-off}_{f,k} Z_{f,k,t}, \quad \forall f \in F, k \in K_f, t \in P'_{f,k} \tag{15}$$

$$E_{f,k,t} \ge (S_{f,k,t+1} - S_{f,k,t} - \sum_{i \in I} \sum_{j \in J_i} (p^O_{i,j,f,k} X_{i,j,f,k,t})) P^{idle}_{f,k} - M Z_{f,k,t}, \quad \forall f \in F, k \in K_f, t \in P'_{f,k} \tag{16}$$

$$E_{f,k,t} \ge E^{off}_{f,k} Z_{f,k,t}, \quad \forall f \in F, k \in K_f, t \in P'_{f,k} \tag{17}$$

$$\sum_{t \in P'_{f,k}} Z_{f,k,t} \le N^{off}_{f,k}, \quad \forall f \in F, k \in K_f \tag{18}$$

$$S_{f,k,t} \ge B_{i,j,f} - M(1 - X_{i,j,f,k,t}), \quad \forall i \in I, j \in J_i, f \in F, k \in K_{i,j,f}, t \in P_{f,k} \tag{19}$$

$$S_{f,k,t} \le B_{i,j,f} + M(1 - X_{i,j,f,k,t}), \quad \forall i \in I, j \in J, f \in F, k \in K_{i,j,f}, t \in P_{f,k} \tag{20}$$

$$C_{f,\max} \ge B_{i,n_i,f} + \sum_{k \in K_{i,n_i,f}} \sum_{t \in P_{f,k}} (pt_{i,n_i,f,k} X_{i,n_i,f,k,t}), \quad \forall i \in I, f \in F \tag{21}$$

$$C_{\max} \ge C_{f,\max}, \quad \forall f \in F \tag{22}$$

$$B_{i,j,f} \le M Y_{i,f}, \quad \forall i \in I, j \in J_i, f \in F \tag{23}$$

$$S_{f,k,t} \ge 0, \quad \forall f \in F, k \in K_f, t \in P_{f,k} \tag{24}$$

$$B_{i,j,f} \ge 0, \quad \forall i \in I, j \in J_i, f \in F \tag{25}$$

where, constraint set (10) restricts that a job can be machined in exactly one factory. Constraint set (11) restricts that the operations of the same job must be processed on the same factory. Constraint set (12) guarantees that an operation of a job can be machined only if its preceding operation of the same job has been completed. Constraint set (13) forces that an operation must be assigned to the preceding positions only when they are occupied by other operations. Constraint set (14) indicates that at most one operation can be assigned to a position of a machine. Constraint sets (15)-(17) are the constraints for Turn Off/On strategy. More specifically, constraint set (15) assures that if a Turn Off/On strategy is implemented between two adjacent positions of a machine, the idle time is no less than the break-even of the machine. Constraint sets (16)-(17) assure the energy consumption between two adjacent positions of a machine. Constraint set (18) limits the maximum times of Turn Off/On strategy. Constraint sets (19)-(20) define the relation between $S_{f,k,t}$ and $B_{i,j,f}$. To be more specific, if operation $O_{i,j}$ is assigned to position t of machine k, then $B_{i,j,f}$ is equal to $S_{f,k,t}$. Constraint set (21) defines the makespan of each factory, and constraint (22) defines the overall makespan. Constraint set (23) shows that if an operation is not assigned to some factory, its corresponding starting in this factory is equal to 0. Finally, constraint set (24)-(25) show that the decision variables are non-negative.

IV. THE PROPOSED ALGORITHM HSFLA FOR DFJSP

SFLA was firstly proposed by Eusuff and Lansey [29] for optimizing water distribution network design. Due to its high efficiency and application simplification, up to now, SFLA has been widely utilized to solve many scheduling problems [30]–[33]. More specifically, SFLA is a population-based algorithm, which combines the benefits of the genetic-based memetic algorithm and social behavior-based particle swarm optimization. In SFLA, each solution is called a frog. The key characteristics of SFLA are that the whole population is divided into a number of parallel subsets named memeplexes with some rules, and they are shuffled after the evolution of each memeplex. Each memeplex includes a set of frogs with their own ideas, which can be influenced by the other frogs in the same memeplex. Moreover, each memeplex performs its own evolutionary process independently and frogs in different memeplexes do not influence each other. During the evolution, the worst frog in each memeplex will be improved by learning form either the best solution of the memeplex or the best solution of the whole population. After the evolution of each memeplex is finished, all the memeplexes are shuffled and combined into a new population so as to exchange information. The evolution of each memeplex and the shuffling process continue until the defined convergence criteria are satisfied. More detailed steps of SFLA refer to papers [29]–[31], [33]–[36].

A. HSFLA FOR ENERGY-EFFICIENT DFJSP

In this section, we introduce HSFLA in six parts, namely encoding scheme, energy-efficient decoding, initialization, terminate criterion, memeplex construction, evolving process of each memeplex and VNS. The following sections will describe these seven parts respectively in detail.

1) ENCODING SCHEME

In this paper, we propose to use three-string coding method, each string of which is used to represent for one sub-problem independently. As shown in Figure 2, the first string represents for the operation sequence (OS), the second string indicates the machine selection (MS) and the last string decides the factory selection (FS). The length of OS and MS strings is equal to total number of operations. The length of FS is equal to the number of jobs. For OS string, each operation is represented by its job index. The left-to-right ordering of operations in OS string represents for the operations sequencing. In MS string, each element represents for the machine index selected for the corresponding operation [37]. With regard to FS string, each element of it stands for the factory selected for its corresponding operation. It must be noted that operations of the same job corresponds to the same factory. For the example in Figure 2, there are two factories, three jobs and two machines. Job 1 and Job 3 are assigned to Factory 1, and Job 2 is allocated to Factory 2.

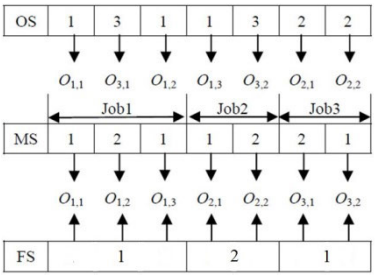

FIGURE 2. The encoding scheme of a solution.

2) ENERGY-EFFICIENT DECODING

Obviously, three sub-problems namely factory selection, machine selection and operations sequencing are decided in the encoding. It should be noted that each factory becomes as a job shop problem (JSP) with determined machine selection for each operation. In each factory, we use the active decoding method (ADM). By using ADM, an active schedule (AS) can be obtained. Moreover, in AS, there are no permissible left shifts. More detailed steps of ADM can refer to paper [38].

Figure 3 shows an example to demonstrate the processes of ADM. In Figure 3, SAS and AS represent for semi-active schedule and active schedule respectively. Compared with SAS, in AS, operation $O_{2,1}$ is inserted at the idle time period before operation $O_{1,2}$, operation $O_{3,3}$ is inserted before $O_{1,3}$, and the makespan is reduced to 13.

By using the ADM, the starting times and completion times of all operations are determined. Then, we implement the postponing strategy and Turn Off/On strategy to further reduce idle time for all machines. For different scheduling problems, the processes of postponing strategy and Turn Off/On strategy are different. Because each factory of DFJSP is a FJSP environment, the detailed processes of these two energy-saving strategies can refer to our previous research FJSP with worker flexibility [39].

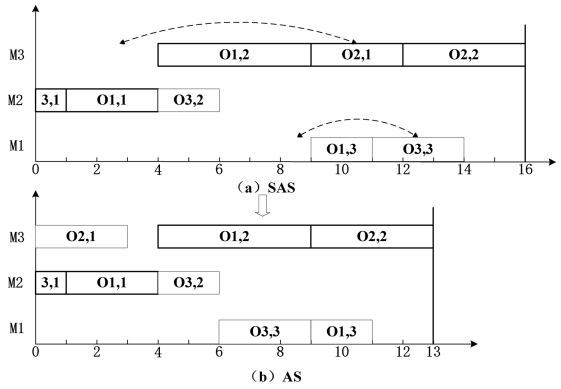

FIGURE 3. The processes of energy-efficient ADM.

3) INITIALIZATION AND TERMINATION CRITERION

Initialization method is very important for efficiency and effectiveness of meta-heuristic algorithms. In this paper, with regard to the objective of minimizing total energy consumption, we propose a greedy machine selection method (GMS) to initiate the machine selections for all operations.

The detailed steps of the initialization are given as below:

Step 1: Generate the OS and FS randomly.

Step 2: Start with the first operation of OS, select the machine with the minimum current total energy consumption of its selected factory. That is to say, operation $O_{i,j}$ will select the machine with the minimum sum of added processing energy consumption, added idle energy consumption and added common energy consumption. The detailed steps can refer to the energy-efficient decoding method proposed in our previous research for HFSP [40].

Step 3: Repeat the Step 2 until the machine of the last operation is decided.

Moreover, to keep the diversity of the initial population, we only apply the GMS for 50 percents of the population. The other 50 percents of the population are obtained randomly.

With regard to termination criterion, we set maximum running time as terminate criterion.

4) MEMEPLEX CONSTRUCTION

With regard to memeplex construction, binary tournament selection is used to divide the population into several memeplexes. The detailed steps of binary tournament selection can refer to paper [35]. By using the binary tournament selection, not all solutions are allocated into memeplexes. The solutions with better fitness have more opportunity to be passed to the child population, which makes a tradeoff between exploration and exploitation of the population.

5) EVOLUTIONARY PROCESS OF EACH MEMEPLEX

This search process is the main path to produce new solutions. In this paper, the new frogs are generated by using precedence operation cross (POX) operator and uniform cross (UC) operator. POX and UC are used for OS string and FS string respectively. These two cross operators are randomly selected with the same probability of 50%. The detailed processes of POX can refer to paper[37].

By using POX crossover, two new frogs can be obtained. Then, compare the best frog of the two with the local worst frog in the memeplex. If the generated best frog is better than the local worst frog, replace the latter with the former. Moreover, Figure 4 shows an example for POX crossover.

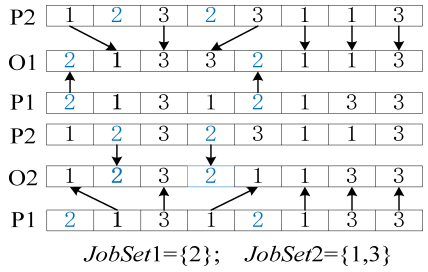

FIGURE 4. POX operator for OS.

For the FS string, UC operator is adopted. In addition, MS and FS strings are combined and crossed so as to prevent the generation of infeasible solution.

The working process of uniform crossover is given as below:

Step1: Randomly generate n binary numbers.

Step2: Generated the two offspring by swapping the elements of FS string of parent P1\P2, of which the binary numbers are equal to 1.

Step3: Copy the MS string of parent P1\P2 to offspring O1\ O2 correspondingly.

Like POX operator, by using UC operator, two new frogs can be obtained. Then, if the better frog of the two is better than the local worst frog in the memeplex, replace the latter with the former. Moreover, Figure 5 shows an example of UC operator for FS.

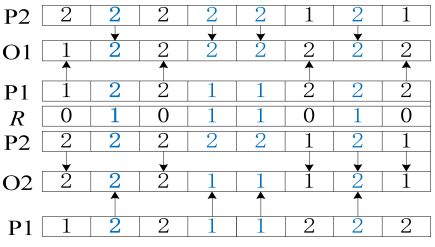

FIGURE 5. UC operator for FS.

In classical SFLA, the only communication of different memeplexes is the shuffling process. In order to increase the communication of different memeplexes [32], we cross the local best frog of each memeplex with the local best frog of another randomly selected memeplex by using POX operator or UC operator with the same possibility of 50%. If the best newly generated solution is better than the local best frog of current memeplex, replace the latter frog with the former one.

B. VARIABLE NEIGHBORHOOD SEARCH

VNS is a well-known local search method [41], and it has been widely used in many scheduling problems [39], [42]. VNS starts from one initial solution and evolves by searching solutions of a set of neighborhood structures. Therefore, VNS has good local search ability. Population-based meta-heuristic algorithms, including SFLA, have better global search ability and worse local search ability. Therefore, in this paper, we embed VNS into SFLA to improve its local search ability. Moreover, VNS is applied to the local best frog in each memeplex and the global best frog. In VNS, five neighborhood structures are adopted. The first one is the Swap neighborhood structure, which works by swapping the positions of two randomly selected operations. The second one is the Insert neighborhood structure, which works by inserting the back operation just before the front operation. The third one is the Inverse neighborhood structure, and it inverts the subsequence between two randomly selected positions. Obviously, the Swap, Insert and Inverse neighborhoods are for OS string. The fourth one, denoted as ReassignF, works by randomly selecting one job and changing its FS string to another factory. The fifth one, denoted as ReassignM, works by randomly selecting one operation and changing its MS string to another feasible machine of its selected factory.

C. THE DETAILED STEPS OF HSFLA

The flow chart of the proposed HSFLA is shown in Figure 6, and the detailed steps of the HSFLA are described as follows:

Step 1: Set the algorithm parameters, such as the number of fogs in each memeplex P_{msize}, the number of memeplex N_m, the evolution times of each memeplex N_{me} and local search times $t_{\max}$ of VNS. Then, generate the initial population by the initialization method in Section IV.A.3).

Step 2: Repeat the following steps until the termination condition is met.

Step 3: Partition the population into N_m memeplexes with the binary tournament selection rule.

Step 4: For each memeplex, repeat the following Steps 4.1-4.2 N_{me} times

Step 4.1: Determine the local best frog x_{lb} and local worst frog x_{lw} in each memeplex. Then, cross x_{lb} and a randomly selected one frog of this memeplex with POX crossover for OS or UC crossover for FS with the same probability of 50%, and two new frogs are generated. If the best new frog x_{nb} of the two newly generated frogs is better than x_{lw}, then replace x_{lw} with x_{nb}. If x_{lw} is not improved, randomly generate a new neighborhood solution of the global best solution of the population x_{gb} by randomly using the five neighborhoods of VNS and replace x_{lw}.

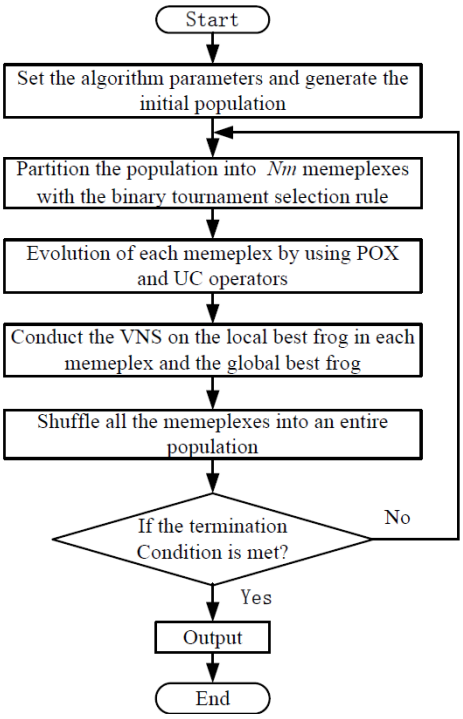

FIGURE 6. The flow chart of the proposed HSFLA.

Step 4.2: Cross the local best frog x_{lb} of the current memeplex with that of randomly selected another memeplex and four new frogs are generated. If the best new frog x_{nb} is better than x_{lw}, then replace x_{lw} with x_{nb}.

Step 5: Conduct the VNS on the local best frog x_{lb} in each memeplex and the global best frog x_{gb}.

Step 5.1: Set the initial solution x and a set of neighborhood structures $N_k(x)$, $k = 1 \ldots 5$.

Step 5.2: Repeat the following Steps 3-6 until the stop condition is satisfied $k > k_{\max}$.

Step 5.3: Set $k = 1$.

Step 5.4: Randomly generate a solution x' from the kth neighborhood of $x(x' \in N_k(x))$.

Step 5.5: Apply following Steps 5.5.1-5.5.4 to solution x'.

Step 5.5.1: Set $t = 1$.

Step 5.5.2: Randomly generate a solution x'' from the kth neighborhood of solution $x'(x'' \in N_k(x'))$.

Step 5.5.3: If $f(x'') < f(x')$, replace x' with x'' and set $t = t + 1$; otherwise, set $t = t + 1$.

Step 5.5.4: Repeat Step 5.2-5.3 until $t > t_{\max}$.

Step 5.6: If $f(x') < f(x)$, replace x with x' and set $k = 1$; otherwise, set $k = k + 1$.

Step 6: Shuffle the N_m memeplexes into an entire population, and go to **Step 2**.

D. COMPUTATIONAL COMPLEXITY ANALYSIS

The computation complexity of HSFLA is determined by the computation complexity of each step of HSFLA. In detail, the computation complexity of population initialization (Step 1) is $O(P_{msize} \times N_m \times \sum_{i \in I} n_i)$. For the partition of memeplexes (Step 3), the computation complexity is $O(P_{msize} \times N_m)$. For Steps 4.1 and 4.2 of HSFLA, POX and UC crossovers are performed $2 \times N_{me} \times N_m$ times, the computation complexity is $O(N_{me} \times N_m \times \sum_{i \in I} n_i)$. The computation complexity of VNS (Step 5) is $O(N_m \times k_{\max} \times t_{\max})$. Therefore, the total computation complexity of each generation of HSFLA is $O(P_{msize} \times N_m \times \sum_{i \in I} n_i + P_{msize} \times N_m + N_{me} \times N_m \times \sum_{i \in I} n_i + N_m \times k_{\max} \times t_{\max})$. Clearly, it depends on the parameters of $P_{msize}, N_m, N_{me}, t_{\max}, k_{\max}$ and $\sum_{i \in I} n_i$, among which P_{msize}, N_m, N_{me} and $t_{\max}$ are variable. The influence of these variable parameters will be investigated by experiments in the following Section of "Parameter settings".

V. COMPARATIVE EVALUATIONS AND DISCUSSIONS

This section aims to evaluate the performance of the proposed MILP model and HSFLA. The HSFLA algorithm runs in C++ on a desktop Dell Vostro 3900, which is with 3.20 GHz Intel processor and 8 GB memory. As to the MILP model, it runs in OPL language of IBM CPLEX Studio IDE 12.7.1. The solving method of CPLEX solver is branch-and-cut method, and it is the combination of cutting plane and branch-and-bound methods [43]. It works by running a branch-and-bound algorithm and using cutting planes to tighten the linear programming relaxations. For the experiments, 20 instances adapted from MFJS01-10 [44] and MK01-10 [45] are used. Each of the 20 instances is obtained by supposing that all the factories are the same. Cases with only two factories are considered. With regard to the processing power, it is randomly produced with uniform distributions U [4], [8]. With regard to the idle power, it is randomly produced among {1, 2, 3}. The time, breakeven time and energy consumption for Turn Off/On strategy of each machine are randomly produced among {8, 12, 16}, {10, 15, 20} and {10, 30, 60} correspondingly. Moreover, the maximum times of Turn Off/On strategy for each machine and the common power are set to 3 and 20 respectively.

A. EVALUATION OF THE MILP MODEL

The computational results of MILP model for 20 instances are reported in Table 1. In Table 1, NBVs, NCVs and NCs denote the number of binary variables, the number of continuous variables and the number of constraints respectively. Nodes and Iterations represent the number of nodes and iterations for solving the instance to optimality. Gap represents the average optimality gap of the obtained solution within the timelimit. A solution with the gap value of 0 is optimal. Time represents for the solving time. Moreover, NBVs, NCVs, NCs, Nodes and Iterations are proportional to the instance size. NBVs, NCVs and NCs can be obtained by the instance size in advance. However, Nodes and Iterations can only be obtained when the instance is solved by solution method to

TABLE 1. Results of MILP model for 20 instances.

Instances	Size($n\times n_{max}\times m$)	NBVs	NCVs	NCs	Nodes	Iterations	Solution	Gap%	T(s)
MFJS01	5×3×6	1054	99	1227	41311	419177	20204.0	0	17.00
MFJS02	5×3×7	1244	111	1477	23981	244863	18584.0	0	12.34
MFJS03	6×3×7	1822	135	2094	296554	4476630	22154.8	0	248.48
MFJS04	7×3×7	2464	157	2671	995340	17823918	26327.8	0	1099.61
MFJS05	7×3×7	2420	155	2615	862477	12995062	25419.0	0	685.63
MFJS06	8×3×7	3102	175	3224	2261535	33038926	30319.4	0	2628.59
MFJS07	8×4×7	5150	223	4912	-	-	43488.4	13.57	3600
MFJS08	9×4×8	6366	247	5315	-	-	48044.4	12.71	3600
MFJS09	11×4×8	9276	297	7405	-	-	60280.4	17.44	3600
MFJS10	12×4×8	10984	323	8726	-	-	70192.2	16.66	3600
MK01	10×6×6	12888	343	12288	-	-	-	-	3600
MK02	10×6×6	28328	599	41776	-	-	-	-	3600
MK03	15×10×8	136216	1205	110297	-	-	-	-	3600
MK04	15×9×8	31318	527	20881	-	-	-	-	3600
MK05	15×9×4	38756	577	36705	-	-	-	-	3600
MK06	10×15×15	147980	1283	117032	-	-	-	-	3600
MK07	20×5×5	57196	769	68808	-	-	-	-	3600
MK08	20×14×10	145566	1097	58410	-	-	-	-	3600
MK09	20×14×10	292112	1695	168914	-	-	-	-	3600
MK10	20×14×15	345130	1915	223396	-	-	-	-	3600

optimality, and they represent for the size of the solution space.

As seen from Table 1, MILP model can solve relatively small-sized instances namely MFJS01-06 to optimality with 17.00s, 12.34s, 248.48s, 1099.61s, 685.63s and 2628.59s. However, with regard to MFJS07 and even larger-sized instances, MILP model cannot solve them to optimality within 3600s. As to relatively large-sized instances namely MK01-10, MILP model fails to find any feasible solution within 3600s. This is because the CPLEX solver uses branch-and-cut method to solve MILP model. Brand-and-cut method is the combination of cutting plane and branch-and-bound methods, which is heavily dependent on the size of the solving problem. When the scale of instance increases, more constraints (NCs), more decision variables (NBVs and NCVs) and bigger solution space (Nodes and Iterations) are with the MILP model, resulting in difficult branching, finding new low bounds and cutting. Obviously, all of these show that the MILP model is not efficient for solving relatively large-sized instances. However, there is no denying that the MILP model can solve small-sized instances to optimality. It is very important to obtain optimal solutions for scheduling problem, particularly for the new one. The optimal solution is the standard to design approximate methods such as heuristic and meta-heuristic algorithms.

B. EVALUATION OF THE PROPOSED HSFLA

In this section, we evaluate the performance of HSFLA. Moreover, this section includes three subsections namely parameters setting of HSFLA, evaluation of the effectiveness of the energy-saving decoding method and comparison of HSFLA with other meta-heuristic algorithms.

1) PARAMETER SETTINGS

According to Section 4, four parameters namely the population size of each memeplex P_{msize}, the number of memeplexes N_m, the memetic evolution times N_{me} and local search times $t_{\max}$ of VNS need to be decided. We used three settings of 3,5,10 for P_{msize}, 5,10,15 for N_m, 5,10,15 for N_{me} and 5,10,15 for $t_{\max}$. Therefore, we use Taguchi method of design of experiment (DOE) to determine these four parameters. Moreover, the MK05 is chosen to conduct the DOE test. For each test, we set the maximum CPU time of nm seconds as the stopping condition. Each test is repeated 20 times, and the mean value (Mean) is set as the response value. Table 2 shows the results of the DOE test. Figure 7 presents the trend of each factor level. Moreover, Table 3 shows the significance rank of each parameter. As can be seen from Table 3, N_{me} is the most significant factor. P_{msize}, N_m and $t_{\max}$ rank the second, the third and the fourth respectively. Based on the results, we select the following settings: $P_{msize} = 10$, $N_m = 10$, $N_{me} = 5$ and $t_{\max} = 10$.

TABLE 2. Results of the DOE test.

Test	P_{msize}	N_m	N_{me}	$t_{\max}$	Mean
1	3	5	5	5	77995.4
2	3	10	10	10	77884.6
3	3	15	15	15	78479.4
4	5	5	10	15	78232.8
5	5	10	15	5	78060.8
6	5	15	5	10	77847.6
7	10	5	15	10	78096.0
8	10	10	5	15	77604.8
9	10	15	10	5	77848.6

TABLE 3. Average response values.

Level	P_{msize}	N_m	N_{me}	$t_{\max}$
1	78120	78008	77816	77968
2	78047	77850	77989	77943
3	77850	78059	78212	78106
Delta	270	256	396	163
Rank	2	3	1	4

2) EVALUATION OF POSTPONING STRATEGY AND TURN OFF/ON STRATEGY

This section intends to evaluate the effectiveness of both postponing strategy and Turn Off/On strategy. To this end,

TABLE 4. Comparisons of ADM and P-ADM (effectiveness of postponing strategy).

Problem	MFJS01	MFJS02	MFJS03	MFJS04	MFJS05	MFJS06	MFJS07	MFJS08	MFJS09	MFJS10
RPI_Min	1.42	2.31	3.23	2.83	3.62	2.83	2.32	3.81	4.20	4.77
RPI_Ave	7.39	8.13	8.09	8.13	7.91	7.68	7.44	9.49	9.62	9.66
RPI_Max	14.06	14.68	15.43	15.26	14.06	15.14	12.04	14.90	14.97	14.05
Problem	MK01	MK02	MK03	MK04	MK05	MK06	MK07	MK08	MK09	MK10
RPI_Min	1.04	0.77	1.10	1.06	0.21	1.01	0.41	1.27	1.29	1.44
RPI_Ave	5.09	4.74	4.00	6.96	1.55	5.63	2.62	4.33	5.78	5.54
RPI_Max	11.00	10.02	7.24	12.79	4.22	10.31	5.76	7.03	9.39	10.53

TABLE 5. Comparisons of P-ADM and EE-ADM (effectiveness of Turn Off/On strategy).

Problem	MFJS01	MFJS02	MFJS03	MFJS04	MFJS05	MFJS06	MFJS07	MFJS08	MFJS09	MFJS10
RPI_Min	0	0	0	0	0	0	0	0	0	0
RPI_Ave	0.42	0.35	0.40	0.47	0.57	0.78	1.25	1.20	1.22	1.34
RPI_Max	4.04	5.26	5.33	3.42	3.06	5.89	5.31	5.90	4.68	4.97
Problem	MK01	MK02	MK03	MK04	MK05	MK06	MK07	MK08	MK09	MK10
RPI_Min	1.90	2.42	6.30	3.14	2.32	11.10	1.76	7.50	8.08	9.85
RPI_Ave	5.75	5.99	11.53	7.54	4.95	15.68	5.00	12.14	12.81	15.83
RPI_Max	11.92	10.74	17.40	14.45	8.32	20.91	8.47	17.57	20.15	21.12

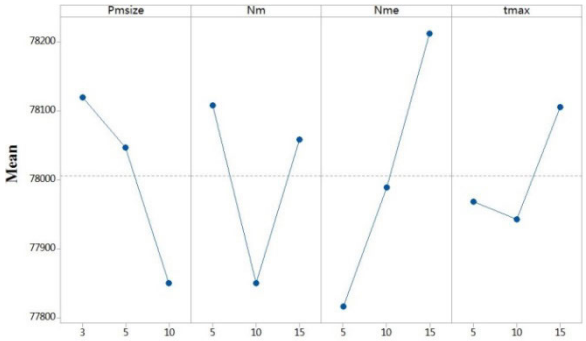

FIGURE 7. The trend of each factor level of each parameter.

firstly, we randomly generate 1000 individuals. Then, we decode them by using three decoding methods namely ADM, ADM with only postponing strategy (P-ADM) and ADM with both postponing strategy and Turn Off/On strategy (EE-ADM). Moreover, the relative percentage increase (RPI) value is set as the comparison norm, and it is computed according to Eq.(26),

$$RPI = (D_a - D_b)/D_a \times 100\%, \quad \forall a, b \in \{1, \ldots, 3\} \quad (26)$$

where, D_a denotes the fitness archived by method a.

In Tables 4 and 5, RPI_Min represents for minimum RPI of the 1000 individuals, RPI_Ave represents for average RPI of the 1000 individuals, and RPI_Max represents for maximum RPI of the 1000 individuals. To be more specific, Table 4 shows the comparison results of ADM and P-ADM. Obviously, P-ADM is better than ADM in terms of RPI_Min, RPI_Ave and RPI_Max. RPI_Min ranges from 0.21 to 4.77, RPI_Ave ranges from 1.55 to 9.66 and RPI_Max ranges from 4.22 to 15.43. Moreover, we can see that postponing strategy is more effective for MFJS01-10 than MK01-10. In all, postponing strategy performs well in reducing idle energy consumption.

Table 5 reports the comparison results of P-ADM and EE-ADM. This is clear that EE outperforms P-ADM. Unlike postponing strategy, Turn Off/On strategy is more effective for MK01-10 than MFJS01-10. This is because the size of MFJS01-10 is relatively small, and the number of idle time periods may be smaller than that of MK01-10. Therefore, postponing strategy is more effective for MFJS01-10 than MK01-10, and Turn Off/On strategy is more effective for MK01-10 than MFJS01-10. Moreover, this phenomenon may also be attributed to that different instance has different processing data such as processing time, processing time, machine flexibility and processing power.

3) COMPARISONS OF HSFLA WITH OTHER ALGORITHMS

In this section, we prove the effectiveness of HSFLA by comparing it with the proposed MILP model, SFLA without VNS, VNS and GA. The parameter settings of HSFLA are set to $P_{msize} = 10, N_m = 10, N_{me} = 5$ and $t_{\max} = 10$. The parameter settings of SFLA are set to $P_{msize} = 10, N_m = 10$, $N_{me} = 5$. $t_{\max}$ of VNS is set to 10. With regard to the GA, the selection operator uses binary tournament selection, the cross operators use the POX and UC described in Section IV.A.5), the mutation operators use Swap, Insert, Inverse, ReassignF and ReassignM described in Section IV.B) for VNS. By try and trail, the population size, the cross possibility and the mutation possibility are set to 100, 0.9 and 0.05 respectively. Moreover, for the sake of fairness, all the algorithms adopt the same stopping criteria of maximum elapsed CPU time of nm seconds. The comparison results are reported in Table 6. In Table 6, the Best denotes the best solution obtained by all the algorithms of MILP model, HSFLA, SFLA, GA and VNS. Min represents the RPI of the fitness of the best solution obtained by one algorithm of 20 runs with the fitness of Best. Similarly, AV is the RPI of the average fitness of the best solution obtained by one algorithm of 20 runs with the fitness of Best. T(s) is the computational time in seconds.

In terms of Min and AV, HSFLA performs no worse than SFLA, GA and VNS for all 20 instances, which shows the superiority of HSFLA. HSFLA outperforms SFLA, which shows the effectiveness of VNS. This can be attributed to that

TABLE 6. Comparison results of HSFLA with other methods.

Instance	Best	MILP		HSFLA			SFLA			GA			VNS		
		Min	T(s)	Min	AV	T(s)	Min	AV	T(s)	Min	AV	T(s)	Min	AV	T(s)
MFJS01	**20204.0**	**0**	17.00	0.10	1.89	30	0.15	2.23	30	0.15	2.67	30	2.784	11.364	30
MFJS02	**18584.0**	**0**	12.34	0.16	1.19	35	0.43	1.96	35	0.82	2.27	35	11.442	23.586	35
MFJS03	**22154.8**	**0**	248.48	**0**	1.90	42	0.32	2.65	42	0.56	3.13	42	14.316	24.354	42
MFJS04	**26327.8**	**0**	1099.61	0.96	3.07	49	2.14	4.23	49	3.77	4.61	49	14.694	24.144	49
MFJS05	**25419.0**	**0**	685.63	0.23	2.18	49	0.23	2.68	49	0.23	2.94	49	17.52	27.684	49
MFJS06	**30319.4**	**0**	2628.59	0.14	3.77	56	2.56	4.12	56	3.04	4.47	56	17.376	27.546	56
MFJS07	**41717.2**	4.25	3600	**0**	2.84	56	1.24	3.22	56	1.44	3.88	56	18.03	24.426	56
MFJS08	**46996.2**	2.23	3600	**0**	2.39	72	1.35	3.12	72	2.09	3.93	72	9.96	18.738	72
MFJS09	**59072.6**	2.04	3600	**0**	2.36	88	0.32	2.68	88	0.53	2.64	88	6.972	16.41	88
MFJS10	**67581.2**	3.86	3600	**0**	0.21	96	1.45	1.96	96	1.60	2.24	96	10.338	16.536	96
MK01	**17551.0**	-	3600	**0**	2.15	60	1.68	2.99	60	2.36	3.38	60	15.06	40.248	60
MK02	**14550.0**	-	3600	**0**	3.10	60	1.76	3.25	60	2.73	4.08	60	38.346	46.182	60
MK03	**91756.0**	-	3600	**0**	0.45	120	0.68	0.66	120	0.05	0.69	120	33.918	37.29	120
MK04	**35262.0**	-	3600	**0**	1.90	120	0.96	2.21	120	0.61	2.45	120	15.03	22.56	120
MK05	**76000.0**	-	3600	**0**	2.09	60	1.66	3.32	60	1.75	3.26	60	6.498	14.34	60
MK06	**37498.0**	-	3600	**0**	2.15	150	1.21	2.65	150	1.68	3.60	150	45.102	58.734	150
MK07	**70746.0**	-	3600	**0**	2.09	100	0.88	3.56	100	0.18	2.71	100	39.27	43.614	100
MK08	**253666.0**	-	3600	**0**	0.39	200	0.55	1.96	200	0.33	1.37	200	9.252	10.896	200
MK09	**200714.0**	-	3600	**0**	1.69	200	1.21	2.55	200	1.23	2.77	200	20.148	22.734	200
MK10	**128085.0**	-	3600	**0**	1.64	260	0.45	1.98	260	0.76	2.30	260	24.396	26.274	260
Mean		-		**0.08**	1.97		1.06	2.70		1.30	2.97		18.52	26.88	

The solution in bold is the best one.

TABLE 7. Paired t-test for the Min values of different algorithms.

Algorithm	95%*CI*-lower	95%*CI*-upper	*t*-value	*p*-value
HSFLAvs.SFLA	0.679	1.285	6.78	0.000
HSFLAvs.GA	0.757	1.675	5.54	0.000
HSFLAvs.VNS	12.89	24.00	6.95	0.000
SFLAvs.GA	-0.016	0.484	1.96	0.065
SFLAvs.VNS	11.95	22.97	6.63	0.000

TABLE 8. Paired t-test for the AV values of different algorithms.

Algorithm	95%*CI*-lower	95%*CI*-upper	*t*-value	*p*-value
HSFLAvs.SFLA	0.502	0.951	6.76	0.000
HSFLAvs.GA	0.789	1.205	10.04	0.000
HSFLAvs.VNS	19.16	30.66	9.07	0.000
SFLAvs.GA	0.067	0.474	2.78	0.012
SFLAvs.VNS	18.38	29.99	8.72	0.000

the local search ability of HSFLA is improved with embedding VNS. SFLA is a population-based algorithm, and its local searching ability is very limited. From Table 6, we can also see that SFLA outperforms GA. Specifically, in terms of mean Min, SFLA can obtain better mean Min of 1.06 than 1.30 of GA. With regard to mean AV, SFLA obtains better mean AV of 2.70 than 2.97 of GA. This is because except for the specific operators for generating new solutions, the flow of the meta-heuristic algorithms is also very important. The flow of SFLA is better than GA for solving energy-efficient DFJSP. SFLA can take the advantages of multi-population evolution and evolution operator s of GA. With regard to VNS, it performs worst of all the four meta-heuristic algorithms in terms of both Min and AV. This may be because that VNS only evolves with one solution, and it is with strong randomicity and cannot search for better solution space of DFJSP with large solution space reliably.

Moreover, Table 7 and Table 8 show the paired-t test at 95% confidence level of Min and AV values of different algorithms. Obviously, in terms of both Min and AV values, the p-values of HSFLAvs.SFLA, HSFLAvs.GA, HSFLAvs. VNS are less than 0.05. Therefore, HSFLA is statistically better than the other algorithms. Tables 7 and 8 also indicate that SFLA is statistically better than VNS. In terms of Min value, the p-value of SFLAvs.GA is 0.065 and is greater than 0.05. With regard to p-value of SFLAvs.GA in Table 9, it is 0.012 and less than 0.05. One the whole, SFLA statistically performs better than GA.

As we can see from Table 6, for all the instances, the maximum solving time is 260s, and it is acceptable. With regard to real instances, the stopping criteria for them should be set according to actual conditions.

VI. CONCLUSION AND FUTURE RESEARCH

This paper addresses DFJSP with considering energy consumption. A novel MILP model is developed to obtain optimal solution for small-sized instances, and an efficient HSFLA is designed to obtain near-optimal solutions for large-sized instances. In HSFLA, the encoding method, the decoding method, the initiation method and the memetic evolution process are specifically designed. VNS is embedded in the algorithm to enhance its local

exploitation capability. Numerical experiments are conducted to prove the effectiveness of the MILP model and the HFSLA.

With future research, we will extend the MILP modeling idea and HSFLA in this paper to other distributed scheduling problems. Moreover, we will design other algorithms such as migrating birds optimization (MBO) algorithm, teaching-learning-based optimization (TLBO) algorithm and evolution (DE) algorithm to solve DFJSP. We welcome other researchers to propose more efficient MILP model and meta-heuristic algorithms for solving the energy-efficient DFJSP of this paper. Furthermore, the multi-objective DFJSP with simultaneously optimizing total energy consumption and makespan will be considered in the near future.

REFERENCES

[1] L. Meng, C. Zhang, X. Shao, and Y. Ren, "MILP models for energy-aware flexible job shop scheduling problem," *J. Cleaner Prod.*, vol. 210, pp. 710–723, Feb. 2019.

[2] B. Zhang, Q.-K. Pan, L. Gao, L.-L. Meng, X.-Y. Li, and K.-K. Peng, "A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, May 29, 2019, doi: 10.1109/TSMC.2019.2916088.

[3] Y. Ren, H. Jin, F. Zhao, T. Qu, L. Meng, C. Zhang, B. Zhang, G. Wang, and J. W. Sutherland, "A multiobjective disassembly planning for value recovery and energy conservation from end-of-life products," *IEEE Trans. Autom. Sci. Eng.*, early access, Jun. 2, 2020, doi: 10.1109/TASE.2020.2987391.

[4] S. Jia, Q. Yuan, W. Cai, J. Lv, and L. Hu, "Establishing prediction models for feeding power and material drilling power to support sustainable machining," *Int. J. Adv. Manuf. Technol.*, vol. 100, nos. 9–12, pp. 2243–2253, Feb. 2019.

[5] S. Jia, Q. Yuan, W. Cai, M. Li, and Z. Li, "Energy modeling method of machine-operator system for sustainable machining," *Energy Convers. Manage.*, vol. 172, pp. 265–276, Sep. 2018.

[6] S. Jia, Q. Yuan, J. Lv, Y. Liu, D. Ren, and Z. Zhang, "Therblig-embedded value stream mapping method for lean energy machining," *Energy*, vol. 138, pp. 1081–1098, Nov. 2017.

[7] M. Dai, D. Tang, A. Giret, and M. A. Salido, "Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints," *Robot. Comput.-Integr. Manuf.*, vol. 59, pp. 143–157, Oct. 2019.

[8] Y. Han, J. Li, H. Sang, Y. Liu, K. Gao, and Q. Pan, "Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106343.

[9] K. Gao, Y. Huang, A. Sadollah, and L. Wang, "A review of energy-efficient scheduling in intelligent production systems," *Complex Intell. Syst.*, vol. 6, no. 2, pp. 237–249, Jul. 2020.

[10] Q. Lu, Y. Ren, H. Jin, L. Meng, L. Li, C. Zhang, and J. W. Sutherland, "A hybrid Metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101828.

[11] Q.-K. Pan, L. Gao, L. Wang, J. Liang, and X.-Y. Li, "Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem," *Expert Syst. Appl.*, vol. 124, pp. 309–324, Jun. 2019.

[12] J. Li, S. Bai, P. Duan, H. Sang, Y. Han, and Z. Zheng, "An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system," *Int. J. Prod. Res.*, vol. 57, no. 22, pp. 6922–6942, 2019, doi: 10.1080/00207543.2019.1571687.

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

[14] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Comput. Ind. Eng.*, vol. 142, Apr. 2020, Art. no. 106347.

[15] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems," *Comput. Oper. Res.*, vol. 80, pp. 101–112, Apr. 2017, doi: 10.1016/j.cor.2016.11.021.

[16] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh, and Y. F. Zhang, "A modified genetic algorithm for distributed scheduling problems," *J. Intell. Manuf.*, vol. 14, nos. 3–4, pp. 351–362, Jun. 2003.

[17] B. Naderi and A. Azab, "Modeling and heuristics for scheduling of distributed job shops," *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7754–7763, Dec. 2014.

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

[19] P.-H. Lu, M.-C. Wu, H. Tan, Y.-H. Peng, and C.-F. Chen, "A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 1, pp. 19–34, Jan. 2018.

[20] M. Ziaee, "A heuristic algorithm for the distributed and flexible job-shop scheduling problem," *J. Supercomput.*, vol. 67, no. 1, pp. 69–83, Jan. 2014.

[21] B. Marzouki, O. B. Driss, and K. Ghédira, "Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic," *Procedia Comput. Sci.*, vol. 126, pp. 1424–1433, Jan. 2018.

[22] X. Wu, X. Liu, and N. Zhao, "An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem," *Memetic Comput.*, vol. 11, no. 4, pp. 335–355, 2018.

[23] E.-D. Jiang, L. Wang, and Z.-P. Peng, "Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition," *Swarm Evol. Comput.*, vol. 58, Nov. 2020, Art. no. 100745.

[24] L. Meng, C. Zhang, X. Shao, B. Zhang, Y. Ren, and W. Lin, "More MILP models for hybrid flow shop scheduling problem and its extended problems," *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 3905–3930, Jul. 2020.

[25] J.-Q. Li and Y.-Q. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Comput.*, Dec. 2019, doi: 10.1007/s10586-019-03022-z.

[26] F. K. Goodarzi, N. A. Taghinezhad, and S. H. Nasseri, "A new fuzzy approach to solve a novel model of open shop scheduling problem," *Univ. Politeh. Buchar. Sci. Bull. A, Appl. Math. Phys.*, vol. 76, no. 3, pp. 199–210, 2014.

[27] H. Naseri and N. A. Taghi-Nezhad, "Reactive scheduling presentation for an open shop problem focused on job's due dates," *J. Prod. Oper. Manage.*, vol. 6, no. 2, pp. 95–112, 2015.

[28] H. M. Wagner, "An integer linear-programming model for machine scheduling," *Nav. Res. Logistics Quart.*, vol. 6, no. 2, pp. 131–140, Jun. 1959.

[29] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J. Water Resour. Planning Manage.*, vol. 129, no. 3, pp. 210–225, May 2003.

[30] Q.-K. Pan, L. Wang, L. Gao, and J. Li, "An effective shuffled frog-leaping algorithm for lot-streaming flow shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 52, nos. 5–8, pp. 699–713, Feb. 2011.

[31] D. Lei, Y. Zheng, and X. Guo, "A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3126–3140, Jun. 2017.

[32] Y. Xu, L. Wang, M. Liu, and S.-Y. Wang, "An effective shuffled frog-leaping algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *Int. J. Adv. Manuf. Technol.*, vol. 68, nos. 5–8, pp. 1529–1537, Sep. 2013, doi: 10.1007/s00170-013-4940-y.

[33] J. Li, Q. Pan, and S. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Appl. Math. Comput.*, vol. 218, no. 18, pp. 9353–9371, May 2012.

[34] K. Lu, L. Ting, W. Keming, Z. Hanbing, T. Makoto, and Y. Bin, "An improved shuffled frog-leaping algorithm for flexible job shop scheduling problem," *Algorithms*, vol. 8, no. 1, pp. 19–31, Feb. 2015.

[35] D. Lei and X. Guo, "A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9333–9339, Dec. 2015.

[36] L. Wang and C. Fang, "An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem," *Inf. Sci.*, vol. 181, no. 20, pp. 4804–4822, Oct. 2011.

[37] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," *Int. J. Prod. Econ.*, vol. 174, pp. 93–110, Apr. 2016.

[38] C. Zhang, Y. Rao, and P. Li, "An effective hybrid genetic algorithm for the job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 39, nos. 9–10, pp. 965–974, Nov. 2008.
[39] L. Meng, C. Zhang, B. Zhang, and Y. Ren, "Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility," *IEEE Access*, vol. 7, pp. 68043–68059, 2019.
[40] L. Meng, C. Zhang, X. Shao, Y. Ren, and C. Ren, "Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines," *Int. J. Prod. Res.*, vol. 57, no. 4, pp. 1119–1145, Feb. 2019.
[41] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, May 2001.
[42] H. Karimi, S. H. A. Rahmati, and M. Zandieh, "An efficient knowledge-based algorithm for the flexible job shop scheduling problem," *Knowl.-Based Syst.*, vol. 36, p. 236, Dec. 2012.
[43] Y. Ren, L. Meng, C. Zhang, F. Zhao, U. Saif, A. Huang, G. P. Mendis, and J. W. Sutherland, "An efficient metaheuristics for a sequence-dependent disassembly planning," *J. Cleaner Prod.*, vol. 245, Feb. 2020, Art. no. 118644.
[44] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *J. Intell. Manuf.*, vol. 18, no. 3, pp. 331–342, Jul. 2007.
[45] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.

LEILEI MENG received the B.S. degree in mechanical engineering from Chang'an University, Xi'an, China, in 2014, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology (HUST), China, in 2020.

He is currently a Lecturer with the School of Computer Science, Liaocheng University. His research mainly focuses on modeling, optimization of scheduling problems, tool wear prediction, and sustainable manufacturing.

YAPING REN received the B.S. degree in communications and transportation engineering from Liaocheng University, Liaocheng, China, in 2014, the M.S. degree from the Transportation College, Northeast Forestry University, Harbin, China, in 2016, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2019. From September 2017 to August 2019, he was a Visiting Scholar with the Environmental and Ecological Engineering (EEE), Purdue University, West Lafayette, IN, USA. He is currently an Associate Professor with the School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai, China. His research mainly focuses on industrial engineering, disassembly planning, transportation planning, decision making, and optimization methods.

BIAO ZHANG received the B.S. degree from the School of Computer Science and Technology, Shandong University of Technology, Zibo, China, the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2012, 2015, and 2019, respectively.

He is currently a Lecturer with the School of Computer Science, Liaocheng University. His current research interests include discrete optimization and scheduling.

JUN-QING LI received the master's degree in computer science and technology from Shandong Economic University, Shandong, China, in 2004, and the Ph.D. degree from Northeastern University, Shenyang, China, in 2016. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, where he became a Professor in 2017. He is also with the School of Computer Science, Liaocheng University. He has authored more than 100 refereed articles. His current research interests include intelligent optimization and scheduling.

HONGYAN SANG received the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2010, and the Ph.D. degree in industrial engineering from the Huazhong University of Science Technology, Wuhan, China, in 2013. Since 2003, she has been with the School of Computer Science, Liaocheng University, where she became an Associate Professor, in 2016. She has authored more than 60 refereed articles. Her current research interests include intelligent optimization and scheduling.

CHAOYONG ZHANG (Member, IEEE) received the B.S. degree in mechanical engineering from the Tianjin University of Science and Technology, Tianjin, China, in 1993, the M.S. degree in mechatronic engineering from the University of Science and Technology Beijing, Beijing, China, in 1999, and the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2007. He is currently a Professor with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology (HUST). His research is mainly focused on modeling, optimization, and scheduling for production manufacturing systems, efficient and effective resource utilization of manufacturing systems, sustainable manufacturing, including clean and high efficient manufacturing processes, power consumption model of machine tools, and so on. He has authored two books, and has authored or coauthored over 70 academic articles. He has been honored by: (1) 2013' the Ministry of Education Natural Science first place prize; (2) 2010'China Machinery Industry Federation first place prize; and (3) 2008' the Ministry of Education Science and Technology Progress first place prize.