Mathematical Modeling and Optimization Of Flexible Job Shop Problem with Sequence-Dependent Setup and Transportation Energy Consumptions

Leilei Meng (mengleilei@lcu-cs.com)
Liaocheng University https://orcid.org/0000-0003-1439-4832

Research Article

Keywords: Energy consumption, flexible job shop scheduling, sequence-dependent setup times, transportation times, mixed integer linear programming, hybrid algorithm

DOI: https://doi.org/10.21203/rs.3.rs-683968/v1

Mathematical Modeling and Optimization Of Flexible Job Shop Problem with Sequence-Dependent Setup and Transportation Energy Consumptions

LEILEI MENG[1], BIAO ZHANG[1], YAPING REN[2], JUNQING LI[1], HONGYAN SANG[1], AND CHAOYONG ZHANG[3]

[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 Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

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

ABSTRACT As environmental awareness grows, energy-aware scheduling is attracting increasing attention. This paper investigates the flexible job shop scheduling problem with sequence-dependent setup times and transportation times (FJSP-SDST-T) and the objective is to minimize total energy consumption. To begin with, the total energy consumption of the workshop is analyzed and a novel mixed integer linear programming (MILP) model is formulated. Due to that FJSP-SDST-T is NP-hard, an effective hybrid algorithm (HGA) that hybridizes the genetic algorithm (GA) and variable neighborhood search (VNS) algorithm is proposed to solve the problem specifically for that with large size. HGA takes advantage of the good global searching ability of GA and the powerful local searching ability of VNS, and it can have a good balance of intensification and diversification. Then, four energy-conscious decoding methods are designed, in which two energy-saving strategies namely postponing strategy and Turn Off/On strategy are specially designed according to the characteristics of FJSP-SDST-T. Finally, experiments are carried out and the results show the effectiveness of the MILP model, the energy-conscious decoding methods and HGA.

INDEX TERMS Energy consumption, flexible job shop scheduling, sequence-dependent setup times, transportation times, mixed integer linear programming, hybrid algorithm

I. INTRODUCTION

Flexible job shop scheduling problem (FJSP), as an extension of the classical job shop scheduling problem (JSP), is of great significance in the modern manufacturing system. For FJSP, an operation is allowed to be processed by different machines, and two sub-problems namely the machine selection of all operations and the operations sequencing on all machines should be decided. In addition, FJSP has proven to be a type of NP-hard problems [1].

In previous studies, for solving FJSP, most of the research aims at minimizing makespan [2]. However, in recent years, with the dual pressure of environmental issues and energy costs, more and more attention are being paid to energy-conscious scheduling, which has been proved to be effective in reducing the energy consumption with no or little capital investment [3-6]. In actual production, some machine tools stay in idle state for a long time, and they can be turned off and then back on for saving energy. This energy-saving strategy is called Turn Off/On strategy [7], and it has been widely implemented in different scheduling problems [3, 4, 8-14].

In the classical FJSP, setup and transportation times are overlooked, which does not conform to the actual production situation. In real cases, jobs cannot be processed on the next machine immediately after its completion on the previous machine; instead, they should be transported between the machines by transportation systems [15]. Moreover, when an operation is completed on a machine, the machine has to be equipped with appropriate tools for the next operation, which incurs sequence-dependent setup times. The FJSP with sequence-dependent setup and transportation times (FJSP-SDST-T) is more in line with actual situation and should be studied.

Our work focuses on minimizing the total energy consumption of FJSP-SDST-T. Moreover, Turn Off/On is implemented to save energy. The aim of our work is to solve FJSP-SDST-T with both an exact method named MILP model and a hybrid mete-heuristic algorithm named HGA.

Comparing with previous studies, the contributions of our work can be summarized as three aspects, which are described as follows:

(1) To the best of our knowledge, this paper is the first attempt to study energy-conscious FJSP-SDST-T with considering Turn Off/On strategy.

(2) A novel MILP model is firstly formulated for FJSP-SDST-T with minimizing total energy consumption.

(3) An effective hybrid algorithm HGA that integrates GA with VNS is proposed. Moreover, in HGA, four energy-conscious decoding methods are designed specifically for FJSP-SDST-T.

The rest of the paper is organized as follows. Section 2 introduces the related works of FJSP-SDST-T. Section 3 formulates the energy-efficient MILP model. Section 4 elaborates the hybrid algorithm HGA. Section 5 gives the experimental evaluations. Conclusions and future works are presented in Section 6.

II. LITERATURE REVIEW

With regard to the scheduling problem, it can be solved by two classes of methods, among which the first one is the exact method and the second one is approximation method. For exact methods, they mainly contain branch-and-bound algorithm [16] and mixed integer programming (MIP) [1, 2, 4, 17-20] among others. As to approximation methods, they include heuristic rules and meta-heuristic algorithms. The MIP model can solve the small-sized problems to optimality and can elaborate all the characteristics of a scheduling problem. Besides, it is very important for designing new dispatching rules. Recent years, MIP solvers such as Cplex and Gurobi improve a lot, and MIP modeling of scheduling problem attracts more and more attention [9, 19, 21-24]. Based on differen modeling ideas, Mehrabad and Fattahi [25] and Shen et al. [22] both proposed a MILP model for FJSP-SDST with the objective of minimizing makespan. Karimi and Ardalan [15] designed two MILP models (a sequence-based model and a position-based model) for FJSP with transportation times. Aimed at minimizing total tardiness, Mousakhani [26] developed a MILP model for FJSP-SDST. With considering different objectives, constraints and modeling ideas, the decision variables and constraint sets of the MILP model vary greatly. Compared with the existing research [15, 22, 25-26], our paper considers the energy consumption objective with turn Off/On energy-saving strategy, and it more complex and novel. Zhang et al. [4] firstly developed a MILP model for FJSP with Turn Off/On strategy, and then Meng et al. [17] proposed five more efficient MILP models based on different modeling ideas. Moreover, Meng et al.[19] designed a MILP model for FJSP with considering worker flexibility and Turn Off/On strategy.

However, MILP model consumes more time and computer memory with the increase of the problem size[27-29]. Therefore, it shall not apply to solve large-scale problems [18]. The approximation methods, especially the meta-heuristic algorithms such as genetic algorithm (GA)[3, 6, 30-31], tabu search(TS) algorithm[22], grey wolf optimization algorithm[32] and virus optimization algorithm(VOA)[33] have proven to be effective for solving the scheduling problems, particularly for large-size problems. To optimize makespan of FJSP-SDST, Shen et al.[22] proposed a tabu search algorithm, and Zhang et al.[34] proposed an improved genetic algorithm. To minimize makespan and the total setup costs of FJSP-SDST, Li et al.[35] designed an elitist non-dominated sorting hybrid algorithm. To minimize energy consumption of FJSP, Zhang et al. designed a Gene Expression Programming (GEP) algorithm to mine dispatching rules. Meng et al.[19] designed a VNS for FJSP with considering worker flexibility and Turn Off/On strategy. For FJSP with controllable processing times (FJSP-CPT), Gong et al.[36] proposed a hybrid GA to simultaneously minimize makespan, worker cost and green objective, Luo et al.[32] designed a multi-objective grey wolf optimization algorithm for simultaneously minimizing makespan and total energy consumption, and Wu and Sun [3] designed a NSGA-II to optimize makespan, energy consumption and the number of Turn Off/On strategy simultaneously.

Reading the relevant literature one can conclude that more and more researchers are paying more and more attention to energy-efficient scheduling both based on MILP formulations [2, 4, 8, 17, 19, 26] and meta-heuristic algorithms[3, 4, 6, 8]. FJSP-SDST-T is more close to actual production workshop; therefore, this research is significant both in theory and application.

III. MILP MODEL FOR FJSP-SDST-T

A. PARAMETERS

Parameters that are used in this paper are presented as below:

i, i'	indices for jobs that are to be machined
n	number of jobs
I	job set and $I = \{1, 2, ..., n\}$
j, j'	index of operations
n_i	total number of operations of job i
J_i	operation set of job i and $J_i = \{1, 2, \cdots, n_i\}$
$J_i^{'}$	operation set of first $n_i - 1$ operations of job i and $J_i^{'} = \{1, 2, \cdots, n_i - 1\}$
k, k'	Indices for machine tools
$O_{i,j}$	the j-th operation of job i
m	number of machines
$m_{i,j}$	number of machines that are able to process operation $O_{i,j}$
K	machine set and $K = \{1, 2, ..., m\}$
$K_{i,j}$	machine set that is able to process operation $O_{i,j}$
$s_{i,i',k}$	setup time when job i' is immediately processed after job i on machine k. Moreover, $s_{i,i',k}$=0 when $i = i'$
t	index for the position of a machine
P_k	maximum number of positions for machine k and $p_k = \sum_{i \in I} \sum_{j \in J_i} x_{i,j,k}$
L_k	position set of machine k and $L_k = \{1, 2, ..., p_k\}$
$L_k^{'}$	set of first $p_k - 1$ positions of machine k and

$L_k' = \{1, 2, \cdots, p_k - 1\}$

T_k required time of Turn Of/On strategy

TB_k break-even time of machine k for implementing Turn Off/On strategy

E_{turn}^k energy consumption for Turn Off/On strategy

N_k maximum allowable times of Turn Off/On strategy of machine k

$x_{i,j,k}$ a binary constant that is equal to 1 if machine k can process operation $O_{i,j}$;0 otherwise

$K_{i,j}$ machine set that is able to process operation $O_{i,j}$

$pt_{i,j,k}$ time of machine k for processing operation $O_{i,j}$

$P_{i,j,k}$ power of machine k for processing operation $O_{i,j}$

$PE_{i,j,k}$ energy consumption of machine k for processing operation $O_{i,j}$

PE total processing energy consumption

IE total idle energy consumption

$IE_{k,t}$ energy consumption of machine k when it is in idle state between position t and $t+1$

P_{idle}^k power consumed by machine k when it is in idle state

P_{set}^k power consumed by machine k when it is in setup state

M a very large positive value, and it is set to 10000 in this paper

CE common energy that is consumed by auxiliary equipments in the workshop

P_t power consumption of the transporter

$t_{k,t}^{idle}$ idle time between position t to $t+1$ of machine k

$set_{k,t}$ setup time between position t to $t+1$ of machine k

$SE_{k,t}$ setup energy consumption between position t to $t+1$ of machine k

$TE_{i,j}$ energy consumption for transporting job i from the machine k that $O_{i,j}$ is processed on to machine k' that $O_{i,j+1}$ is assigned to

$T_{i,j}^r$ transportation time for transporting job i from the machine k that $O_{i,j}$ is processed on to machine k' that $O_{i,j+1}$ is assigned to

$Tt_{k,k'}$ transportation time between machine k and machine k'

P_0 common power that is consumed by auxiliary equipments

The decision variables for the MILP model as given as follows:

$Z_{k,t}$ a binary decision variable that is equal to 1 if Turn Off/On is applied between position t and $t+1$ of machine k ; 0 otherwise

$X_{i,j,k}$ a binary decision variable that is equal to 1 if operation z is processed on machine k ;0 otherwise

$Y_{i,j,k,t}$ a binary decision variable that is equal to 1 if operation $O_{i,j}$ is processed on position t of machine k ; 0 otherwise

$E_{k,t}$ a continuous decision variable that represents for energy consumed by machine k between position t and $t+1$

$EO_{i,j}$ a continuous decision variable that stands for transportation energy consumption $TE_{i,j}$

$S_{k,t}$ a continuous decision variable that denotes the starting time of position t of machine k

$F_{k,t}$ a continuous decision variable that indicates the completion time of position t of machine k

$B_{i,j}$ a continuous decision variable that represents for the starting time of operation $O_{i,j}$

B. PROBLEM DESCRIPTION

The FJSP-SDST-T can be stated as follows: there are a set $I = \{1, 2, \cdots, n\}$ of independent jobs and a set $K = \{1, 2, \cdots, m\}$ of machines. Each job i is with n_i operations $\{O_{i,1}, O_{i,2}, ..., O_{i,n_i}\}$ that must follow processing route. For each operation $O_{i,j}$, it can be machined by a set $K_{i,j}$ of machines. Motivated by practical applications such as changing tool, replacing fixture and positioning that have to be done before the process of another job, sequence and machine dependent setup times are considered. A setup time $s_{i,i',k}$ is needed when operations of jobs i and i' are processed successively on machine k .When an operation of a job is finished, the job needs to be moved to another machine by a transporting vehicle. Therefore, transportation times should be took into consideration. With the objective of minimizing energy consumption, Turn Off/On and postponing strategies (it is detailedly described in following Sections IV.C.4), IV.C.5) and IV.C.6)) are taken into consideration to reduce idle energy consumption.

The problem is to assign each operation to an appropriate machine (machine selection subproblem), to sequence the operations on the machines (operations sequencing subproblem) and to decide whether to apply the Turn Off/On strategy or not when the machine is in idle state (Turn Off/On strategy decision subproblem). Moreover, we take the following assumptions into consideration:

- The number of the transporters is considered to be infinite.
- No more than operations can be simultaneously machined on a machine.
- The different operations of one job cannot be processed simultaneously and must following the given processing route.
- All the jobs, machines and transporters are available at time 0.
- For every operation, preemption is not allowed.
- The power of the transporter is the same for transporting all jobs.
- A transporter can transport at most one job at a time and cannot be interrupted during transportation.
- The magnitude of the transportation times depends on the distance among the machines.

C. TOTAL ENERGY CONSUMPTION OF THE WORKSHOP

In this paper, we divide the energy consumed by the workshop into five parts namely processing energy

consumption (PE), setup energy consumption (SE), idle energy consumption (IE), transportation energy consumption (TE) and common energy consumption (CE)[37], among which PE, SE, and IE are the energy consumed by the machine tools when they are in processing, setup and idle modes respectively. TE is the energy consumed by the transportation systems and CE represents for the energy consumed by auxiliary equipments in the workshop. The total energy consumption (TotalE) is the sum of PE, SE, IE, TE and CE.

1) PROCESSING ENERGY CONSUMPTION

As for processing energy consumption, it is the energy consumption of machines when they are at processing state and can be calculated as,

$$PE = \sum_{i\in I}\sum_{j\in J_i}\sum_{k\in K_{i,j}} P_{i,j,k} pt_{i,j,k} X_{i,j,k} \tag{1}$$

2) SETUP ENERGY CONSUMPTION

Setup energy consumption is the energy consumed by the machine tool for adjustments such as changing tool and replacing fixture[38], and it can be computed as,

$$SE = \sum_{k\in K}\sum_{t\in L_k} SE_{k,t} = \sum_{k\in K}\sum_{t\in L_k} P^k_{set} set_{k,t} \tag{2}$$

where, the setup time $set_{t,k}$ is depended on the jobs that are assigned to the two adjacent positions of a machine, and it can be computed as,

$$set_{k,t} = \sum_{i\in I}\sum_{i'\in I}\sum_{j\in J_i}\sum_{j'\in J_{i'}} Y_{i,j,k,t} Y_{i',j',k,t+1} s_{i,i',k} \tag{3}$$

where, the purpose of binary variable $Y_{i,j,k,t}$ is to decide the operation sequence on each machine. Besides, $Y_{i,j,k,t}$ can play the role of $X_{i,j,k}$ to decide the machine selection subproblem.

3) IDLE ENERGY CONSUMPTION

As to idle energy consumption, it is the energy consumed by the machines in idle mode. In actual production, a machine often waits for jobs due to their late arrivals. This energy consumption is useless and should be reduced as far as possible. The proposed two energy-saving strategies are for reducing it. *IE* can be computed as,

$$IE = \sum_{k\in K}\sum_{t\in L'_k} IE_{k,t} = \sum_{k\in K}\sum_{t\in L'_k} P^k_{idle} t^{idle}_{k,t} \tag{4}$$

where, $t^{idle}_{k,t}$ is the difference of $S_{k,t+1}$, $F_{k,t}$ and $set_{k,t}$, and it can be calculated as,

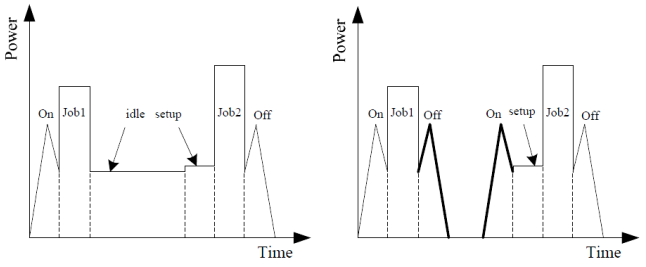
$$t^{idle}_{k,t} = S_{k,t+1} - F_{k,t} - set_{k,t} \tag{5}$$

When a machine is in setup state, it must be kept on so as to do the adjustments. While if a machine stays in idle state for a long time, it economically justifiable to turn it off and then turn it on. Then, the energy consumed in the idle period will be reduced to the energy consumption for Turn Off/On of the machine. A power curve that includes the idle, setup, processing and Turn Off/On states is displayed in Fig. 1. Here, we define an important variable TB_k, for which Turn Off/On is economically justifiable instead of letting the machine run in idle state.

$$TB_k = \max\{T_k, E^k_{turn} / P^k_{idle}\} \tag{6}$$

For the sake of determining the Turn Off/On strategy decision subproblem, we introduce a binary variable $Z_{k,t}$. If the Turn Off/On strategy is implemented, $Z_{k,t} = 1$; otherwise, $Z_{k,t} = 0$. *IE* with considering Turn off/On strategy can be processed as,

$$IE = \sum_{k\in K}\sum_{t\in L_k} ((1 - Z_{k,t}) P^k_{idle} t^{idle}_{k,t} + Z_{k,t} E^k_{turn}) \tag{7}$$



(a) Idle mode **(b) Off\On mode**

FIGURE 1. The power curve of a machine tool with and without Turn Off/On strategy

4) TRANSPORTATION ENERGY CONSUMPTION

With regard to the transportation energy consumption, it is the energy consumed by the transporters for transporting the jobs between different machines. Moreover, it can be defined as,

$$TE = \sum_{i\in I}\sum_{j\in J'_i} TE_{i,j} = P_t \sum_{i\in I}\sum_{j\in J'_i}\sum_{k\in K_{i,j}}\sum_{k'\in K_{i,j+1}} Tt_{k,k'} X_{i,j,k} X_{i,j+1,k'} \tag{8}$$

where, $TE_{i,j}$ is the product of transportation time $T^r_{i,j}$ and the power of the transporter P_t, and it is calculated as below,

$$TE_{i,j} = P_t T^r_{i,j} = P_t \sum_{k\in K_{i,j}}\sum_{k'\in K_{i,j+1}} Tt_{k,k'} X_{i,j,k} X_{i,j+1,k'} \tag{9}$$

5) COMMON ENERGY CONSUMPTION

In actual production, many auxiliary and supporting equipments such as lighting, air conditioning, ventilation and heating are needed to keep the workshop environment, the energy consumed by which is the common energy consumption. *CE* is calculated as below,

$$CE = P_0 C_{\max} \tag{10}$$

6) TOTAL ENERGY CONSUMPTION WITH CONSIDERING TURN OFF/ON STRATEGY

To summarize, the total energy consumption of the workshop with considering Turn Off/On strategy is calculated by equation (11),

D. MILP MODEL

As can be seen from objective function (11), there are four non-linear terms, namely $Z_{k,t} S_{k,t+1}$, $Z_{k,t} F_{k,t}$, $Z_{k,t} Y_{i,j,k,t} Y_{i',j',k,t+1}$ and $X_{i,j,k} X_{i,j+1,k'}$. Thus, the model is hardly non-linear and non-convex. Non-linear models are much more difficult to solve than linear ones. Owing to the existing of many local optical solutions in the feasible solution space of the non-convex models, it is NP-hard to solve their optimal solutions. Therefore, we linearize the nonlinear model by introducing two intermediate variables namely $EO_{i,j}$ and $E_{k,t}$. $EO_{i,j}$ represents for $TE_{i,j}$. $E_{k,t}$ represents for energy between

position t and $t+1$ of machine k, and it is the sum of $SE_{k,t}$ and $IE_{k,t}$.

As can be seen from the following Eqs.(12) and (13), the binary variable $X_{i,j,k}$ is the summary of $Y_{i,j,k,t}$. Therefore, $X_{i,j,k}$ can be removed. Moreover, $F_{k,t}$ is the aggregation of $S_{k,t}$ and $pt_{i,j,k}Y_{i,j,k,t}$, thus, it is redundant and can also be deleted.

$$\sum_{t\in L_k} Y_{i,j,k,t} = X_{i,j,k},\ \forall i\in I, j\in J_i, k\in K_{i,j} \tag{12}$$

$$F_{k,t} = S_{k,t} + \sum_{i\in I}\sum_{j\in J_i}(pt_{i,j,k}Y_{i,j,k,t}), \forall k\in K, t\in L_k \tag{13}$$

1) DECISION VARIABLE

In all, the MILP model includes seven decision variables, among which two ones are binary variables namely $Y_{i,j,k,t}$ and $Z_{k,t}$, and five ones are continuous variables namely $S_{k,t}$, $B_{i,j}$, $C_{\max}$, $EO_{i,j}$ and $E_{k,t}$.

2) OBJECTIVE FUNCTION

$$TotalE = \sum_{i\in I}\sum_{j\in J_i}\sum_{k\in K_{i,j}}\sum_{t\in L_k} P_{i,j,k}pt_{i,j,k}Y_{i,j,k,t} + \sum_{k\in K}\sum_{t\in LL_k} E_{k,t} + \sum_{i\in I}\sum_{j\in JJ_i} EO_{i,j} + P_0C_{\max} \tag{14}$$

In this function, the first term to the last term represent for the total processing energy consumption, the total idle energy consumption, the total transportation energy consumption and the common energy consumption respectively.

3) CONSTRAINT SETS

$$\sum_{k\in K_{i,j}}\sum_{t\in L_k} Y_{i,j,k,t} = 1,\ \forall i\in I, j\in J_i \tag{15}$$

Constraint set (15) restricts that each operation $O_{i,j}$ is performed once and is exactly assigned to one position of an optional machine.

$$\sum_{i\in I}\sum_{j\in J_i} Y_{i,j,k,t} \le 1,\ \forall k\in K, t\in L_k \tag{16}$$

Constraint set (16) states that each position of each machine can execute no more than one operation at the same time.

$$\sum_{i\in I}\sum_{j\in J_i} Y_{i,j,k,t} \ge \sum_{i\in I}\sum_{j\in J_i} Y_{i,j,k,t+1}, \forall k\in K, t\in L_k^{'} \tag{17}$$

Constraint set (17) reveals that the front of the position takes priority to process operation. That is to say, the back position cannot be assigned to an operation if any of the front positions is free.

$$B_{i,j} + \sum_{k\in K_{i,j}}\sum_{t\in L_k}(pt_{i,j,k}Y_{i,j,k,t}) + \sum_{k\in K_{i,j}}\sum_{t\in L_k} Tt_{k,kk}Y_{i,j,k,t} \le B_{i,j+1} + M(1-\sum_{t\in L_{k'}} Y_{i,j+1,k',t}) \tag{18}$$
$$\forall i\in I, j\in J_i^{'}, k'\in K_{i,j+1}$$

Constraints (18) enforces that each operation can only be started when its preceding operation has been finished and transported to current machine.

$$S_{k,t} \le B_{i,j} + M(1-Y_{i,j,k,t}), \forall i\in I, j\in J_i, k\in K_{i,j}, t\in L_k \tag{19}$$

$$S_{k,t} + M(1-Y_{i,j,k,t}) \ge B_{i,j}, \forall i\in I, j\in J_i, k\in K_{i,j}, t\in L_k \tag{20}$$

Constraint sets (19) and (20) altogether force that $B_{i,j}$ is equal to $S_{k,t}$ when $O_{i,j}$ is assigned to position t of machine k. Specifically, if $Y_{i,j,k,t}=0$, both the constraint sets are relaxed; otherwise, they enforce $B_{i,j}$ to be equal to $S_{k,t}$.

$$TB_kZ_{k,t} + \sum_{ii\in I}\sum_{jj\in J_{ii}}(Y_{i',j',k,t+1}s_{i,i',k}) \le S_{k,t+1} - S_{k,t} - \sum_{i\in I}\sum_{j\in J_i}(pt_{i,j,k}Y_{i,j,k,t}) + M(1-Y_{i,j,k,t}) \tag{21}$$
$$\forall i,i'\in I, j\in J_i, j'\in J_{i'}, k\in K_{i,j}\cap K_{i',j'}, t\in L_k^{'}$$

$$S_{k,t} + \sum_{i\in I}\sum_{j\in J_i}(pt_{i,j,k}Y_{i,j,k,t}) \le S_{k,t+1}, \forall k\in K, t\in L_k^{'} \tag{22}$$

$$E_{turn}^kZ_{k,t} + \sum_{i'\in I}\sum_{j'\in J_{i'}}(Y_{i',j',k,t+1}s_{i,i',k}P_{set}^k) \le E_{k,t} + M(1-Y_{i,j,k,t}) \tag{23}$$
$$\forall i,i'\in I, j\in J_i, j'\in J_{i'}, k\in K_{i,j}\cap K_{i',j'}, t\in L_k^{'}$$

$$(S_{k,t+1} - S_{k,t} - \sum_{i\in I}\sum_{j\in J_i}(pt_{i,j,k}Y_{i,j,k,t}) - \sum_{i'\in I}\sum_{j'\in J_{i'}}(Y_{i',j',k,t+1}s_{i,i',k}))P_{idle}^k + \sum_{i'\in I}\sum_{j'\in J_{i'}}(Y_{i',j',k,t+1}s_{i,i',k})P_{set}^k \le E_{k,t} + MZ_{k,t} + M(1-Y_{i,j,k,t}) \tag{24}$$
$$\forall i\in I, j\in J_i, k\in K_{i,j}, t\in L_k^{'}$$

Constraint sets (21), (23) and (24) concurrently demonstrate the Turn Off/On strategy. Constraint set (21) restricts that for two adjacent operations which have been assigned to machine k, the succeeding operation can start only if the precedent operation has been completely finished and the setup operations have been finished. Specifically, if $Z_{k,t}$=1, constraint set (21) ensures that the idle time between position t and $t+1$ of machine k is no less than the breakeven time. If a position of a machine is not assigned to any operation, constraint set (22) guarantees its starting time to be no less than that of its preceding position; otherwise, constraint set (22) is valid inequality and constraint set (21) holds. Constraint sets (23)-(24) work together to ensure the energy consumption between position t and $t+1$ of machine k. To be more specific, if the waiting time is longer than the breakeven time, the machine is turned off and back on ($Z_{k,t}$=1), constraint set (23) shows that the energy consumption $E_{k,t}$ is equal to the sum of E_{turn}^k, and setup energy consumption between position t and $t+1$ of machine k; otherwise, machine k keeps idle and constraint set (24) shows that $E_{k,t}$ is equal to the sum of real idle energy consumption and setup energy consumption.

$$\sum_{t\in L_k} Z_{k,t} \le N_k, \forall k\in K \tag{25}$$

The Turn Off/On method may significantly reduce energy consumption. However, frequently using the Turn Off/On method could shorten the service life of a machine tool, and constraint set (25) aims to limit its maximum allowable times. Transportation energy consumption constraint set:

$$\sum_{k\in K_{i,j}}\sum_{t\in L_k} Y_{i,j,k,t}Tt_{k,kk}P_t \le EO_{i,j} + M(1-\sum_{t\in L_{kk}} Y_{i,j+1,kk,t}), \forall i\in I, j\in JJ_i, kk\in K_{i,j+1} \tag{26}$$

This constraint set shows that the decision variable $EO_{i,j}$ is depended on the machines that $O_{i,j}$ and $O_{i,j+1}$ are assigned to. Constraint set (27) is for restricting makespan.

$$C_{\max} \ge B_{i,S_i} + \sum_{k\in K_{i,S_i}}\sum_{t\in L_k}(pt_{i,S_i,k}Y_{i,S_i,k,t}), \forall i\in I, j\in J_i \tag{27}$$

Constraint set (27) defines that the makespan is no less than the completion times of the last operations of all jobs.

$$B_{i,j}, S_{k,t} \ge 0, \forall i\in I, j\in J_i, k\in K, t\in L_k \tag{28}$$

$$0 \le E_{k,t}, \forall k\in K, t\in L_k^{'} \tag{29}$$

$$0 \le EO_{i,j}, \forall i \in I, j \in J_i^{'} \quad (30)$$

$$X_{i,j,k}, Y_{i,j,k,t}, Z_{k,t} \in \{0,1\}, \forall i \in I, j \in J_i, k \in K, t \in L_k \quad (31)$$

Constraint sets (28)-(30) enforce that the continuous decision variables are positive, and constraint set (31) defines the binary decision variables.

IV. THE PROPOSED HYBRID ALGORITHM FOR FJSP-SDST-T

Meta-heuristic algorithms can be divided into two categories. The first category denotes the swarm intelligent algorithm, which often comes from nature, especially biological systems. For example, GA is inspired by process of natural selection, SFLA is proposed by imitating the behavior of a group of frogs for foraging, ABC is designed by referring to the intelligent foraging behavior of honey bee swarm, particle swarm optimization(PSO) is inspired by the social behavior of bird flocks. The common features of these swarm intelligent algorithms are that they solve a problem by population of candidate solutions. Therefore, they commonly have good ability of global searching and weaker ability of local searching. Being totally different with the first category, the second category, denoted as local search algorithm, only uses solution to solve a problem. These kinds of algorithms start only with one solution and generate new solutions by searching specific neighborhood structures. The most commonly used local search algorithms are VNS, simulate anneal(SA)algorithm and tabu search (TS) algorithm. Therefore, being contrary to swarm intelligent algorithms, local search algorithms poss good ability of local searching and weaker ability of global searching. Therefore, many researchers propose hybrid algorithms that combine swarm intelligent algorithm and local search algorithm for better solving a problem. For example, Li et al.[39] proposed a hybrid algorithm that combines GA and TS for solving classical FJSP with minimizing makespan, Dai et al.[12] combined GA and SA to solve HFSP with minimizing energy consumption, Li et al.[40] combined GA and VNS to solve integrated process planning and scheduling (IPPS) with minimizing makespan and Gao et al.[54] combined GA and VNS for solving classical FJSP with minimizing makespan. In consideration of the good performances of GA and VNS for solving FJSP [3, 19, 39, 41-43], therefore, in this paper, we propose the hybrid algorithm HGA that combines GA and VNS to solve energy conscious FJSP-SDST-T. We insert the VNS into GA to improve its local searching ability. The proposed HGA can make full use of both the advantages of GA and VNS, which can balance the intensification and diversification very well.

A. WORKFLOW OF THE PROPOSED HGA

Fig. 2 shows the flowchart of the HGA. For HGA, eight components are very important namely encoding, decoding, selection, crossover, mutation, VNS, initiation and termination criteria, which will be described in the subsequent sections.

The overall procedure of HGA is given as below:

Step 1: Set the parameters of HGA such as population size (p_s), crossover probability (p_c), mutation probability (p_m), the maximum number of iterations (*maxIter*), the maximum CPU time (*maxTime*) and the maximum number of evaluations (*maxEval*).

Step 2: Initiate the population based on the encoding method and set *Gen*=1.

Step 3: Evaluate all the fitness values of individuals by using the decoding method.

Step 4: Judge whether the termination criteria is satisfied or not. If yes, go to Step8; else, go to Step5.

Step 5: Generate the new population by using selection, crossover and mutation operators.

Step 6: Apply the VNS to top $0.1 \times p_s$ individuals with good fitness.

Step 7: *Gen*=*Gen*+1 and go to Step 3.

Step 8: Output the best solution.

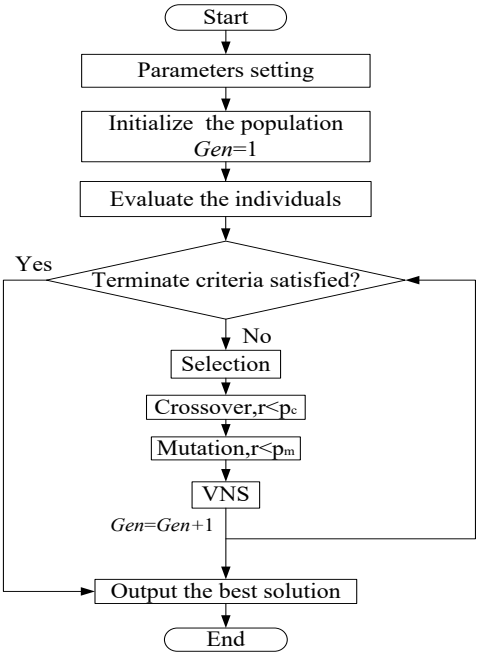


FIGURE 2.The flowchart of the proposed HGA

B. ENCODING SCHEME

Encoding explains how to represent a real solution. Encoding of the individual is very important in GA. In this paper, we use the encoding method used in paper [34]. Chromosomes represent for the solutions of FJSP-SDST-T. The chromosome includes two strings namely operation sequence (OS) string and machine selection (MS) string. The OS string defines all operations of a job with the same symbol and then interprets them according to the sequence of their appearance, the length of which is equal to the total number of operations. The genes of MS string describe the selected machines of the corresponding operations, whose length is also equal to the total number of operations. It is important to note that each element of MS does not

represent for the actual machine number but the index in the matrix of alternative machine set [39, 43].

Fig. 3 shows an example to illustrate the encoding method. With regard to MS string, for example, the machine index of operation $O_{1,4}$ is 3, and it corresponds to the real Machine 6.

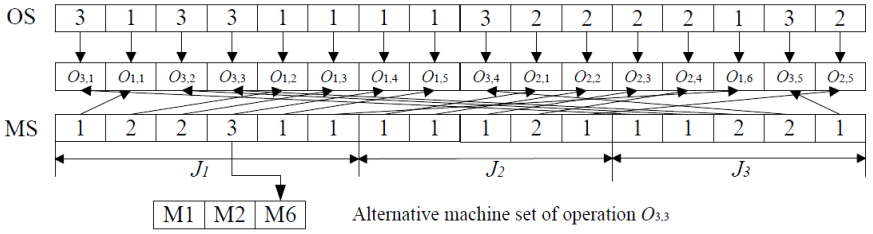


FIGURE 3. The encoding chromosome

C. DECODING SCHEME

Decoding is for transforming a chromosome to a real schedule. As to the same chromosome, with different decoding methods used, different schedules will be obtained. Below are four types of decoding methods, among which the first one is semi-active decoding (SAD), the second one is active decoding (AD), and the third one is energy-conscious greedy decoding (GD). The last type of methods represent for the decoding methods with considering energy-efficient strategies. Thereinto, AD and SAD decoding methods can be designed according to classical FJSP [39]. GD method and decoding methods with considering energy-efficient strategies are specifically designed for the objective of minimizing total energy consumption.

1) SEMI-ACTIVE DECODING (SAD)

In SAD, each operation is assigned to machines in accordance with OS and MS strings. Each operation is assigned after the last operation of its selected machine. The procedures of SAD are as follows:

Step 1: Determine the set of operations for every machine $M_k = \{O_{i,j}\}, k \in K$.

Step 2: Each operation $O_{i,j}$ is allocated after the last operation of its selected machine, of which the allowable earliest starting time $B^*_{i,j}$ can be calculate as,

$$B^*_{i,j} = \max\{E^*_{i,j-1} + Tt_{k^*_{i,j-1},k^*_{i,j}}, E^*_{i',j'} + s_{i',i,k^*_{i,j}}\} \quad (31)$$

where, operation $O_{i',j'}$ is the last operation on machine $k^*_{i,j}$; $k^*_{i,j}$ is the selected machine for operation $O_{i,j}$.

Specifically, if $O_{i',j'}$ is non-existent, in other words, $O_{i,j}$ is the first operation of machine $k^*_{i,j}$, $B^*_{i,j} = \max\{E^*_{i,j-1} + Tt_{k^*_{i,j-1},k^*_{i,j}}\}$; if $O_{i,j}$ is the first operation of job i, $B^*_{i,j} = \max\{E^*_{i',j'} + s_{i',i,k^*_{i,j}}\}$; if $O_{i,j}$ is the first operation of machine $k^*_{i,j}$ and is the first operation of job i simultaneously, $B^*_{i,j} = 0$.

The allowable earliest completion time $E^*_{i,j}$ of operation $O_{i,j}$ is calculated as,

$$E^*_{i,j} = B^*_{i,j} + pt_{i,j,k^*_{i,j}} \quad (32)$$

Step 3: Repeat Steps 2 until all operations are finished.

Step 4: Generate the starting times and completion times of all operations.

2) ACTIVE DECODING (AD)

With regard to AD method, it is different from SAD. AD fully utilizes the idle-time intervals between consecutive operations that have been assigned, and it works by shifting the operations to the left idle-time intervals of a semi-active schedule without delaying other operations. Working steps of AD are described as follows:

Step 1: The same as Step 1 of SAD.

Step 2: Check the idle-time intervals of the machine $k^*_{i,j}$ that $O_{i,j}$ selects, and obtain the idle intervals $[t\_s, t\_e]$. Then, check these intervals in turn. As can be seen from Fig. 4(a), if there is enough time span for allocate $O_{i,j}$ and Eq. (33) is true, the starting time $B^*_{i,j}$ is calculated as Eq. (34); otherwise, check the next interval.

$$\max\{E^*_{i,j-1} + Tt_{k^*_{i,j-1},k^*_{i,j}}, t\_s + s_{i_1,i,k^*_{i,j}}\} + pt_{i,j,k^*_{i,j}} + s_{i,i_2,k^*_{i,j}} \le t\_e \quad (33)$$

$$B^*_{i,j} = \max\{E^*_{i,j-1} + Tt_{k^*_{i,j-1},k^*_{i,j}}, t\_s + s_{i_1,i,k^*_{i,j}}\} \quad (34)$$

where, i_1 and i_2 represent for the two jobs that are on the front and at the back of idle-time interval respectively; j_1 and j_2 are indexes of operations for job i_1 and job i_2 respectively.

Specifically, if operation $O_{i,j}$ is the first operation of job i, $E^*_{i,j-1}$ and $Tt_{k^*_{i,j-1},k^*_{i,j}}$ in Eqs.(33)-(34) are set as 0;if operation $O_{i,j}$ is the first operation of machine $k^*_{i,j}$, $s_{i_1,i,k^*_{i,j}}$ in Eqs.(33)-(34) are set as 0. If all the idle-time intervals have been traversed and no insertable one is found, operation $O_{i,j}$ is allocated at the end of the last operation of machine $k^*_{i,j}$, and $B^*_{i,j}$ is calculated by Eq. (31).

Step 3: Repeat Steps 2 until all operations are finished.

Step 4: Generate the starting times and completion times of all operations.

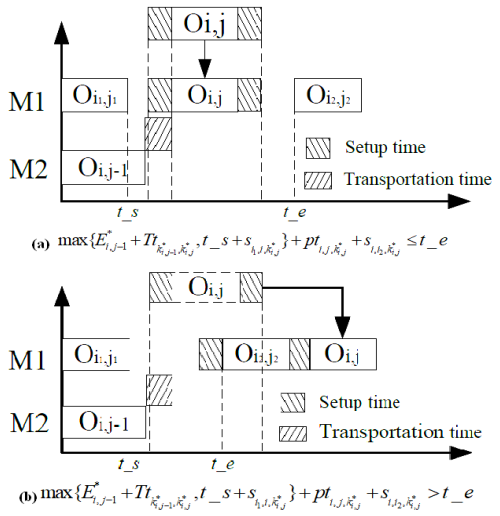


FIGURE 4. Insertable-idle interval is existent and non-existent

3) GREEDY DECODING (GD)

With regard to GD, it uses the OS string alone and the machine selection for each operation is decided in the decoding process with greedy selection for minimizing energy consumption, and its detailed steps are as below:

Step 1: Decide the set of eligible machines $K_{i,j}$ for each operation $O_{i,j}$.

Step 2 (Greedy selection**)**: Starting with the first operation of OS, we try assigning each operation $O_{i,j}$ to all its eligible machines $K_{i,j}$. Afterwards, we select the machine with the least added energy consumption $k^*_{i,j}=\min_{k_{i,j}\in K_{i,j}}\{\Delta TotalE_{O_{i,j},k_{i,j}}\}$. The added energy $\Delta TotalE_{O_{i,j},k_{i,j}}$ of operation $O_{i,j}$ being assigned to machine $k_{i,j}, k_{i,j}\in K_{i,j}$ is computed to Eq.(35). In Eq.(35), operation $O_{i',j'}$ is the last operation on machine $k_{i,j}$; $C_{\max}^{O^*_{i,j},k_{i,j}}$ denotes the current makespan before operation $O_{i,j}$ being assigned to machine $k_{i,j}$ and $C_{\max}^{O_{i,j},k_{i,j}}$ gives the makespan after operation $O_{i,j}$ has been assigned to machine $k_{i,j}$. Specifically, if operation $O_{i,j}$ is the first operation of machine $k_{i,j}$, $\Delta TEC_{O_{i,j},k_{i,j}}$ is calculated to Eq.(36). If operation $O_{i,j}$ is the first operation of job i, $\Delta TEC_{O_{i,j},k_{i,j}}$ is calculated to Eq. (37). If operation $O_{i,j}$ is the first operation of machine $k_{i,j}$ and job i simultaneously, $\Delta TEC_{O_{i,j},k_{i,j}}$ is calculated to Eq.(38).

$$\Delta TotalE_{O_{i,j},k_{i,j}}=\Delta PE_{O_{i,j},k_{i,j}}+\Delta SE_{O_{i,j},k_{i,j}}+\Delta IE_{O_{i,j},k_{i,j}}+\Delta TE_{O_{i,j},k_{i,j}}+\Delta CE_{O_{i,j},k_{i,j}}=$$
$$P_{i,j,k_{i,j}}pt_{i,j,k_{i,j}}+P^{k_{i,j}}_{set}s_{i^*,i,k_{i,j}}+P^{k_{i,j}}_{idle}(\max\{E^*_{i,j-1}+Tt_{k^*_{i,j-1},k_{i,j}},E^*_{i^*,j^*}+s_{i^*,i,k_{i,j}}\}-E^*_{i^*,j^*}-s_{i^*,i,k_{i,j}})+P_tTt_{k^*_{i,j-1},k_{i,j}}+P_0(C^{O_{i,j},k_{i,j}}_{\max}-C^{O^*_{i,j},k_{i,j}}_{\max}) \quad (35)$$

$$\Delta TotalE_{O_{i,j},k_{i,j}}=\Delta PE_{O_{i,j},k_{i,j}}+\Delta TE_{O_{i,j},k_{i,j}}+\Delta CE_{O_{i,j},k_{i,j}}=P_{i,j,k_{i,j}}pt_{i,j,k_{i,j}}+P_tTt_{k^*_{i,j-1},k_{i,j}}+P_0(C^{O_{i,j},k_{i,j}}_{\max}-C^{O^*_{i,j},k_{i,j}}_{\max}) \quad (36)$$

$$\Delta TotalE_{O_{i,j},k_{i,j}}=\Delta PE_{O_{i,j},k_{i,j}}+\Delta SE_{O_{i,j},k_{i,j}}+\Delta CE_{O_{i,j},k_{i,j}}=P_{i,j,k_{i,j}}pt_{i,j,k_{i,j}}+P^k_{set}s_{i^*,i,k_{i,j}}+P_0(C^{O_{i,j},k_{i,j}}_{\max}-C^{O^*_{i,j},k_{i,j}}_{\max}) \quad (37)$$

$$\Delta TotalE_{O_{i,j},k_{i,j}}=\Delta PE_{O_{i,j},k_{i,j}}+\Delta CE_{O_{i,j},k_{i,j}}=P_{i,j,k_{i,j}}pt_{i,j,k_{i,j}}+P_0(C^{O_{i,j},k_{i,j}}_{\max}-C^{O^*_{i,j},k_{i,j}}_{\max}) \quad (38)$$

Step 3: Update the machine selection of MS string for operation $O_{i,j}$.

Step 4: Repeat Steps 2-3 until all operations are finished.

Step 5: Generate the starting times and completion times of all operations.

4) THE ENERGY-SAVING STRATEGIES FOR REDUCING IDLE ENERGY CONSUMPTION

With regard to the energy-saving strategies, they must be implemented after all the starting and completion times of all operations have been decided by using SAD, AD or GD methods. Two strategies namely postponing strategy and Turn Off/On strategy will be described as below.

The first energy-saving strategy is postponing strategy, and its procedures can refer to paper [19]. To implement this strategy, critical operations must be decided. The method of deciding critical operations works as below: let $PJ_{i,j}$ and $SJ_{i,j}$ be the preceding and succeeding operations of operation $O_{i,j}$ in job i. Let $PM^k_{i,j}$ and $SM^k_{i,j}$ be the preceding and succeeding operations of operation $O_{i,j}$ in the same machine k. $S^E_{i,j}, E^E_{i,j}, S^L_{i,j}$ and $E^L_{i,j}$ are the earliest starting time, the earliest completion time, the latest starting time and the latest completion time of operation $O_{i,j}$. Thereinto, $E^E_{i,j}$ is the sum of $S^E_{i,j}$ and $pt_{i,j,k}$. $E^L_{i,j}$ is the sum of $S^L_{i,j}$ and $pt_{i,j,k}$. If $S^E_{i,j}$ is equal to $S^L_{i,j}$, operation $O_{i,j}$ is a critical operation. By using SAD, AD or GD methods, we can get the earliest starting time and earliest completion time of each operation. Besides, makespan $C_{\max}$ is archived. To obtain the latest starting time and the latest completion time of each operation, it can be done as follows:

Step 1: The last operation on each machine cannot be moved. Therefore, we set their latest completion times as their corresponding earliest completion times. The maximum latest completion time of the last operation of all machines is equal to $C_{\max}$.

Step 2: Starting from the last operation to the first operation according to OS string, the latest completion time of each operation except for operations in Step1 can be computed as,

$$E^L_{i,j}=\min\{S^L_{SM^k_{i,j}}-s_{i,ii,k},S^L_{SJ_{i,j}}-Tt_{k,kk}\} \quad (39)$$

where, job ii denotes the job that operation $SM^k_{i,j}$ belongs to; machine kk represents for the machine that operation $SJ_{i,j}$ is assigned to.

Step 3: Repeat Step 2 until the completion times of all operations are decided.

Step 4: Decide the critical operations, whose latest completion time is equal to its earliest completion time.

5) DECODING METHODS WITH CONSIDERING ENERGY-CONSCIOUS STRATEGIES

In this section, we propose four decoding methods with considering postponing strategy and Turn Off/On strategy namely energy-conscious active decoding (ECAD), energy-conscious greedy decoding (ECGD), greedy hybrid decoding (GHD) and random hybrid decoding (RHD). Thereinto, ECAD is obtained by combining AD with energy-saving strategies, and ECGD is obtained by combining GD with energy-saving strategies. More specifically, ECAD is taken as an example, and it is obtained by following steps:

Step 1: Decode a solution with the AD decoding method described in Section 4.3.2.

Step 2: Implement two energy saving strategies described in Section 4.3.4.

As to GHD, it is obtained by hybridizing ECAD and ECGD. Moreover, it progresses by selecting the best method of the two methods, and its steps are given as below:

Step 1: Each chromosome is decoded by both ECAD and ECGD.

Step 2: Fitness values that are obtained by ECAD and ECGD are compared.

Step 3: The decoding method that gets smaller fitness value is choose.

As to RHD, it is obtained by randomly selecting ECAD or ECGD with the same probability.

6) AN EXAMPLE ILLUSTRATION

For the sake of showing the decoding methods more intuitively, an individual is took as an example, whose OS and MS are [3,1,3,3,1,1,1,1,3,2,2,2,2,1,3,2] and [1,2,2,3,1,1,1,1,1,2,1,1,1,2,2,1] respectively. By transforming the elements of MS into actual machine numbers, we can get the actual machine selection string [1,3,6,6,3,3,2,3,1,4,1,2,3,2,3,1]. According to the decoding processes, the Gantt charts obtained by different decoding methods are shown in Fig. 5. In Fig.5, operations in red are critical ones. Fig.5 (a) shows the Gantt chart that is obtained by SAD method. Fig.5 (b) and Fig.5 (d) show the Gantt charts that are obtained by AD and GD methods respectively. As seen from Fig.5(b), operations $O_{2,2}$ and $O_{3,4}$ are inserted at the allowable idle time interval in Machine 3 before operation $O_{1,5}$ .Besides, operation $O_{3,4}$ is inserted at the allowable idle time interval in Machine 1 before operation $O_{2,3}$ when compared with those in Fig.5(a). Obviously, AD can get smaller makespan and idle time than SAD, and thus the total energy consumption is reduced.

As shown in Fig.5 (b), three no-critical operations, namely, $O_{1,1}$, $O_{3,4}$ and $O_{3,5}$, can be postponed. In Fig.5 (d), five no-critical operations namely $O_{1,1}$, $O_{1,2}$, $O_{2,1}$, $O_{3,3}$ and $O_{3,5}$ can be postponed. Fig.5(c) and Fig.5 (e) show the Gantt charts of ECAD and ECGD respectively, in which all the postponable operations have been postponed and Turn Off/On strategy has been applied. For AD, the TotalE, PE, SE, IE, TE, CE are 9329.8, 3856, 130.8, 569, 564 and 4210 respectively. For ECAD, the WE and TotalE are reduced to 134 and 8894.8 respectively. For GD, the TotalE, PE, SE, IE, TE, CE are 7132.2, 2608, 145.2, 705, 474 and 3200 respectively. With regard to ECGD, both the WE and TotalE are reduced by 532 when compared with GD. Above all, postponing strategy and Turn Off/On strategy have proven to be effective in reducing idle energy consumption.

D. Selection operator

In HGA, the role of selection operator is to select the individuals according to the fitness (total energy consumption in this paper). For the purpose of this paper, we adopt two selection operators namely the elitist selection and the binary tournament selection [39, 44]. The elitist selection aims to preserve the individual with the best fitness to the offspring. With regard to binary tournament selection, we randomly select two individuals from the population and select the one with better fitness.

E. Crossover operator

1) CROSSOVER FOR OS STRING

In this paper, precedence operation crossover (POX) has been adopted for OS string, which works as follow:

Step 1: Divide the Job set into two sets namely *Jset*1 and *Jset*2 randomly.

Step 2: Copy the elements in parent P1\P2 that belong to Jset1\ Jset2 to offspring O1\ O2, and preserve their positions.

Step 3: Copy the remaining elements in P2\P1 that are not copied at Step2 to O1\ O2, preserving their order.

Fig. 6(a) shows an example of the POX crossover.

2) CROSSOVER OPERATOR FOR MS STRING

For the MS string, uniform crossover is adopted. With regard to the crossover, firstly, $\sum_{i\in I} n_i$ binary numbers are generated randomly; then, the offspring are generated by swapping all elements of the two parents' strings whose corresponding binary numbers are equal to 1. Because the crossover operator only changes the elements and preserves their order, the offspring are feasible as long as the parents are feasible. Fig. 6(b) shows an example of the uniform crossover.

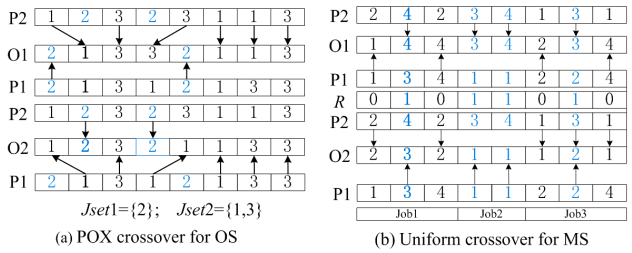
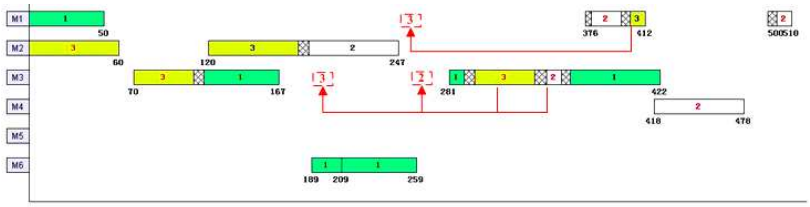
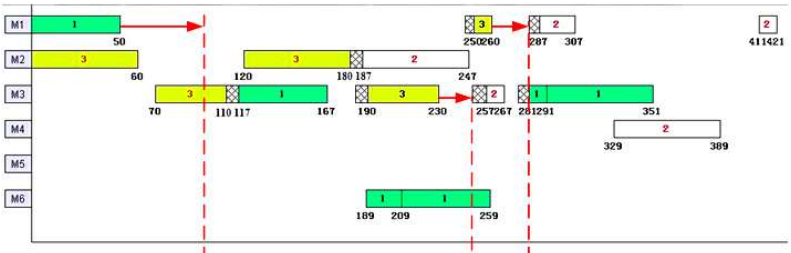


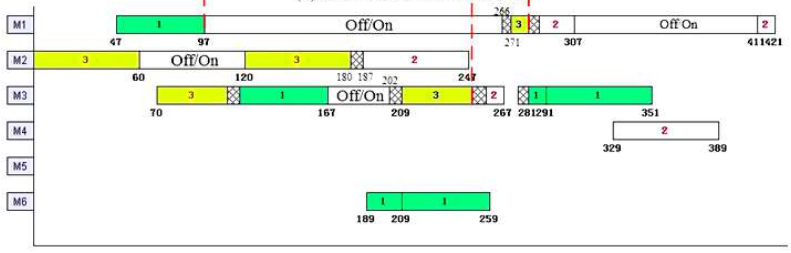
FIGURE 6. POX and uniform crossovers for OS and MS respectively



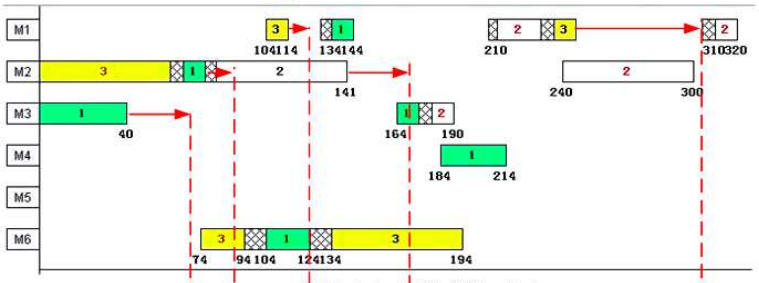
(a) Gantt chart of the SAD method



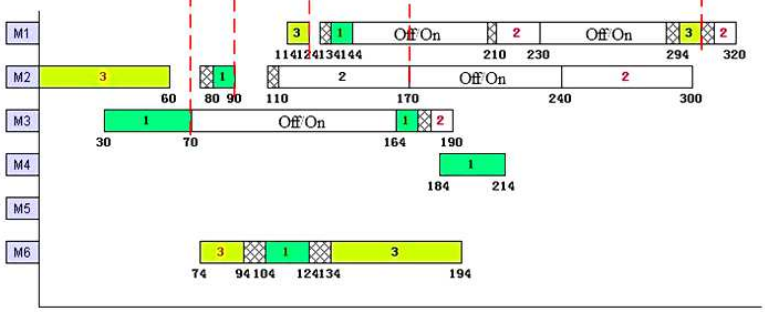
(b) Gantt chart of the AD method



(c) Gantt chart of the ECAD method



(d) Gantt chart of the GD method



(e) Gantt chart of the ECGD method

FIGURE 5. **Gantt charts of different decoding methods**

F. Mutation operator

In this paper, swap mutation and one-point-reassign mutation have been adopted for the OS string and MS string respectively. Swap mutation works by randomly selecting two different positions and exchanging their elements. With regard to one-point-reassign mutation, one position is randomly selected and then its value is changed to other eligible machine.

G. VNS

VNS, as a well-known local search method, works by systematically exploring several different neighborhood structures, and thus local optimal solutions in these neighborhoods are obtained. By comparing these local optima, better solution even the global optimal solution can be archived. In general, VNS is based on three perceptions, which are given as follows [45]:(1) A local optimum of one

neighborhood structure may not be a local one for another neighborhood structure. (2) A global optimum is a local optimum of all possible neighborhood structures. (3) As to many optimization problems, local optima of one or several neighborhoods are relatively close to each other.

For VNS, the design of neighborhoods is very important. In this study, four neighborhood structures are applied to produce new solutions. The first three neighborhood structures namely Swap, Insertion and Reversion are for OS string. The fourth neighborhood structure is Reassign, and it is for MS string.

***N*1(*x*)(Swap):** The same with the Swap mutation.

***N*2(*x*)(Insertion):** Firstly, randomly select two different positions; then, the operation in the second position is moved just before the operation in the first location and the operations between the two positions are moved right accordingly.

***N*3(*x*)(Reversion):** Randomly select two different positions and reverse the operations between them.

***N*4(*x*)(Reassign):** One-point-reassign is used.

Fig.7 shows an example of the four neighborhood structures. The detailed steps of VNS are given as follows:

Step 1(Initialization): Randomly generate the initial solution x and define a set of neighborhood structures $N_k(x), k = 1...k_{\max}$.

Step 2: Repeat the following Steps 3-6 until the stop criteria ($k > k_{\max}$)is satisfied.

Step 3: Set $k = 1$.

Step 4 (Shaking): Randomly produce a solution x' from the *k*th neighborhood of x ($x' \in N_k(x)$).

Step 5 (Local search): Apply some local search method with x' as initial solution. The local search method used in this paper is described as below:

Step 5.1: Set $t = 1$;

Step 5.2: Randomly produce a solution x'' from the *k*th neighborhood of x' ($x'' \in N_k(x')$);

Step 5.3: If x'' is better than solution x', set $x' = x''$ and $t = t + 1$; otherwise, set $t = t + 1$;

Step 5.4: Repeat Step 5.2-5.3 until $t > t_{\max}$.

Step 6 (Updating): If solution x' is better than incumbent solution x, set $x = x'$ and $k = 1$; otherwise, set $k = k + 1$.

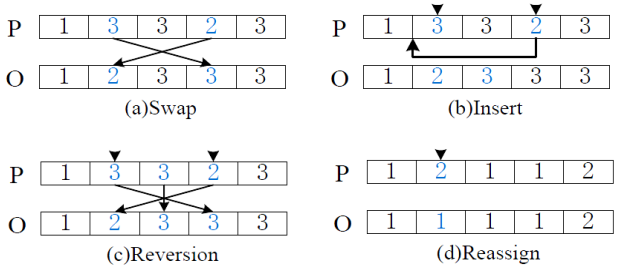


FIGURE 7. Four neighborhood structures of VNS

H. Initiation and termination criteria

With regard to the initiation of the population, it is randomly generated based on the encoding principle. For the terminate criteria, the algorithm terminates when the number of evaluations reaches to the maximum or the CPU time reaches to the timelimit. If the terminate criteria is satisfied, the algorithm ends and the best solution is output.

V. COMPUTATIONAL RESULTS AND DISCUSSIONS

In order to test the MILP and the HGA, two sets of instances namely MFJS01-10[46] and MK01-10[47] are used. We adapt all the instances by adding energy consumption information and magnify the processing times of MK01-10 by 10 times. The setup time $s_{i,i',k}$ is equal to $i + i' + k$ .The processing power is randomly generated within [4, 8]. The idle powers of machines are randomly generated among {1, 2, 3}. The setup power can be obtained by $P_{set}^k = 1.2 * P_{idle}^k$ .The energy consumption of the Turn Off/On for each machine is randomly generated among {10, 30, 60}. The breakeven time of Turn Off/On for each machine is randomly generated among {10, 15, 20}. The Turn Off/On time for each machine is randomly generated among {8, 12, 16}. The transporter power and the common power are set as 3 and 10 respectively. The maximum time of Turn Off/On for each machine is set to 3.

IBM ILOG CPLEX 12.7.1 is used to solve MILP model. The timelimit is set as 3600s. The solving method of CPLEX solver is branch-and-cut method, which is the combination of cutting plane and branch-and-bound method. For all the meta-heuristic algorithms (GA, VNS and HGA), they are all coded in C++. All the methods are run on computer with Win 7 system, Intel(R) Core(TM) 2 Duo CPU of 3.20 GHz processor and 8 GB of RAM memory.

A.THE EFFECTIVENESS OF THE PROPOSED MILP MODEL

In Table 1, Gap is the average optimality gap of the solution, and it can be calculated (CS-BS)*100%/CS. Where, CS denotes the best feasible solution that is obtained within the timelimit, and BS represents for the lower bound obtained within the timelimit. Moreover, a solution is optimal when its Gap is equal to 0. Therefore, Gap value is usually used for judging whether the optimal solution is achieved and evaluating different solutions[23, 48-50]. Moreover, NBVs, NCVs and NCs represent for the number of binary variables, the number of continuous variables, and the number of constraints respectively. Moreover, the CPU time in Table 1 is the time when the optimal solution is proved or the timelimit is reached.

It can be seen from Table 1 that the MILP model only could solve very small-sized instances such as MFJS01-02 to optimality efficiently. However, the solving time exponentially increases with the increasing size of the instance. For MFJS04-10, MK01-02 and MK04-06, it can only archive feasible solutions within 3600s. For MK03 and 07-10, it cannot obtain any feasible solution within 3600s. The reason behind this is NBVs, NCVs and NCs increase sharply when the problem scale becomes larger. The solving

efficiency of the MILP model is inversely proportional to NBVs, NCVs and NCs.

TABLE 1. RESULTS OF MILP MODEL FOR ALL INSTANCES

Problem	NBVs	NCVs	NCs	CPU(s)	TotalE	Gap(%)
MFJS01	212	86	1066	5.71	16234.0[a]	0
MFJS02	263	97	1318	24.24	14787.6[a]	0
MFJS03	393	120	1962	417.99	18386.8[b]	0
MFJS04	515	141	2569	3600(8705.37)	22374.2[b](22069.6 [a])	4.14(0)
MFJS05	503	139	2510	3600	21414.8 [b]	3.11
MFJS06	635	158	3168	3600	25610.2 [b]	2.56
MFJS07	1009	206	5046	3600	35722.6 [b]	20.58
MFJS08	1086	228	5434	3600	41535.8 [b]	18.32
MFJS09	1558	276	7793	3600	53495.6 [b]	25.27
MFJS10	1862	301	9312	3600	60622.0 [b]	24.34
MK01	2732	325	13692	3600	22437.2 [b]	44.26
MK02	9848	581	49108	3600	33068.4 [b]	68.21
MK03	26372	1180	131791	3600	-	-
MK04	4696	502	23556	3600	49061.2 [b]	50.58
MK05	8600	556	43101	3600	132944.8 [b]	66.64
MK06	27980	1261	139822	3600	178419.0 [b]	84.15
MK07	16433	742	82079	3600	-	-
MK08	13507	1066	67858	3600	-	-
MK09	40536	1663	202730	3600	-	-
MK10	53937	1882	269606	3600	-	-

[a] Optimum solution.
[b] Feasible but not optimum solution.
- The MILP model cannot get a feasible solution within 3600s.

B. THE EFFECTIVENESS OF THE PROPOSED HYBRID ALGORITHM

1) PARAMETER SETTINGS

With regard to the $t_{\max}$ of VNS, four settings {5, 10, 20, 30} are tried. Experiment results show that VNS with $t_{\max}=10$ performs better than that with other settings, and $t_{\max}$ is set to 10. HGA has three key parameters namely the population size (p_s), the crossover probability (p_c) and the mutation probability (p_m). To investigate the effect of these parameters on the performance of HGA, the Taguchi method of design of experiment (DOE) is employed to MK03. We set 3 levels for each parameter with an orthogonal array L9 (34) as shown in Fig. 8. For HGA, GHD is used. Moreover, for each combination of parameter values, the HGA is run 20 times independently, each of which uses a total number of 100,000 evaluations. The average total energy consumption is set as the response variable (RV).

Further, the trend of each parameter is figured out in Fig. 8 to show its effect on the performance. According to the result of DOE, the best parameter setting is set to $p_s=100, p_c=0.7, p_m=0.15$.

Experiment number	p_s	p_c	p_m	RV
1	100	0.7	0.05	88474.8
2	100	0.8	0.1	88547.5
3	100	0.9	0.15	88452.1
4	150	0.7	0.1	88891.8
5	150	0.8	0.15	89092.8
6	150	0.9	0.05	89346.2
7	200	0.7	0.15	89211.8
8	200	0.8	0.05	89397.9
9	200	0.9	0.1	89323.6

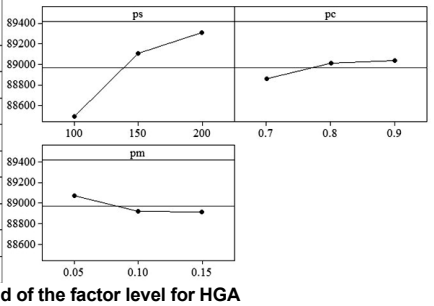


FIGURE 8. The trend of the factor level for HGA

2) THE EFFECTIVENESS OF ENERGY-SAVING STRATEGIES

This section evaluates the effectiveness of the proposed two energy-saving strategies. For this purpose, firstly, 1000 solutions are randomly generated. Then, both AD and the ECAD methods are utilized to obtain the fitness of each solution. In addition, we set the relative percentage increase (RPI) value as the evaluating indicator [19].

$$RPI_i=(D_i-DE_i)/D_i\times 100\%, \forall i=\{1,...,1000\} \quad (39)$$

where, D denotes the fitness archived by AD method, and DE is the fitness obtained by ECAD method.

In Table 2, RPI_Min, RPI_Max and RPI_Ave represent for the minimum RPI, maximum RPI and average RPI of the 1000 individuals. Obviously, Table 2 indicates that the energy-saving strategies perform well for reducing energy consumption, especially for the large-sized problems. This is because the proportion of idle energy consumption increases with the increasing size of the problem.

TABLE 2. COMPARISONS OF AD AND ECAD DECODING METHODS

Problem	MFJS01	MFJS02	MFJS03	MFJS04	MFJS05	MFJS06	MFJS07	MFJS08	MFJS09	MFJS10
RPI_Min	0	0	0	0	0	0	0.35	0.89	0.60	0.94
RPI_Max	11.29	14.52	13.59	12.91	12.22	13.49	12.29	11.71	12.41	15.75

RPI_Ave	2.94	2.68	3.22	3.24	3.73	3.64	4.62	5.27	5.42	5.90
Problem	MK01	MK02	MK03	MK04	MK05	MK06	MK07	MK08	MK09	MK10
RPI_Min	2.51	2.56	4.42	4.49	1.54	7.02	1.63	6.39	5.63	7.26
RPI_Max	15.89	18.50	16.62	16.62	9.84	19.65	11.56	13.01	17.32	17.69
RPI_Ave	9.87	8.57	9.12	10.33	5.11	11.23	5.94	9.12	9.56	10.28

3) COMPARISON RESULTS OF DIFFERENT DECODING METHODS WITH CONSIDERING ENERGY-CONSCIOUS STRATEGIES

So as to evaluate the effectiveness of the four decoding rules, we separately embedded them in the proposed algorithm HGA, namely HGAA, HGAG, HGAH and HGAR, which represent for the hybrid algorithm with ECAD, ECGD, GHD and RHD decoding methods respectively. For HGAA, HGAG and HGAR, the stopping criterion is set as 200,000 evaluations. For GHD method, it uses both ECAD and ECGD once. Thus, for HGAH, the maximum times of evaluation is set to 100,000. For all the instances, the algorithms with different decoding methods run 20 times independently. Moreover, the performance of each decoding methods is measured with the following relative permillage deviation (RPD), which can be calculated as,

$$RPD_b = (TotalE_b - TotalE_{\min}) / TotalE_{\min} \times 1000‰ \quad (40)$$

where, $TotalE_b$ denotes the fitness got by algorithm b and $TotalE_{\min}$ indicates the minimum fitness obtained by all algorithms. The comparative results are reported in Table 3. In Table 3, MRPD, ARPD and SRPD denote the minimum, average and standard variance of RPD in 20 runs, respectively.

Obviously, Table 3 shows that HGAA outperforms HGAG in terms of both the overall MRPD and ARPD values. However, with regard to overall SRPD, HGAG outperforms HGAA. This is because that HGAA can search all over the solution space, which is bigger than what HGAG can search. The bigger solution space means the more possibility to find better solutions. However, bigger solution space may lead to a larger volatility on the solution. Compared with HGAA and HGAG, HGAR and HGAH perform better in terms of the overall MRPD, ARPD and SRPD values. The reason behind this can be explained that HGAR and HGAH can make full use of the superiorities of both ECAD and ECGD. Moreover, HGAH performs the best in terms of the overall MRPD, ARPD and SRPD values. This is because GHD selects the better one of ECAD and ECGD as the final decoding method, while RHD randomly selects ECAD or ECGD. RHD is of more randomicity than GHD.

TABLE 3. COMPARISONS OF DIFFERENT DECODING METHODS

Problem	MRPD				ARPD				SRPD			
	HG_A	HGA_G	HGA_H	HGA_R	HG_A	HGA_G	HGA_H	HGA_R	HG_A	HGA_G	HGA_H	HGA_R
MFJS01	**0.00**	**0.00**	**0.00**	**0.00**	0.06	**0.00**	**0.00**	**0.00**	0.15	**0.00**	**0.00**	**0.00**
MFJS02	**0.00**	**0.00**	**0.00**	**0.00**	0.33	**0.00**	**0.00**	**0.00**	0.94	**0.00**	**0.00**	**0.00**
MFJS03	**0.00**	**0.00**	**0.00**	**0.00**	10.44	**0.00**	**0.00**	0.88	11.83	**0.00**	**0.00**	2.93
MFJS04	**0.00**	**0.00**	**0.00**	**0.00**	5.33	**1.62**	2.99	2.06	5.35	2.63	**1.97**	3.01
MFJS05	**0.00**	**0.00**	**0.00**	**0.00**	0.49	0.25	**0.00**	**0.00**	0.93	0.59	**0.00**	**0.00**
MFJS06	**0.00**	**0.00**	**0.00**	**0.00**	3.31	7.21	**2.23**	5.44	5.84	7.51	**1.28**	6.28
MFJS07	**0.00**	7.25	**0.00**	**0.00**	15.92	10.93	5.03	**1.52**	19.10	4.61	6.06	**3.08**
MFJS08	5.32	0.00	**0.00**	2.42	11.38	5.32	**5.26**	5.38	7.65	6.53	**3.94**	4.56
MFJS09	0.94	6.28	5.96	**0.00**	13.89	13.18	13.86	**8.89**	16.30	9.29	**6.95**	11.48
MFJS10	16.02	8.63	**0.00**	3.16	27.45	14.14	**11.63**	12.27	14.40	**7.06**	12.84	10.69
MK01	3.11	4.12	**0.00**	0.73	15.08	13.74	**7.90**	12.27	14.89	10.55	**9.08**	14.75
MK02	10.28	9.68	7.47	**0.00**	34.77	24.28	16.46	15.37	29.76	17.74	**11.88**	19.32
MK03	4.51	32.64	4.52	**0.00**	24.81	47.76	**10.63**	13.49	21.44	16.04	**6.93**	15.53
MK04	7.41	30.13	**0.00**	6.33	16.16	34.74	**11.24**	15.27	9.46	**5.96**	15.10	12.27
MK05	0.00	14.58	4.74	4.24	10.47	26.68	**8.99**	11.67	11.89	14.90	**4.67**	8.64
MK06	19.89	41.80	**0.00**	9.45	29.96	62.85	**16.67**	27.23	11.07	24.97	**14.49**	20.95
MK07	4.09	23.23	**0.00**	4.71	16.28	32.69	**8.00**	11.17	16.78	12.96	**6.94**	7.15
MK08	**0.00**	8.89	2.77	1.24	6.26	18.53	**6.30**	6.70	8.45	12.57	**5.31**	7.20
MK09	3.72	25.52	**0.00**	6.46	25.42	40.51	**11.60**	18.65	26.64	17.29	**10.18**	13.24
MK10	48.78	13.56	**0.00**	16.19	86.79	32.35	**18.68**	24.20	42.70	25.28	**17.23**	18.71
Mean	6.20	11.32	**1.27**	2.75	17.73	19.34	**7.87**	9.62	13.78	9.82	**6.74**	8.99

Values in bold indicate that they are the best

4) COMPARISON RESULTS OF HGA,GA AND VNS

This section intends to compare HGA with GA and VNS so as to investigate the effectiveness of HGA. For all the algorithms, the GHD decoding method is used. Each algorithm runs 20 times for each instance. GA and VNS use the same stopping criterion of 100,000 evaluations with that of HGA. For fair comparison, the GA adopts the same search operators such as selection, crossover and mutation in HGA. The parameter setting of the GA is adopted as the same with HGA.

Table 4 reports the results of these three algorithms with the same stopping criterion of 100,000 evaluations. Moreover, the CPU time in Table 4 represents for the running time of 20 runs when the stopping criteria is reached. As can be seen from Table 4, HGA performs best among all the three algorithms in terms of the overall MRPD, ARPD and SRPD values. HGA obtains the best results for all instances and generates an overall MRPD of 0 when compared with GA (5.50‰) and VNS (1.95‰), which demonstrates the better converge of HGA than GA and VNS.

Moreover, HGA obtains the minimum SRPD values for all the instances and generates the lowest overall SRPD of 6.64‰ compared with GA (11.57‰) and VNS (9.06‰), which shows better robustness of HGA than the other algorithms. As we can see from Table 4, for all the instances, the maximum solving time of HGA is 85.15s, and it is acceptable. With regard to real instances, the stopping criteria for them should be set according to actual conditions.

With regard to CPU time, we can see that the CPU time consumed by HGA is longer than that consumed by GA and VNS on average. Thus, for fair comparison, we compare these algorithms with the same stopping criteria of CPU time. The stopping criterion is set as the CPU time consumed by HGA with 100,000 evaluations. The comparative results are reported in Table 5. From Table 5, it can be seen that with the same CPU time as stopping criteria, HGA is still significantly better than GA and VNS in terms of the overall MRPD, ARPD and SRPD values.

In conclusion, the above experimental results allow us to conclude that the proposed HGA is more effective and robust than GA and VNS for solving FJSP-SDST-T problem with the objective of minimizing total energy consumption. This is because HGA is the hybrid of GA and VNS, in which VNS is inserted into GA to improve its local searching ability. Thus, HGA has both good global and local searching abilities.

TABLE 4. COMPARISONS OF HGA, GA AND VNS WITH THE STOPPING CRITERIA OF SAME EVALUATIONS

Instances	MRPD			ARPD			SRPD			CPU(s)		
	GA	VNS	HGA	GA	VNS	HGA	GA	VNS	HGA	GA	VNS	HGA
MFJS01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	12.50	**9.94**	12.69
MFJS02	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	13.68	**10.71**	13.98
MFJS03	**0.00**	**0.00**	**0.00**	8.07	**0.00**	**0.00**	9.93	**0.00**	**0.00**	12.60	**11.20**	12.88
MFJS04	0.43	**0.00**	**0.00**	5.54	2.99	**1.51**	5.68	2.39	**1.97**	12.17	**10.77**	12.56
MFJS05	0.00	**0.00**	**0.00**	2.27	**0.00**	**0.00**	2.87	**0.00**	**0.00**	12.05	**11.16**	12.96
MFJS06	5.46	**0.00**	**0.00**	9.33	2.23	**0.88**	4.06	1.71	**1.28**	12.40	**11.37**	13.24
MFJS07	0.00	**0.00**	**0.00**	12.05	5.36	**5.03**	12.44	6.44	**6.06**	13.03	**12.25**	13.38
MFJS08	8.17	**0.00**	**0.00**	13.26	5.26	**4.71**	7.52	5.02	**3.94**	15.38	**13.72**	15.54
MFJS09	1.77	**0.00**	**0.00**	7.99	7.86	**5.57**	8.52	7.09	**6.91**	15.92	**15.77**	16.24
MFJS10	11.92	0.05	**0.00**	24.39	14.65	**11.63**	13.00	17.17	**12.84**	16.69	**15.29**	17.43
MK01	7.21	1.27	**0.00**	15.35	7.90	**7.10**	10.47	9.33	**9.08**	17.42	**17.22**	17.59
MK02	6.91	2.14	**0.00**	24.47	8.93	**7.64**	18.60	12.44	**11.79**	17.26	**16.94**	17.55
MK03	0.84	8.75	**0.00**	15.33	19.68	**6.08**	17.12	13.62	**6.90**	45.74	**43.27**	48.58
MK04	6.64	9.52	**0.00**	17.80	19.99	**11.24**	14.35	15.10	**13.19**	27.62	**26.86**	30.95
MK05	5.85	0.74	**0.00**	17.79	8.24	**4.22**	13.55	9.80	**4.65**	24.59	**23.64**	25.42
MK06	4.06	0.00	**0.00**	26.72	20.34	**16.67**	24.23	23.69	**14.49**	43.78	**42.96**	48.25
MK07	7.22	6.51	**0.00**	21.05	12.27	**8.00**	17.01	7.83	**6.94**	26.20	**25.88**	26.51
MK08	5.00	1.50	**0.00**	9.04	18.06	**3.53**	5.46	20.67	**5.30**	62.35	**63.91**	75.37
MK09	17.99	8.49	**0.00**	29.43	16.18	**11.60**	14.90	10.29	**10.18**	67.76	**68.87**	77.73
MK10	20.62	**0.00**	**0.00**	43.18	18.68	**12.45**	31.63	18.53	**17.23**	77.66	**70.17**	85.15
Mean	5.50	1.95	**0.00**	15.15	9.43	**5.89**	11.57	9.06	**6.64**	27.34	**26.10**	29.70

Values in bold indicate that they are the best

TABLE 5. COMPARISONS OF HGA,GA AND VNS WITH THE STOPPING CRITERIA OF SAME CPU TIME

Instances	CPU	MRPD			ARPD			SRPD		
		GA	VNS	HGA	GA	VNS	HGA	GA	VNS	HGA
MFJS01	12.69	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
MFJS02	13.98	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
MFJS03	12.88	**0.00**	**0.00**	**0.00**	7.68	**0.00**	**0.00**	8.53	**0.00**	**0.00**
MFJS04	12.56	**0.00**	**0.00**	**0.00**	5.54	2.65	**1.51**	5.55	2.22	**1.97**
MFJS05	12.96	**0.00**	**0.00**	**0.00**	2.12	**0.00**	**0.00**	2.69	0.00	**0.00**
MFJS06	13.24	3.69	**0.00**	**0.00**	7.69	2.12	**0.88**	3.95	1.75	**1.28**
MFJS07	13.38	**0.00**	**0.00**	**0.00**	12.24	5.42	**5.03**	12.64	7.25	**6.06**
MFJS08	15.54	6.68	**0.00**	**0.00**	12.96	5.11	**4.71**	6.95	5.62	**3.94**
MFJS09	16.24	**0.00**	**0.00**	**0.00**	7.23	7.23	**5.57**	8.12	7.65	**6.91**
MFJS10	17.43	9.88	**0.00**	**0.00**	22.53	14.98	**11.63**	13.33	16.56	**12.84**
MK01	17.59	7.21	1.27	**0.00**	15.66	7.62	**7.10**	11.23	9.56	**9.08**
MK02	17.55	6.91	2.14	**0.00**	23.55	9.12	**7.64**	17.95	12.12	**11.79**
MK03	48.58	**0.00**	7.56	**0.00**	14.22	17.65	**6.08**	16.35	12.56	**6.90**
MK04	30.95	5.42	7.36	**0.00**	17.12	17.69	**11.24**	14.12	15.25	**13.19**
MK05	25.42	5.32	**0.00**	**0.00**	17.21	8.11	**4.22**	13.68	9.99	**4.65**
MK06	48.25	4.09	**0.00**	**0.00**	23.43	18.56	**16.67**	20.34	21.64	**14.49**
MK07	26.51	7.22	6.51	**0.00**	21.22	12.14	**8.00**	17.43	7.95	**6.94**
MK08	75.37	3.66	1.50	**0.00**	8.12	16.58	**3.53**	4.98	18.56	**5.30**
MK09	77.73	13.95	6.66	**0.00**	25.99	15.42	**11.60**	13.8	10.55	**10.18**
MK10	85.15	15.66	**0.00**	**0.00**	32.48	14.56	**12.45**	26.98	17.68	**17.23**
Mean		4.48	1.65	**0.00**	13.85	8.75	**5.89**	10.93	8.85	**6.64**

Values in bold indicate that they are the best

C. COMPARISON RESULTS OF MILP MODEL AND HGA

Table 6 shows that although HGA can obtain seventeen better solutions than that MILP model can obtain, HGA cannot guarantee to obtain the optimal solution even for small-sized problems such as MFJS01-02. This reflects that the formulation of MILP model is also very important. This is because that optimal solutions obtained by MILP model can be set as the reference standard when one develops the approximation methods such as meta-heuristic algorithms, especially for new scheduling problems.

TABLE 6. COMPARISONS OF MILP MODEL AND HGA

Instances	MILP model		HGA	
	CPU(s)	TotalE	CPU(s)	TotalE
MFJS01	5.71	**16234.0**	12.69	16236.0
MFJS02	24.24	**14787.6**	13.98	14813.6
MFJS03	417.99	**18386.8**	12.88	**18386.8**
MFJS04	3600 (8705.37)	22374.2 (**22069.6**)	12.56	**22069.6**
MFJS05	3600	21414.8	12.96	**21414.8**
MFJS06	3600	25610.2	13.24	**25488.2**
MFJS07	3600	35722.6	13.38	**34999.8**
MFJS08	3600	41535.8	15.54	**40352.4**
MFJS09	3600	53495.6	16.24	**50565.4**
MFJS10	3600	60622.0	17.43	**57848.2**
MK01	3600	22437.2	17.59	**17535.6**
MK02	3600	33068.4	17.55	**15293.4**
MK03	3600	-	48.58	**87009.8**
MK04	3600	49061.2	30.95	**36394.4**
MK05	3600	132944.8	25.42	**68881.2**
MK06	3600	178419.0	48.25	**45862.2**
MK07	3600	-	26.51	**66459.6**
MK08	3600	-	75.37	**233519.0**
MK09	3600	-	77.73	**196334.0**
MK10	3600	-	85.15	**136184.0**

Values in bold indicate that they are the best
CPU time for MILP model is the time when the optimal solution is proved or the timelimit is reached.
CPU time for HGA is mean time of all 20 runs when the HGA is stopped (the stopping criteria of 100,000 evaluations is reached.

VI. CONCLUSIONS AND FUTURE RESEARCH

This study studies the energy-conscious FJSP-SDST-T, and proposes a MILP model as well as an effective hybrid algorithm HGA to solve it. To the best of our knowledge, there are no published papers addressing this problem. We firstly analyze the energy composition of the workshop and formulate a MILP model. Then, the hybrid algorithm HGA that hybridizes GA and VNS is proposed. With regard to the objective of minimizing the total energy consumption, we propose four energy-conscious decoding methods, in which two energy-saving strategies namely postponing strategy and Turn On/Off are specifically designed according to the characteristic of FJSP-SDST-T. Finally, computational experiments of twenty instances are conducted and results show that the MILP model could solve small-sized instances to optimality. The proposed two energy-conscious strategies are very effective in reducing idle energy consumption. Greedy hybrid decoding is proved to be the most effective method. Moreover, HGA outperforms GA and VNS for solving the FJSP-SDST-T with the objective of minimizing total energy consumption.

Our future research will focus on exploration of problem-specific characteristics to develop more effective heuristics for the energy-conscious FJSP-SDST-T such as the hybrid algorithm of GA and TS or SA, and other meta-heuristic algorithms (e.g., grey wolf optimization (GWO) algorithm, tabu search (TS) algorithm, migrating birds optimization (MBO) algorithm, virus optimization algorithm (VOA) algorithm and evolution (DE)). We would mostly welcome related researchers propose more efficient meta-heuristic algorithms for solving the problem in this paper. It is also welcome to design improved MILP model for solving FJSP-SDST-T. Moreover, it will be an interesting topic to study multi-objective FJSP-SDST-T with simultaneously minimizing total energy consumption and makespan.

REFERENCES

[1]Meng L, Zhang C, Ren Y, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2020,142:106347.

[2]Karimi S, Ardalan Z, Naderi B, et al. Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm[J]. Applied Mathematical Modelling, 2017,41:667-682.

[3]Wu X, Sun Y. A green scheduling algorithm for flexible job shop with energy-saving measures[J]. Journal of Cleaner Production, 2018,172:3249-3264.

[4]Zhang L, Tang Q, Wu Z, et al. Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops[J]. Energy, 2017(138):210-227.

[5]Moon J, Park J. Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage[J]. International Journal of Production Research, 2014,52(13):3922-3939.

[6]Yin L, Li X, Gao L, et al. A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem[J]. Sustainable Computing: Informatics and Systems, 2017,13:15-30.

[7]Mouzon G, Yildirim M B, Twomey J. Operational methods for minimization of energy consumption of manufacturing equipment[J]. International Journal of Production Research, 2007,45(18-19):4247-4271.

[8]Meng L, Zhang C, Shao X, et al. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines[J]. International Journal of Production Research, 2019,4(57):1119-1145.

[9]Che A, Wu X, Peng J, et al. Energy-efficient bi-objective single-machine scheduling with power-down mechanism[J]. Computers & Operations Research, 2017,85:172-183.
[10]Liu Y, Dong H, Lohse N, et al. A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance[J]. International Journal of Production Economics, 2016,179:259-272.
[11]Shrouf F, Ordieres-Meré J, García-Sánchez A, et al. Optimizing the production scheduling of a single machine to minimize total energy consumption costs[J]. Journal of Cleaner Production, 2014,67:197-207.
[12]Dai M, Tang D, Giret A, et al. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm[J]. Robotics and Computer-Integrated Manufacturing, 2013,29(5):418-429.
[13]Yildirim M B, Mouzon G. Single-Machine Sustainable Production Planning to Minimize Total Energy Consumption and Total Completion Time Using a Multiple Objective Genetic Algorithm[J]. IEEE Transactions on Engineering Management, 2012,59(4):585-597.
[14]Lin W, Yu D Y, Zhang C, et al. A multi-objective teaching−learning-based optimization algorithm to scheduling in turning processes for minimizing makespan and carbon footprint[J]. Journal of Cleaner Production, 2015,101:337-347.
[15]Karimi S, Ardalan Z, Naderi B, et al. Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm[J]. Applied Mathematical Modelling, 2017,41:667-682.
[16]Adams J, Balas E, Zawack D. The Shifting Bottleneck Procedure for Job Shop Scheduling[J]. Management Science, 1988,34(3):391-401.
[17]Meng L, Zhang C, Shao X, et al. MILP models for energy-aware flexible job shop scheduling problem[J]. Journal of Cleaner Production, 2019,210:710-723.
[18]Zhang B, Pan Q, Gao L, et al. A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019:1-16.
[19]Meng L, Zhang C, Zhang B, et al. Mathematical Modeling and Optimization of Energy-Conscious Flexible Job Shop Scheduling Problem With Worker Flexibility[J]. IEEE Access, 2019,7(1):68043-68059.
[20]Du Y, Li J Q, Luo C, et al. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations[J]. Swarm and Evolutionary Computation, 2021(3):100861.
[21]Che A, Zhang S, Wu X. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs[J]. Journal of Cleaner Production, 2017,156:688-697.
[22]Shen L, Dauzère-Pérès S, Neufeld J S. Solving the flexible job shop scheduling problem with sequence-dependent setup times[J]. European Journal of Operational Research, 2018,265(2):503-516.
[23]Ren Y, Yu D, Zhang C, et al. An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem[J]. International Journal of Production Research, 2017,55(24):7302-7316.
[24]Caumond A, Lacomme P, Moukrim A, et al. An MILP for scheduling problems in an FMS with one vehicle[J]. European Journal of Operational Research, 2009,199(3):706-722.
[25]Saidi-Mehrabad M, Fattahi P. Flexible job shop scheduling with tabu search algorithms[J]. The International Journal of Advanced Manufacturing Technology, 2007,32(5-6):563-570.
[26]Mousakhani M. Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness[J]. International Journal of Production Research, 2013,51(12):3476-3487.
[27] Taghi-nezhad N A, Naseri H, Khalili Goodarzi F, et al. Reactive scheduling presentation for an open shop problem focused on jobs' due dates[J]. Journal of production and operations management, 2015, 6(2): 95-112.
[28]Meng L, Ren Y, Zhang B, et al. MILP Modeling and Optimization of Energy-Efficient Distributed Flexible Job Shop Scheduling Problem[J]. IEEE Access, 2020,8:191191-191203.
[29] Taghi-Nezhad N A. The p-median problem in fuzzy environment: proving fuzzy vertex optimality theorem and its application[J]. Soft computing, 2019, 23(22): 11399-11407.
[30]Liu Y, Dong H, Lohse N, et al. Reducing environmental impact of production during a Rolling Blackout policy – A multi-objective schedule optimisation approach[J]. Journal of Cleaner Production, 2015,102:418-427.
[31]Liu Y, Dong H, Lohse N, et al. An investigation into minimising total energy consumption and total weighted tardiness in job shops[J]. Journal of Cleaner Production, 2014,65:87-96.
[32]Luo S, Zhang L, Fan Y. Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization[J]. Journal of Cleaner Production, 2019,234:1365-1384.
[33]Lu C, Li X, Gao L, et al. An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times[J]. Computers & Industrial Engineering, 2017,104:156-174.
[34]Zhang G, Hu Y, Sun J, et al. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints[J]. Swarm and Evolutionary Computation, 2020,54:100664.
[35]Li Z C, Qian B, Hu R, et al. An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups[J]. Knowledge-Based Systems, 2019,173:83-112.
[36]Gong G, Deng Q, Gong X, et al. A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators[J]. Journal of Cleaner Production, 2018,174:560-576.
[37]Yan J, Li L, Zhao F, et al. A multi-level optimization approach for energy-efficient flexible flow shop scheduling[J]. Journal of Cleaner Production, 2016,137:1543-1552.
[38]Li L, Li C, Tang Y, et al. An integrated approach of process planning and cutting parameter optimization for energy-aware CNC machining[J]. Journal of Cleaner Production, 2017,162:458-473.
[39]Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem[J]. International Journal of Production Economics, 2016,174:93-110.
[40]Li X, Gao L, Pan Q, et al. An Effective Hybrid Genetic Algorithm and Variable Neighborhood Search for Integrated Process Planning and Scheduling in a Packaging Machine Workshop[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems:1-13.
[41]Gao J, Sun L, Gen M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems[J]. Computers & Operations Research, 2008,35(9):2892-2907.
[42]Karimi H, Rahmati S H A, Zandieh M. An efficient knowledge-based algorithm for the flexible job shop scheduling problem[J]. Knowledge-Based Systems, 2012,36:236-244.
[43]Zhang G, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem[J]. Expert Systems with Applications, 2011,38(4):3563-3573.
[44]Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002,6(2):182-197.
[45]Roshanaei V, Naderi B, Jolai F, et al. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan[J]. Future Generation Computer Systems, 2009,25(6):654-661.
[46]Fattahi P, Saidi Mehrabad M, Jolai F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems[J]. Journal of Intelligent Manufacturing, 2007,18(3):331-342.
[47]Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993,41(3):157-183.
[48] Li J, Liu Z M, Li C, et al. Improved artificial immune system algorithm for Type-2 fuzzy flexible job shop scheduling problem[J]. IEEE Transactions on Fuzzy Systems, 2020, doi: 10.1109/TFUZZ.2020.3016225.
[49] Li J Q, Du Y, Gao K Z, et al. A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem[J]. IEEE Transactions on Automation Science and Engineering, 2021, doi: 10.1109/TASE.2021.3062979.
[50] Li J, Han Y, Duan P, et al. Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems[J]. Journal of Cleaner Production, 2020, 250: 119464.