A Hybrid Iterated Greedy Algorithm for a Crane Transportation Flexible Job Shop Problem

Jun-Qing Li, *Member, IEEE*, Yu Du, Kai-Zhou Gao, Pei-Yong Duan, Dun-Wei Gong, *Member, IEEE*, Quan-Ke Pan, and P. N. Suganthan, *Fellow, IEEE*

***Abstract*— In this study, we propose an efficient optimization algorithm that is a hybrid of the iterated greedy and simulated annealing algorithms (hereinafter, referred to as IGSA) to solve the flexible job shop scheduling problem with crane transportation processes (CFJSP). Two objectives are simultaneously considered, namely, the minimization of the maximum completion time and the energy consumptions during machine processing and crane transportation. Different from the methods in the literature, crane lift operations have been investigated for the first time to consider the processing time and energy consumptions involved during the crane lift process. The IGSA algorithm is then developed to solve the CFJSPs considered. In the proposed IGSA algorithm, first, each solution is represented by a 2-D vector, where one vector represents the scheduling sequence and the other vector shows the assignment of machines. Subsequently, an improved construction heuristic considering the problem features is proposed, which can decrease the number of replicated insertion positions for the destruction operations. Furthermore, to balance the exploration abilities and time complexity of the proposed algorithm, a problem-specific exploration heuristic is developed. Finally, a set of randomly generated instances based on realistic industrial processes is tested. Through comprehensive computational comparisons and statistical analyses, the highly effective performance of the proposed algorithm is favorably compared against several efficient algorithms.**

***Note to Practitioners*—The flexible job shop scheduling problem (FJSP) can be extended and applied to many types of practical manufacturing processes. Many realistic production processes should consider the transportation procedures, especially for the limited crane resources and energy consumptions during the transportation operations. This study models a realistic production process as an FJSP with crane transportation, wherein two objectives, namely, the makespan and energy consumptions, are to be simultaneously minimized. This study first considers the height of the processing machines, and therefore, the crane lift operations and lift energy consumptions are investigated. A hybrid iterated greedy algorithm is proposed for solving the problem considered, and several problem-specific heuristics are embedded to balance the exploration and exploitation abilities of the proposed algorithm. In addition, the proposed algorithm can be generalized to solve other types of scheduling problems with crane transportations.**

***Index Terms*— Crane lift operation, energy consumption, flexible job shop, iterated greedy (IG), simulated annealing (SA).**

I. Introduction

THE flexible job shop scheduling problem (FJSP) has been investigated and applied in many realistic industrial applications [1]–[4]. It can be considered as an extended version of the classical job shop scheduling problem, which has been verified to be an NP-hard problem [1]. In the classical FJSP, n jobs have to be processed on m machines. Each job has a given number of operations, which should be processed on a selected machine from a set of suitable machines. Each machine can process only one job at a time, and each job can be processed on only one machine at a time. Preemption is generally not permitted. However, the classical FJSP cannot be directly applied to practical industrial processes because many realistic constraints, including crane transportation, and energy consumptions, have to be considered.

In many realistic production processes, the weight of the jobs should be considered, and therefore, the time and energy consumptions during crane transportation between different machines should be considered. Moreover, energy consumptions during machine processing should also be considered to adapt to green production requirements. However, there is less literature considering FJSP with crane transportation, hereinafter referred to as CFJSP, especially regarding the requirements to lift materials to a certain height.

Therefore, to solve CFJSPs, we propose a hybrid algorithm that combines an iterated greedy (IG) algorithm and simulated annealing (SA). The hybrid algorithm is hereinafter referred to as the IGSA algorithm. The following are the main contributions of this study: 1) for the first time, crane lift operations have been considered in CFJSP, and a mathematical model is proposed; 2) simple yet effective 2-D vectors, namely,

Manuscript received October 6, 2020; revised December 23, 2020; accepted February 26, 2021. This article was recommended for publication by Associate Editor S. Dadras and Editor B. Vogel-Heuser upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation of China under Grant 61773192, Grant 61603169, and Grant 61803192; and in part by the Shandong Key Scientific Innovation Engineering Projects under Grant 2019JZZY020812. *(Corresponding author: Jun-Qing Li.)*

Jun-Qing Li is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China, and also with the School of Computer Science, Liaocheng University, Liaocheng 252059, China (e-mail: lijunqing@lcu-cs.com).

Yu Du and Pei-Yong Duan are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China.

Kai-Zhou Gao is with the Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau.

Dun-Wei Gong is with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China.

Quan-Ke Pan is with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China.

P. N. Suganthan is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2021.3062979.

Digital Object Identifier 10.1109/TASE.2021.3062979

a scheduling vector and a machine-assignment vector, are presented; 3) a problem-specific exploration heuristic, which can balance the exploration abilities and time complexity of the proposed algorithm, is designed; 4) an improved construction heuristic, which can decrease the number of insertion positions by omitting the replicated positions and, therefore, increase the construction efficiency, is proposed; and 5) a simple exploration heuristic and an SA-based acceptance method enhance the global search ability of the proposed algorithm.

The rest of this article is organized as follows. Section II briefly presents a literature review. Section III gives the problem descriptions. Next, the developed IGSA framework and its detailed components are presented in Section IV. To demonstrate the superiority of the proposed algorithm, Section V illustrates the experimental results and compares them with the performance results of several algorithms from the literature. Finally, Section VI presents the concluding remarks and future research directions.

II. Literature Review

In this section, we will briefly review the related literature regarding the scheduling problems with robotic transportations, scheduling problems with crane transportations, and other scheduling problems related to energy consumption and metaheuristics.

A. Scheduling Problems With Robotic Transportations

The robotic transportations in several types of scheduling problems, namely the flow shop problem, hybrid flow shop (HFS) problem, job shop problem (JSP), and FJSP, respectively, are reviewed.

For the robotic transportations in a flow shop scheduling problem, Dawande *et al.* [5] analyzed the development of sequencing and scheduling in robotic cells and classified the problem on the basis of the machine environment, processing restrictions, and objective functions. Sethi *et al.* [6] considered a robotic flow shop scheduling problem in dual gripper robot cell environments and solved it by a practical heuristic algorithm. Kats and Levner [7] studied a special case with a fixed robot route and interval processing times. Che *et al.* [8] proposed an efficient bicriteria algorithm for a stable robotic flow shop scheduling problem. For considering a flow shop scheduling problem with more realistic constraints, the no-wait constraint with robot transportation was considered in [9] and [10]. Meanwhile, job-independent processing time constraints were also investigated in [11]. Other realistic applications, such as a two-robot flow shop [12] and multidegree cyclic flow shop [13], were also studied. However, these studies mainly focused on only one or two factors of flow shop scheduling problems and often neglected the influence of transportation energy consumption.

For solving an HFS problem with robotic transportation, Burnwal and Deb [14] applied a cuckoo search-based approach. Elmi and Topaloglu [15] addressed a robotic problem in blocking HFS cells that considered multiple robots and machine eligibility constraints. Sangsawang *et al.* [16] addressed a two-stage reentrant flexible flow shop problem by the hybridization of a genetic algorithm (GA) and particle swarm optimization (PSO). Batur *et al.* [17] constructed an SA-based heuristic to solve this type of problem. Zabihzadeh and Rezaeian [18] presented a mixed-integer linear programming (MILP) model for an HFS with robotic transportation. However, for green and sustainable development requirements, the energy consumptions of robots should be considered for this type of problem.

For JSP with robotic transportations, Hurink and Knust [19] considered a single transport robot to minimize the sum of all the traveling and waiting times. Furthermore, Hurink and Knust [20] considered a JSP with only a single transport robot and solved it by a tabu search (TS) algorithm. Nouri *et al.* [21] proposed a hybrid metaheuristic approach for a JSP with transportation times and many robots. Liu and Kozan [22] considered a blocking JSP with robotic transportation by a hybrid TS and threshold accepting metaheuristic algorithm. Elmi and Topaloglu [23] studied the cyclic job shop robotic cell scheduling problem and solved it by an ant colony optimization (ACO) algorithm. Ahmadi-Javid and Hooshangi-Tabrizi [24] addressed a ternary-integration scheduling problem that considered transporters in a job-shop environment with a finite number of heterogeneous transporters. However, in realistic productions, flexible manufacturing features and realistic energy consumptions should be simultaneously considered to make the problem more real. Nouri *et al.* [25] proposed a hybrid metaheuristic based on a clustered holonic multiagent model for the robotic transportations in FJSP.

B. Scheduling Problems With Crane Transportations

Crane operations are of many types, including overhead cranes, tower cranes, mobile cranes, and quay crane operations.

For overhead crane optimization problems, Liu *et al.* [26] addressed a novel integrated green scheduling problem consisting of FJSP and overhead crane transportation to minimize the total cost of the comprehensive energy consumption and makespan by using an integrated algorithm. Karimi *et al.* [27] proposed an adaption of the imperialist competitive algorithm (ICA) hybridized by an SA-based local search to solve an FJSP with transportation times. Li *et al.* [29] developed an integrated mathematical model for the simultaneous optimization of both layout and FJSP scheduling, where a single overhead crane transportation was considered. Zhou and Liao [30] considered an FJSP with crane transportation in a baking-free brick machine manufacturing company, and they solved the FJSP by a multiobjective PSO algorithm.

For the tower crane optimization problems, the main challenge is to guarantee that no crane overlap occurs considering the potential crane conflicts. Irizarry and Karan [31] performed a study to find the optimal number of tower cranes and locations on construction sites by using building information modeling (BIM) and geographic information systems (GISs). Marzouk and Abubakr [32] performed investigations to select the tower crane type and an ideal number of tower cranes and locations, as well as to simulate tower crane operations, and they solved the problem by using BIM and

GA. Khodabandelu *et al.* [33] developed a Java-based, agent-based, modeling simulation tool to investigate the effects of dynamic supply selection on the tower crane efficiency.

For problems with the mobile cranes, Kawada *et al.* [34] investigated the controller for a mobile crane and solved it by using a GA algorithm. Peng *et al.* [35] proposed a hybrid multiobjective GA to simulate the interactions between mobile cranes and associated work crews on site. Taghaddos *et al.* [36] developed an auction-based simulation method to solve mobile crane lifting optimization problems in a prefabricated system.

Similar to mobile crane problems, quay crane problems were also investigated. Nam and Lee [37] considered a vehicle routing problem with quay crane scheduling on a mobile harbor platform, and they solved it by a random key-based GA. Zhao *et al.* [38] studied a multiple double-load crane scheduling problem in steel slab yards and solved it by a pointer-based discrete differential evolution algorithm. Tostani *et al.* [39] addressed an integrated class-based storage location assignment and dual shuttle cranes scheduling problem by a bilevel optimization model.

Similar to these works, this study investigates an FJSP with overhead crane transportations, wherein the lift operations and energy consumptions are simultaneously considered.

C. Scheduling Problems Considering Energy Consumptions and Metaheuristics

In recent years, many studies have considered energy consumptions. Bukata *et al.* [40] proposed a novel parallel branch-and-bound algorithm to optimize the energy consumption of robotic cells. Chen *et al.* [41] considered resource-constrained scheduling problems in a multirobot system. Gürel *et al.* [42] studied the tradeoff between the cycle time and the energy consumption of a robot in a robotic cell. Kats and Levner [43] considered a four-machine robotic cell for a no-wait multicyclic scheduling problem.

Moreover, many types of heuristics and metaheuristics were developed and applied for solving different types of optimization problems, such as local-search-based algorithms (including IG [44]–[48], TS [19], [20], and SA [17]) and population-based searching methods (including the artificial bee colony (ABC) algorithm [49]–[51], ICA [27], [52], estimation of distribution algorithm (EDA) [53], and multiobjective optimization algorithms [54]–[56]).

From the above studies, it is evident that many algorithms have been applied in scheduling problems for different environments. However, only a few studies consider crane transportation in FJSPs. Notably, considering the height of devices in FJSPs with crane transportation, many realistic factors need to be considered, such as energy consumption of the machines and crane lift operations. Accordingly, this study investigates an effective hybrid algorithm for an FJSP with crane transportations in a realistic industrial system.

III. Problem Descriptions

In this study, we investigate an FJSP with a single overhead crane device, hereinafter referred to as CFJSP, where there is a set of jobs that has to be processed without preemption on m machines. Each job j has a set of operations, where each operation $O_{i,j}$ should be processed on a suitable machine selected from a candidate set of machines. In addition, a single crane device is present for enabling transport operations between different machines.

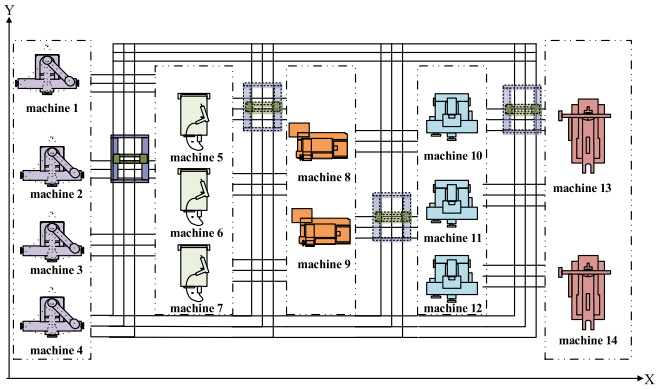

Fig. 1. Machine layout and transportation mode.

Fig. 1 represents an example of a machine layout, where there exist five stages and 14 machines. At each stage, there are different numbers of parallel machines. The crane is used to transport the job to the destination machine. All these machines are fixed at certain locations, and all jobs should be transported according to their given route from stages to stages. The crane should move along with the given route, which is represented by using lines in the figure. For example, given a job that needs to be transported from machines 1 to 6, the route of the crane is given as follows: 1) move horizontally along the X-axis; 2) move horizontally along the Y-axis; and 3) move vertically to the given height of the second slag skimming device.

Four types of tasks should be solved in the problem considered: 1) assigning a suitable machine to each operation; 2) sequencing each assigned job on each machine; 3) deciding the starting and completion times of each operation; and 4) deciding the crane transportation routes.

A. Notations

1) Assumptions:

1) All the jobs are released at time zero and must be processed through all the given stages in the given processing sequence.
2) All the machines are available at time zero and remain continuously available during the entire production horizon.
3) Each job can be processed on one and only one machine selected from the available machines set at a time.
4) Each machine can process only one job at a time.
5) Preemption is not permitted.
6) A single overhead crane is considered to transport the job from one machine to another.
7) For each job, the first operation does not need the crane service.

8) The crane can serve only one operation at a time, i.e., crane service overlapping is not considered.
9) The location of the processing machine for the first operation is the initial location of the crane device.
10) The crane will continuously work without shutting down.
11) There are sufficient numbers of intermediate buffers that can store the completed jobs that are waiting for the crane.
12) The transportation arrival time of the next job must coincide with or be after the idle time of the next machine; otherwise, the crane should go on standby waiting for the idle time of the machine.
13) If the consecutive operations of the same job are processed on the same machine, then obviously no crane is required.
14) There are two types of energy consumptions, i.e., machine processing energy consumption and transportation energy consumption.

2) Notation:

a) Index:

1) m: Total number of machines.
2) n: Total number of jobs.
3) j, $j1$, and $j2$: Index of jobs.
4) i, $i1$, $i2$: Index of operations.
5) k: Index of machines.
6) r: Processing position index of the crane transport.

b) Parameters:

1) θ_j: Number of operations of job j, where $j = 1, 2, \ldots, n$.
2) θ: Total number of operations, where $\theta = \sum_{j=1}^{n} \theta_j$.
3) $O_{i,j}$: ith operation of job j, where $i = 1, 2, \ldots, \theta_j$.
4) $T_{i,j,k}$: Processing time of $O_{i,j}$ on machine k.
5) Ep_k: Processing power of machine k.
6) K^1: Processing machine for the first operation.
7) P_i: Initial position of the crane.
8) $P_{nl}(O_{i,j})$: No-load position of the crane for $O_{i,j}$.
9) $P_l(O_{i,j})$: Load position of the crane for $O_{i,j}$.
10) $Ts_{i,j}$: Starting time of $O_{i,j}$.
11) $Tc_{i,j}$: Completion time of $O_{i,j}$.
12) Tc_r: Completion time of the operation that is processing at position r of the crane transport.
13) ω: Weight value for the two objective values.
14) Wj: Weight of job j.
15) Ws: Weight of the lifting appliance.
16) Wh: Weight of the horizontal device.
17) Wv: Weight of the vertical device.
18) Qn: Lifting weight of the crane.
19) Lx_k: Horizontal coordinate x of machine k.
20) Ly_k: Horizontal coordinate y of machine k.
21) Lz_k: Vertical coordinate z of machine k.
22) Vh: Horizontal operating speed of the crane train.
23) Vv: Vertical operating speed of the crane train.
24) a_{hs}: Horizontal full-load starting acceleration.
25) a_{hb}: Horizontal full-load brake acceleration.
26) a_{vs}: Vertical full-load starting acceleration.
27) a_{vb}: Vertical full-load brake acceleration.
28) P_{oh}: Related horizontal power of the crane.
29) P_{ov}: Related vertical power of the crane.
30) P_{ohs}: Horizontal starting power of the crane.
31) P_{ohb}: Horizontal brake power of the crane.
32) P_{ovs}: Vertical starting power of the crane.
33) P_{ovb}: Vertical brake power of the crane.
34) Ps: Standby power of the crane.
35) E_{mp}, E_{no}, E_{ns}, E_{ls}, E_{lo}, and E_{ct}: Total energy consumption of the machine processing, no-load operation, no-load standby, load standby, load operation, and crane transportation, respectively.
36) $E_{mp}(O_{i,j})$, $E_{no}(O_{i,j})$, $E_{ns}(O_{i,j})$, $E_{ls}(O_{i,j})$, and $E_{lo}(O_{i,j})$: Energy consumptions during the machine processing, no-load operation, no-load standby, load standby, and load operation processes, respectively, of $O_{i,j}$
37) $T_{no}(O_{i,j})$, $T_{ns}(O_{i,j})$, $T_{ls}(O_{i,j})$, $T_{lo}(O_{i,j})$, and $T_{lf}(O_{i,j})$: Processing times of the no-load operation, no-load standby, load standby, load operation, and load lifting, respectively.
38) $T_{hnos}(O_{i,j})$: Horizontal no-load operation starting time of $O_{i,j}$.
39) $T_{hnob}(O_{i,j})$: Horizontal no-load operation braking time of $O_{i,j}$.
40) $T_{hnod}(O_{i,j})$: Horizontal no-load operation duration time of $O_{i,j}$.
41) $T_{hlds}(O_{i,j})$: Horizontal load operation starting time of $O_{i,j}$.
42) $T_{hldb}(O_{i,j})$: Horizontal load operation braking time of $O_{i,j}$.
43) $T_{hldd}(O_{i,j})$: Horizontal load operation duration time of $O_{i,j}$.
44) $T_{vlds}(O_{i,j})$: Vertical load operation starting time of $O_{i,j}$.
45) $T_{vldb}(O_{i,j})$: Vertical load operation braking time of $O_{i,j}$.
46) $T_{vldd}(O_{i,j})$: Vertical load operation duration time of $O_{i,j}$.
47) $\Psi < O_{i1,j1}, O_{i,j} >$: Relationship set when the crane is to serve $O_{i,j}$ after completing the service for $O_{i1,j1}$.
48) $\Theta < O_{i2,j2}, O_{i,j} >$: If $O_{i,j}$ is the immediate successor operation of $O_{i2,j2}$ being processed on the same machine.
49) L: Significantly large number.

c) Decision variables:

1) $X_{i,j,k}$: Binary value set to 1 when $O_{i,j}$ is assigned to machine k; otherwise, $X_{i,j,k}$ is set to 0.
2) $Y_{i1,j1,i2,j2}$: Binary value set to 1 if $< O_{i1,j1}, O_{i2,j2} > \in \Psi$; otherwise, $Y_{i1,j1,i2,j2}$ is set to 0.
3) $Z_{i2,j2,i,j}$: Binary value set to 1 if $< O_{i2,j2}, O_{i,j} > \in \Theta$; otherwise, $Z_{i2,j2,i,j}$ is set to 0.
4) $R_{i,j,r}$: Binary value set to 1 if $O_{i,j}$ is assigned to the r position of the crane transport, where $r1 \leq r \leq \sum_{j=1}^{n} \theta_j - n$.
5) $\mathrm{RM}_{r,k}$: Binary value set to 1 if the operation at the r position of the crane transport is to be processed on machine k; otherwise, $\mathrm{RM}_{r,k}$ is set to 0.

B. Energy Consumption During the Machine Processing

Each machine can process only one job at a time, and each job can be processed on any of the machines available at the current stage. The machine processing energy consumption of

$O_{i,j}$ is calculated by (1). The total machine processing energy consumption of the system is then expressed as (2)

$$E_{mp}(O_{i,j}) = \sum_{k=1}^{m} Ep_k \cdot T_{i,j,k} \cdot X_{i,j,k} \tag{1}$$

$$E_{mp} = \sum_{j=1}^{n} \sum_{i=1}^{\theta_j} E_{mp}(O_{i,j}). \tag{2}$$

C. Energy Consumption During Crane Transportation

Similar to that in [26], the crane transportation process can be divided into four procedures: no-load operation, no-load standby, load standby, and load operation. Suppose that the crane has completed the previous task on machine k1 for operation $O_{i1,j1}$; then, operation $O_{i,j}$ needs the crane transportation, which has the predecessor $O_{i-1,j}$ processed on machine k. Machine k2 is used to process $O_{i,j}$, and an operation $O_{i2,j2}$ is immediately processed before $O_{i,j}$.

1) Energy Consumption of the No-Load Operation: During the transportation procedure, the crane should first move from the current machine k1 to machine k if the two machines are different than each other to fetch operation $O_{i-1,j}$. This procedure is the no-load operation, wherein the crane should first move horizontally and then move vertically to machine k. The transportation time of the no-load operation is divided into three parts, i.e., horizontal no-load starting time, horizontal no-load duration time, and horizontal no-load braking time, which are computed by (4)–(6), respectively. In (7), the total time of the no-load operation is calculated, and the total energy consumption for $O_{i,j}$ during the no-load operation is expressed in (8). Equation (9) is used to calculate of the total energy consumption for all the jobs in the no-load operation procedure

$$\alpha = \frac{Ws + Wh + Wv}{Qn + Wh + Wv} \tag{3}$$

$$T_{hnos}(O_{i,j}) = \alpha \cdot \frac{Vh}{a_{hs}} \tag{4}$$

$$T_{hnob}(O_{i,j}) = \alpha \cdot \frac{Vh}{a_{hb}} \tag{5}$$

$$T_{hnod}(O_{i,j}) = \sum_{k=1}^{m} \sum_{k1=1}^{m} X_{i-1,j,k} \cdot X_{i1,j1,k1} \cdot \left(\frac{|Lx_k - Lx_{k1}|}{Vh} + \frac{|Ly_k - Ly_{k1}|}{Vh} - \frac{Vh}{2} \cdot \alpha \cdot \left(\frac{1}{a_{hs}} + \frac{1}{a_{hb}} \right) \right) \tag{6}$$

$$T_{no}(O_{i,j}) = T_{hnos}(O_{i,j}) + T_{hnob}(O_{i,j}) + T_{hnod}(O_{i,j}) \tag{7}$$

$$E_{no}(O_{i,j}) = T_{hnos}(O_{i,j}) \cdot P_{ohs} + T_{hnob}(O_{i,j}) \cdot P_{ohb} + \frac{Ws}{Qn} \cdot T_{hnod}(O_{i,j}) \cdot P_{oh} \tag{8}$$

$$E_{no} = \sum_{j=1}^{n} \sum_{j1=1}^{n} \sum_{i=2}^{\theta_j} \sum_{i1=1}^{\theta_j} E_{no}\left(O_{i,j}\right) \cdot Y_{i1,j1,i,j}. \tag{9}$$

2) Energy Consumption of the No-Load Standby: If operation $O_{i-1,j}$ is not yet completed on machine k, then the crane should wait in the no-load standby mode. The waiting time for the completion of the previous operation $O_{i-1,j}$ is computed by (10), where the waiting time is the difference between the completion time of $O_{i-1,j}$ and the sum of the starting time of $O_{i1,j1}$ and no-load transportation time. The energy consumption for $O_{i,j}$ during the no-load standby is computed as (11), and the total no-load standby for all the jobs is expressed by (12)

$$T_{ns}(O_{i,j}) = \max\left(Tc_{i-1,j} - Ts_{i1,j1} - T_{no}(O_{i,j}), 0\right) \tag{10}$$

$$E_{ns}(O_{i,j}) = T_{ns}(O_{i,j}) \cdot Ps \tag{11}$$

$$E_{ns} = \sum_{j=1}^{n} \sum_{j1=1}^{n} \sum_{i=2}^{\theta_j} \sum_{i1=1}^{\theta_j} E_{ns}\left(O_{i,j}\right) \cdot Y_{i1,j1,i,j}. \tag{12}$$

3) Energy Consumption of the Load Standby: For safety reasons, the crane should transport the operation at the destination machine k2 only after the completion time of the predecessor operation on machine k2. Therefore, the crane should load operation $O_{i-1,j}$ after its completion time and wait in the load standby status. The horizontal load operation starting time is given by (14). Equation (15) gives the horizontal load operation braking time, while the horizontal load operation duration from the machine processing $O_{i-1,j}$ to the machine that is to process $O_{i,j}$ is expressed by (16). The total load operation time is computed by (17). The three types of times related to the vertical load operation, i.e., vertical load starting time, vertical load breaking time, and vertical duration time, are given by (19)–(21), respectively. The total time to lift the jobs is computed by (22). Equation (23) expresses the load standby time, which is the difference between the completion time of $O_{i2,j2}$ and the sum of the horizontal load operation time to transport the job and the vertical load operation time to lift the job. The load standby energy consumption for $O_{i,j}$ and those for all the jobs are given by (24) and (25), respectively

$$\beta = \frac{Ws + Wj + Wh + Wv}{Qn + Wh + Wv} \tag{13}$$

$$T_{hlds}(O_{i,j}) = \beta \cdot \frac{Vh}{a_{hs}} \tag{14}$$

$$T_{hldb}(O_{i,j}) = \beta \cdot \frac{Vh}{a_{hb}} \tag{15}$$

$$T_{hldd}(O_{i,j}) = \sum_{k=1}^{m} \sum_{k2=1}^{m} X_{i-1,j,k} \cdot X_{i2,j2,k2} \cdot \left(\frac{|Lx_{k2} - Lx_k|}{Vh} + \frac{|Ly_{k2} - Ly_k|}{Vh} - \frac{Vh}{2} \cdot \beta \cdot \left(\frac{1}{a_{hs}} + \frac{1}{a_{hb}} \right) \right) \tag{16}$$

$$T_{lo}(O_{i,j}) = T_{hlds}(O_{i,j}) + T_{hldb}(O_{i,j}) + T_{hldd}(O_{i,j}) \tag{17}$$

$$\gamma = \frac{Ws + Wj + Wv}{Qn + Wv} \tag{18}$$

$$T_{vlds}(O_{i,j}) = \gamma \cdot \frac{Vv}{a_{vs}} \tag{19}$$

$$T_{vldb}(O_{i,j}) = \gamma \cdot \frac{Vv}{a_{vb}} \tag{20}$$

$$T_{vldd}(O_{i,j}) = \sum_{k2=1}^{m} \left(\frac{Lz_{k2}}{Vv} - \frac{Vv}{2} \cdot \gamma \cdot \left(\frac{1}{a_{vs}} + \frac{1}{a_{vb}} \right) \right) \cdot X_{i,j,k2} \tag{21}$$

$$T_{lf}(O_{i,j}) = T_{vlds}(O_{i,j}) + T_{vldb}(O_{i,j}) + T_{vldd}(O_{i,j}) \quad (22)$$

$$T_{ls}(O_{i,j}) = \max\left(Tc_{i2,j2} - T_{lf}(O_{i,j}) - T_{lo}(O_{i,j}), 0\right) \quad (23)$$

$$E_{ls}(O_{i,j}) = T_{ls}(O_{i,j}) \cdot Ps \quad (24)$$

$$E_{ls} = \sum_{j=1}^{n}\sum_{i=1}^{\theta_j}\sum_{k=1}^{m} E_{ls}(O_{i,j}) \cdot X_{i,j,k}. \quad (25)$$

4) Energy Consumption of the Load Operation: Equation (26) expresses the load operation energy consumption for $O_{i,j}$. The energy consumption occurs in the following two parts. The first energy consumption occurs during the horizontal load operation starting time, braking time, and transportation duration at a general speed. The second energy consumption occurs during the vertical load operation starting time, braking time, and lift duration. Equation (27) expresses the load operation energy consumption for all the jobs

$$E_{lo}(O_{i,j}) = T_{hlds}(O_{i,j}) \cdot P_{ohs} + T_{hldb} \cdot P_{ohb} + T_{vlds} \cdot P_{ovs} + T_{vldb} \cdot P_{ovb} + \left(\frac{Ws + Wj}{Qn}\right) \cdot (T_{vldd} \cdot P_{ov} + T_{hldd} \cdot P_{oh}) \quad (26)$$

$$E_{lo} = \sum_{j=1}^{n}\sum_{i=1}^{\theta_j}\sum_{k=1}^{m} E_{lo}(O_{i,j}) \cdot X_{i,j,k}. \quad (27)$$

5) Total Energy Consumption of Crane Transportation: Considering the four types of transportation energy consumptions, the total energy consumption during the entire transportation process can be calculated as

$$E_{ct} = E_{no} + E_{ns} + E_{ls} + E_{ld}. \quad (28)$$

D. Formulation of CFJSP

$$\min f = \omega \cdot f_1 + (1 - \omega) \cdot f_2 \quad (29)$$

$$\min f_1 = \max(Tc_{\theta j,j}),\ j = 1, 2, \ldots, n \quad (30)$$

$$\min f_2 = E_{mp} + E_{ct} \quad (31)$$

$$Tc_{i-1,j} + T_{ls}(O_{i,j}) + T_{ns}(O_{i,j}) \le Ts_{i,j} \quad \forall j, 1 < i < \theta_j \quad (32)$$

$$Tc_{i,j} \ge Tc_{i2,j2} + T_{i,j,k} - L \times (3 - X_{i,j,k} - X_{i2,j2,k} - Z_{i2,j2,i,j}) \quad \forall k, i, j, i2, j2 \quad (33)$$

$$\sum_{k=1}^{m} X_{i,j,k} = 1 \quad \forall j, i \quad (34)$$

$$P_i = K^1 \quad (35)$$

$$P_{nl}(O_{1j}) = P_l(O_{1,j}) = P_l(O_{i1,j1}) \quad \forall < O_{i1,j1}, O_{1,j} > \in \Psi \quad (36)$$

$$P_{nl}(O_{i,j}) = P_l(O_{i,j}) = P_l(O_{i1,j1}) \quad \forall < O_{i1,j1}, O_{i,j} > \in \Psi \wedge (X_{i,j,k} = X_{i-1,j,k}) \quad \forall k, i, j, i1, j1 \quad (37)$$

$$\sum_{j=1}^{n}\sum_{i}^{\theta_j} R_{i,j,r} = 1 \quad \forall r \quad (38)$$

$$\sum_{r=1}^{\theta} R_{i,j,r} = 1 \quad \forall j, i \quad (39)$$

$$\sum_{r=1}^{\theta} \mathrm{RM}_{r,k} = 1 \quad \forall k \quad (40)$$

$$Tc_r \le Tc_{i,j} + L \times (1 - R_{i,j,r}) \quad \forall i, j, r \quad (41)$$

$$Tc_{i,j} \le Tc_r + L \times (1 - R_{i,j,r}) \quad \forall i, j, r \quad (42)$$

$$\mathrm{RM}_{r,k} \ge R_{i,j,r} - L \times (1 - X_{i,j,k}) \quad \forall i, j, r, k \quad (43)$$

$$X_{i,j,k} = \begin{cases} 1, & \text{if } O_{i,j} \text{ is processed on machine } k \\ 0, & \text{otherwise} \end{cases} \quad \forall i, j, k \quad (44)$$

$$Y_{i1,j1,i2,j2} = \begin{cases} 1, & \text{if } < O_{i1,j1}, O_{i2,j2} > \in \Psi \\ 0, & \text{otherwise} \end{cases} \quad \forall i1, j1, i2, j2 \quad (45)$$

$$Z_{i2,j2,i,j} = \begin{cases} 1, & \text{if } < O_{i2,j2}, O_{i,j} > \in \Theta \\ 0, & \text{otherwise} \end{cases} \quad \forall i2, j2, i, j. \quad (46)$$

Constraint (III-D) describes the total objective of this problem, which is to minimize the weighted sum of two objectives, i.e., the makespan in (30) and total energy consumption in (31). Constraint (32) guarantees the processing sequence relation among the consecutive operations of the same job, i.e., the starting time of the successor operation must be after the completion time of the predecessor operation plus the crane transport and lift times. The processing relation between two immediate jobs on the same machine is ensured in constraint (33). Constraint (34) ensures that only one machine can be selected by each operation. Constraint (35) guarantees that the initial position of the crane is equal to the location of the processing machine for the first operation. Constraint (36) describes that noncrane services for the first operation of each job. Constraint (37) denotes that noncrane services for a job whose two operations are consecutively processed on the same machine. Constraint (38) ensures that any position of the crane transport can be occupied by only one operation. Constraint (39) guarantees that one operation can select only one transport position of the crane. Constraint (40) ensures that any position of the crane transport can select only one machine. Constraints (41) and (42) ensure that the completion time of any position of the crane transport should be equal to the completion time of the processing operation. Constraint (43) guarantees the corresponding relationship between the processing operation and the crane transport position. Constraints (44)–(46) show the range of three binary decision variables.

IV. Methodology

This section presents the hybrid IGSA algorithm used to solve the CFJSP problems considered. First, the encoding, decoding, and problem-specific destruction and construction heuristics are presented. Subsequently, the main framework of the proposed IGSA algorithm is described.

A. Representation and Encoding

For solving the CFJSP problems considered, each solution in the proposed algorithm is represented by a 2-D vector. The first-dimensional vector is named the scheduling vector with a length equal to the total number of operations $\Pi = \{\pi_1, \pi_2, \ldots, \pi_\theta\}$. Each element of the scheduling vector

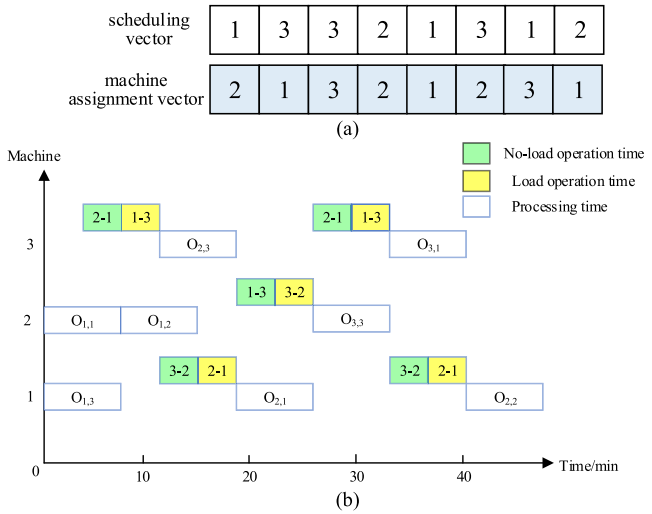

Fig. 2. Solution representation. (a) Encoding. (b) Decoding Gantt chart.

π_i is represented by a job number, and the sequence of each job is also the relative processing order. The second-dimensional vector is named the machine assignment vector $\Xi = \{\delta_1, \delta_2, \ldots, \delta_\theta\}$, where each element δ_i is represented by using a machine number. Similar to the scheduling vector, the length of the machine assignment vector is also equal to the total number of operations. Fig. 2(a) shows an example of solution representation, wherein there are three jobs numbered 1, 2, and 3. The total numbers of the operations for these three jobs are 3, 2, and 3, respectively. The machine assignment vector tells the machine assigned for the corresponding position of operations. For example, the first operation of job 1 is assigned to machine 2, the first operation of job 3 is assigned to machine 1, and the last operation of job 2 is scheduled on machine 1.

From the solution representation, we can see that two tasks of the problem considered have been completed, i.e., to assign a suitable machine to each operation and to sequence each assigned job on each machine. Fig. 2(b) presents the Gantt chart for the example, wherein each operation is represented by a rectangle labeled starting with "O" followed by two numbers. For example, the first operation that is processing on machine 3 is $O_{2,3}$, which represents the second operation of job 3. The two rectangles on the top right-hand corner of $O_{2,3}$ represent the two movements of the crane, where the first movement is from machines 2 to 1 and the second from machines 1 to 3.

B. Decoding Heuristic

After completing the two tasks by the encoding method, another two tasks, i.e., how to decide the starting and completion times of each operation and how to decide the crane transportation routes, are to be completed by a decoding heuristic. The detailed procedure of the decoding heuristic is given as follows.

To decide the starting and completion times of each operation, we should assign and schedule each operation from left to right one by one. To schedule each operation, we should consider four key factors, i.e., idle time of the crane, idle time of the predecessor operation of the same job, idle time of the processing machine, and crane movement route. The first three idle times are simple to decide, while the movement route of the crane should be carefully estimated.

The movement routes of the crane are of several types. The first four types are similar to that in [26], which includes no-load operation, no-load standby, load operation, and load standby. The no-load operation phase is to move the crane from the current position to the machine that is processing the predecessor operation of the same job. The no-load standby means to wait for the completion of the predecessor operation. The load standby means to wait for the idle time of the destination machine while loading the predecessor operation. The load operation is to move the crane from the machine that is processing the predecessor operation to the destination machine to process the current operation. However, different from [26], we also consider the crane lifting movement to take into consideration the different heights of machines. Therefore, this is the first study to consider the crane lifting operation, which is to move the crane to a certain height.

For the example solution shown in Fig. 2(a), the corresponding Gantt chart is shown in Fig. 2(b). As evident from the figure, to transport operation $O_{2,3}$, the crane should first move from machines 2 to 1 to load the predecessor operation $O_{1,3}$ and then from machines 1 to 3 to transport operation $O_{1,3}$ to the destination machine 3. Subsequently, the crane should move from machines 3 to 2 and from machines 2 to 1 for serving operation $O_{2,1}$. Therefore, the complete crane movement routes are given as follows: {⟨2-1⟩, ⟨1-3⟩, ⟨3-2⟩, ⟨2-1⟩, ⟨1-3⟩, ⟨3-2⟩, ⟨2-1⟩, ⟨1-3⟩, ⟨3-2⟩, ⟨2-1⟩}. In addition, different colors for the crane movement represent different crane movement types. For example, green represents the no-load movement, and yellow denotes the load operation movement.

C. Problem-Specific Mutation Heuristic

To perform a local search around the given solution, the mutation operator is commonly used heuristic. Considering the CFJSP problems in this study, there are two vectors in a given solution. In addition, the problem of generating a neighboring solution while considering both the scheduling and machine assignment vectors should be solved. We propose a simple method to randomly select one of the following five types of mutation operators to generate a neighboring solution.

1) Two-Point Reverse (TPR) Operator: It generates a neighboring solution by reversing a selected segment. Fig. 3(a) describes the procedure of the TPR operator, and the detailed steps are given as follows:

Step 1: Randomly generate two positions p_1 and p_2 in the scheduling vector.

Step 2: Record into a set named *MS* the processing machine for each operation between p_1 and p_2 in the scheduling vector

Step 3: Reverse the elements between p_1 and p_2 in the scheduling vector.

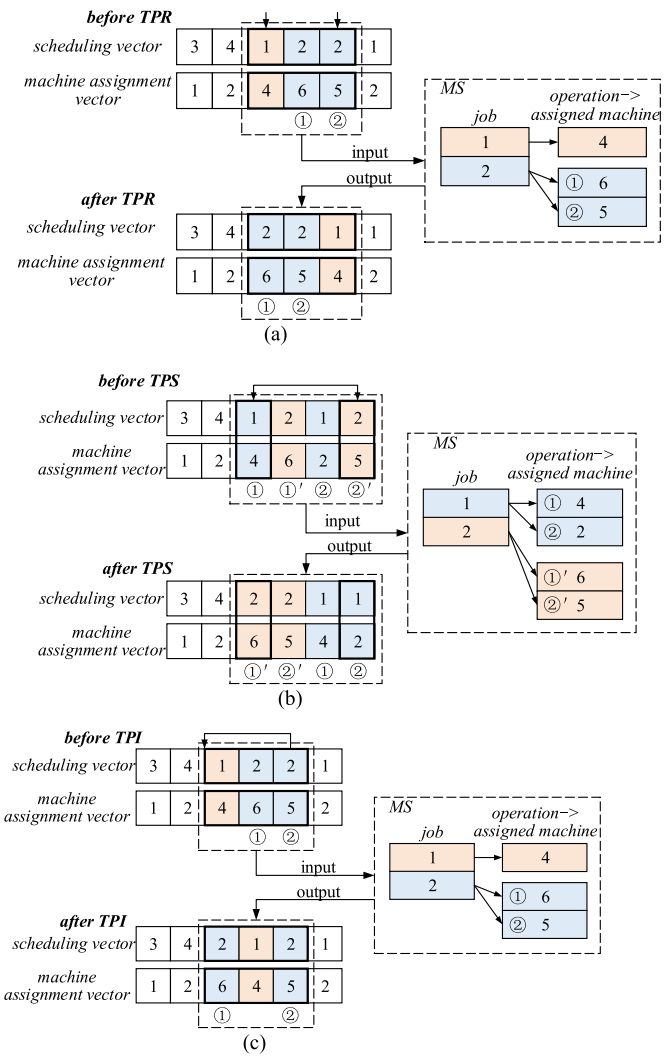

Fig. 3. Problem-specific mutation heuristic. (a) TPR operator. (b) TPS operator. (c) TPI operator.

Step 4: In the machine assignment vector, from position p_1 to p_2, replace the element with the assigned machine in *MS* for each corresponding operation in the scheduling vector.

2) Two-Point Swap (TPS) Operator: It generates a neighboring solution by swapping two selected jobs. Fig. 3(b) describes the procedure of the TPS operator. The main difference between TPS with TPR is in Step 3, wherein TPS swaps the two jobs at the two positions p_1 and p_2 in the scheduling vector.

3) Two-Point Swap With Random Machine (TPSM) Operator: It is similar to the TPS operator, except for the machine vector between the two selected positions p_1 and p_2, TPSM will select a randomly available machine for each corresponding operation.

4) Two-Point Insertion (TPI) Operator: It generates a neighboring solution by inserting one job before the position of another selected job. Fig. 3(c) describes the procedure of the TPI operator. The main difference between TPI with TPR is in Step 3, wherein the TPI operator inserts the job at position p_2 before position p_1 in the scheduling vector.

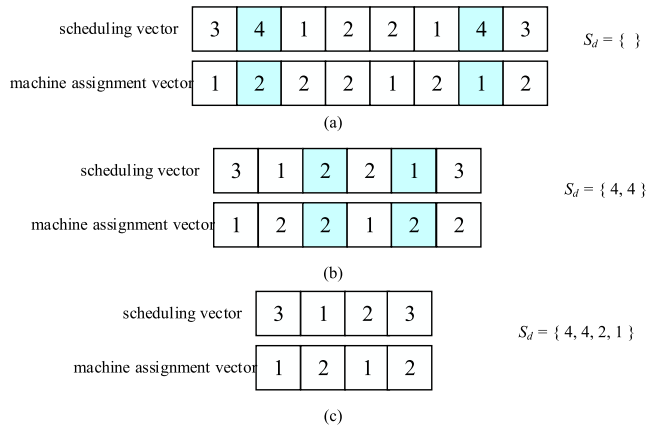

Fig. 4. IGSA Destruction heuristic. (a) Initial solution vectors. (b) Solution vectors after deleting $J_{\max}$. (c) Solution vectors after the destruction heuristic.

5) Two-Point Insertion With Random Machine (TPIM) Operator: It is similar to the TPI operator, except for the machine vector between the two selected positions p_1 and p_2, TPIM will select a randomly available machine for each corresponding operation.

D. IGSA Destruction Heuristic

The destruction procedure is to delete several jobs from the solution representation. A common way to delete jobs is to randomly select a certain number of jobs and delete them from the 2-D vectors. However, this random way does not consider any information regarding differences among the jobs. It is obvious that jobs with the maximum completion times are generally critical jobs and, hence, affect the makespan of the solution; therefore, these critical jobs should be first considered to be deleted from the current solution. In this study, similar to that in [46], we propose a novel destruction heuristic as in Algorithm 1.

Given an example with four jobs and two machines, the corresponding scheduling and machine assignment vectors are given in Fig. 4(a). The destruction length is $d = 4$. All the jobs are sorted according to the decreasing order of their completion times, and the result is recorded in a set as {4, 3, 1, 2}. Subsequently, job 4 is selected, and the total number of operations of job 4 is 2. Therefore, $d_1 = \min(d/2, \theta_j) = 2$ and $d_2 = 2$. Subsequently, from right to left, the two operations of job 4 are deleted and placed in set S_{d}. Fig. 4(b) shows the resultant status. The second loop procedure is to randomly delete d_2 number of selected jobs. For example, in Fig. 4(b), positions 3 and 5 are selected, and the corresponding jobs in these selected positions are deleted and placed in set S_{d}. Fig. 4(c) shows the solution vectors after implementing the entire destruction heuristic.

E. IGSA Construction Heuristic

In the canonical IG algorithm, the construction procedure is to test the best insertion position for each deleted job. For solving the considered CFJSP problem, the two main challenges of the insertion heuristic are as follows: 1) how to

Algorithm 1 Destruction Heuristic

input: current solution; destruction length d
output: a set of deleted jobs named S_d; the remaining part named S_r

Sort all jobs according to the decreasing order of their completion time
Select the first job j with the maximum completion time
Let θ_j be the total number of operations of job j. Let $d_1 = \min(d/2, \theta_j)$. Let $d_2 = d\text{-}d_1$. Let $count = 0$.
While $count < d_1$ **do**
    From right to left in the scheduling vector, find the first job equals to job j, let P_j be the first found position.
    Delete the element at the position P_j from both the scheduling vector and machine assignment vector
    Put the job number j into S_d
    Increase $count$
end
Let $count = 0$
While $count < d_2$ **do**
    Randomly select a position r_1 in the scheduling vector, Record the job at the selected position r_1.
    Put the job number j into S_d
    Delete the element at the position r_1 from both the scheduling vector and machine assignment vector
    Increase $count$
end

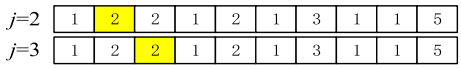

Fig. 5. Insertion example.

select the best insertion position without any replication and 2) how to remain the machine assignment after the insertion.

1) To overcome the first challenge, i.e., to delete the replicated insertion positions with the same scheduling sequence, we propose the following lemma.

Lemma 1: Given a solution with scheduling vector $\Pi = \{\pi_1, \pi_2, \ldots, \pi_i, \ldots, \pi_n\}$ and machine assignment vector $\Xi = \{\delta_1, \delta_2, \ldots, \delta_i, \ldots, \delta_n\}$, to insert π_i into all the possible positions in Π, we should not consider the position j where $\pi_j = \pi_i$, and the total number of replicated positions to be omitted when inserting π_i is $|\Pi_{\pi i}|$, where $\Pi_{\pi i}$ denotes the set where all the elements are equal to π_i.

Proof: Fig. 5 shows an insertion example, where $\pi_i = 2$ is to insert all the possible positions. When π_i is inserted in the second position, i.e., $j = 2$, where the inserted π_i becomes the first operation of job 2. However, considering the insertion position j = 3, where the inserted π_i becomes the second operation of job 2, and the resultant solution is the same as that obtained with the second-position insertion process. Therefore, the insertion position before and after the element equals to π_i will obtain the same solution and should omit one of them.

In addition, let $\Pi_{\pi i}$ denote the set of elements equal to π_i and $|\Pi_{\pi i}|$ the length of the set. It is obvious that the total number of positions to be omitted is $|\Pi_{\pi i}|$. Hence, the lemma is proved.

2) To solve the second problem, i.e., to maintain the machine assignment after the insertion, we propose a machine assignment remaining construction heuristic. The main concept of this heuristic is to assign a suitable machine for the insertion operation and record the assigned machines for other operations, as the insertion of one job may change the operation number for several subsequent jobs.

Algorithm 2 Construction Heuristic

input: S_r and S_d
output: a feasible solution

Record the assignment machine for all operations in the initial solution.
While $|\ S_d\ | > 0$ **do**
    Select the first job j in S_d, and delete it from S_d.
    Select all operations of job j, and construct the set Π_j (cf. Lemma 1)
    For each position except Π_j in S_r **do**
        Test all the positions in S_r to insert job j
        For each insertion position, find the best machine with the minimum partial objective for the inserting job j
        Adjust the machine assignment vector for the following operations
    end
    Find the best insertion position for job j, and the update the scheduling and machine assignment vector after inserting job j
end
Output the feasible solution
end

Algorithm 2 describes the proposed construction heuristic. Fig. 6 shows an example to describe the steps of the construction heuristic. Fig. 6(a) shows the solution vectors after implementing the destruction heuristic. The first operation number to be inserted is job 4, which has not occurred in the current scheduling vector and Π_4 is empty. All the possible positions in the scheduling vector should be attempted. Fig. 6(b) shows the resultant solution vectors after inserting the first operation of job 4. For the second operation of job 4, $\Pi_4 = \{2\}$, and the second position should not be tested. Fig. 6(c) shows the solution vectors after inserting the second operation of job 4. Fig. 6(d) shows the resultant solution vectors after applying the construction heuristic. Notably, the machine assignment should be maintained to avoid violating the machine availability constraints.

It is obvious that the time complexity of the destruction heuristic is $O(n^2)$, and that of the construction heuristic is $O(n^3m)$.

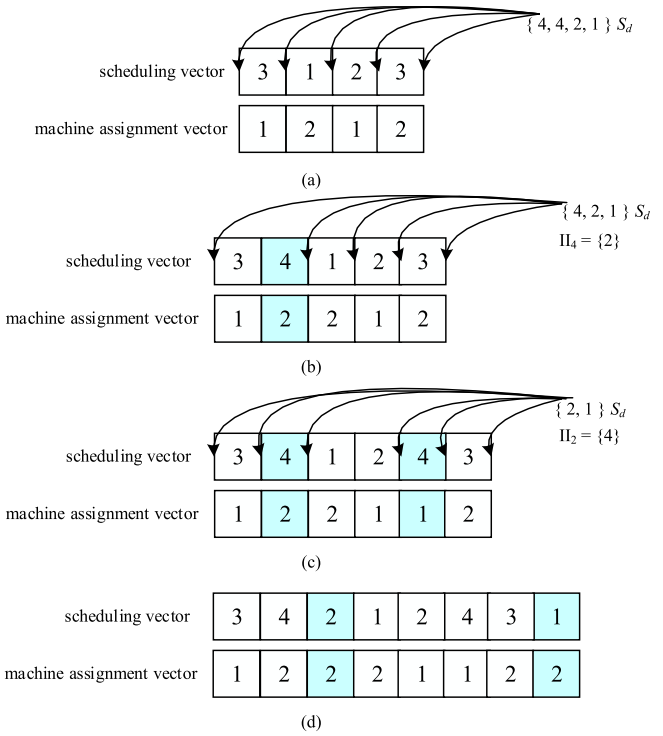

Fig. 6. IGSA Construction heuristic. (a) Solution vectors after the destruction heuristic. (b) Solution vectors after inserting the first operation of job 4. (c) Solution vectors after inserting the second operation of job 4. (d) Solution vectors after applying the construction heuristic.

F. IGSA Exploitation Heuristic

To complete the exploitation tasks of the proposed algorithm, we include a simple local search heuristic in the proposed algorithm. The heuristic is given as follows: 1) for the best individual found so far, generate N_{lo} number of neighboring solutions by using randomly selected mutation operators from the five problem-specific mutation approaches discussed in Section IV-C, where N_{lo} is usually set to $n/4$ (n denotes the total number of jobs) similar to that in [27] and 2) evaluate each newly generated solution and update the best solution if the former is better than latter.

G. IGSA Exploration Heuristic

It is obvious that, using the exploitation heuristic discussed above can easily be stuck to the local optimal. If the solution cannot be improved after a certain number of iterations, a strong exploitation heuristic should lead the search to more promising spaces. To enhance the exploration abilities of the proposed algorithm and also to decrease the time complexity of the algorithm, a simple yet effective exploration heuristic is investigated, with detailed steps as follows.

Step 1: Record the number of updates, U_{t}, done for the best solution found so far.

Step 2: Each time the best individual is updated, U_{t} is set to 0; otherwise, U_{t} is increased by one.

Step 3: If $U_{\text{t}} > U_{\text{limit}}$, where U_{limit} denotes the maximum limit update iteration count, then implement the destruction and construction heuristics on the best solution to generate a neighboring solution.

Step 4: Update the best solution as the neighboring solution if the latter is better than former.

H. IGSA Acceptance Criterion

After implementing the destruction and construction procedures, a newly generated solution will be selected to replace the current solution. Similar to Ruiz and Stützle [44], in the proposed IGSA algorithm, an SA-based heuristic is also used for the acceptance criterion, thereby providing the algorithm the ability to escape from the local optimal. In this study, we also adopt a very simple constant temperature acceptance criterion where Temperature $= (T \cdot \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{i=1}^{\theta_j} T_{i,j,k})/(n \cdot m \cdot 10)$, and T denotes a parameter to be calibrated.

I. IGSA Framework

On the basis of the component descriptions, the IGSA framework is given in Algorithm 3.

Algorithm 3 IGSA Framework

Input: System parameters; data for the problems
Output: the best solution found so far

Initialize a population of solutions, and select the best one as the initial solution π^0
π =*localsearch*(π^0) by randomly applying one of the five problem-specific mutation heuristics (cf. Section IV-C)
While the stop condition is not satisfied **do**
  π^1 = *IGSA_exploitation* (π) (cf. Section IV-F)
  Record the update times U_{t} for the best solution found so far (cf. Section IV-G).
  If $U_{\text{t}} > U_{\text{limit}}$ **then** (cf. Section IV-G)//exploration
    π_D =*IGSA_destruction*(π^1) (cf. Section IV-D)
    π^2 =*IGSA_construction*(π_D) (cf. Section IV-E)
  end
  π = *Acceptance criterion*(π^2) (cf. Section IV-H)
end
Output the best solution found so far

V. Experimental Results

This section discusses the computational experiments used to evaluate the performance of the proposed algorithm. Our algorithm was implemented in C++ on an Intel Core i7 3.4-GHz PC with 16 GB of memory. The compared algorithms include the ICA proposed by Karimi *et al.* [27], the integrated GA and glowworm swarm optimization (GA-GSO-GTHS) algorithm proposed by Liu *et al.* [26], the modified IG (MIG) algorithm by Aqel *et al.* [47], and the discrete ABC (DABC) algorithm [50]. In addition, to verify the searching quality, we compared the proposed algorithm with the exact solver IBM ILOG CPLEX on small-scale instances.

The following are the main reasons to select the abovementioned compared algorithms: 1) the GA-GSO-GTHS algorithm

is designed forCFJSP problems similar to those in this study, and the first types of instances are directly collected from the reference; 2) although the MIG algorithm is mainly designed for FJSPs without crane transportation, almost all its components can be easily embedded to solve CFJSP problems; 3) the DABC algorithm, which is also designed to solve FJSPs, can be easily adapted to CJSP problems easily; and 4) the ICA algorithm aimed to solve FJSPs with transportation time, rather than FJSPs with crane transportation. Notably, only a few works are considered FJSPs with crane transportation and energy consumptions; therefore, we select the above four algorithms to make fair comparisons. Moreover, we use CPLEX to make a fair comparison for small-scale instances, as it can verify the quality of the results obtained by the proposed algorithm.

Notably, this is the first study to investigate a CFJSP problem with crane transportation and energy consumption during the horizontal move and lifting crane movements. The adaption of the compared algorithm is given as follows.

1) The original GA-GSO-GTHS algorithm in the literature was used to study a similar problem, however, except lift transportation; therefore, all the components including the GA operators, two green transport heuristic strategies, and GSO operators were embedded. The only difference in our GA-GSO-GTHS algorithm with the original algorithm in the literature is that we have changed the decoding method to consider lift transportation.
2) The DABC algorithm is designed for solving multiobjective FJSPs, and therefore, a TS-based exploitation heuristic, self-adaptive strategy, and population initialization method were embedded. Notably, the encoding and decoding methods in the proposed IGSA algorithm were embedded in the DABC algorithm to adapt to CFJSP problems.
3) The ICA algorithm is designed for FJSPs with transportation time, and therefore, a mutation method, assimilation policy, imperialist updating and competition mechanism, imperialist development and local search, and SA-based strategy were included. The decoding method was also modified to adapt to the new problems.
4) The MIG algorithm is also for FJSP problems. Therefore, destruction and reconstruction strategies, dispatching rules, and control methods were embedded.

All the compared algorithms were recoded to adapt to solving the considered problem, where the parameters for them were set according to their literature. To verify the effectiveness and efficiency of the proposed algorithm, after 30 independent runs, the resulting best solutions were collected for performance comparisons. The performance measure is the relative percentage increase (RPI) calculated as follows:

$$\mathrm{RPI}(C) = \frac{f_c - f_b}{f_b} \times 100 \tag{47}$$

where f_b denotes the best fitness value collected by a compared algorithm and f_c the minimum fitness value found by a given algorithm.

A. Experimental Instances

Similar to that in [26], this study considered the extension instances of the manufacturing process in a large cement equipment manufacturing company in Tianjin, China. The detailed parameters of the crane in the workshop were directly collected from [26].

To test the performance of the proposed algorithm, we selected three types of instances. The first type of instances (hereinafter referred to as CFJSP-I) was obtained from [26], wherein four products are presented. The numbers of jobs are $n = \{7, 9, 10, 10\}$ for the four instances, respectively. The number of machines was set to six for all four instances. To further verify the performance of the proposed algorithm, we also generated a second type of instances (hereinafter referred to as CFJSP-II), which included 30 instances with a different number of jobs and machines, where the number of jobs is $n = \{20, 30, 40, 50, 80, 100\}$, and the number of machines is $m = \{6, 7, 8, 9, 10\}$. In addition, the total number of operations of each job was uniformly distributed in the interval of $[m/2, m]$. The instances can be found from the website *www.researchgate.net/publication/334083698_CFJSP_instances*, wherein crane lift operations are considered. Furthermore, to verify the quality of the results compared with lower bounds, we also randomly generated 18 small-scale instances and made detailed comparisons by using CPLEX. In the small-scale instances, the number of jobs is $n = \{5, 6, 7, 8, 9, 10\}$, and the number of machines was $m = \{2, 3, 4\}$.

It should be noted that the two objectives are in different units; in order to set the weighted values for them, the two objectives are transferred to cost objectives as follows: the average price of unit processing time is 60 \$/h and the price of unit energy consumption is 60 \$/kWh.

B. Experimental Parameters

The stopping criterion for each instance was set to be different with the instance scale, which is 0.6^*n s, where n is the total number of jobs. Notably, the time limit allows more computation time as the number of jobs increases. The two key parameters include the destruction length (d) and the acceptance criterion temperature parameter (T). We have used the design of experiments (DOE) approach for tuning the parameter of the algorithm. Precisely, we have carried out a full factorial design using the two parameters as factors at the following levels.

1) T, *Six Levels:* 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5.
2) d, *Five Levels:* 2, 3, 4, 5, and 6.

Totally, we yield 30 different combinations to be calibrated, where each combination was for one pair of the two factors. To calibrate the parameters, we randomly generated a set of instances, where the number of jobs was $n = \{20, 30, 40, 50, 80\}$ and the number of machines $m = \{5, 8, 10\}$, and the processing times were uniformly distributed in the interval of [5, 30]. In addition, the total number of operations of each job was uniformly distributed in the interval of $[m/2, m]$. All the 30 combinations to be calibrated were tackled in 30 independent runs.

TABLE I
ANOVA TABLE ON RPI FOR THE PARAMETERS

Source	ss	DF	MS	F	Prob>F
d	39562.7	4	9890.66	90.11	0
T	11402.3	5	2280.45	20.78	0
d*T	2317.5	20	115.88	1.06	0.4092
Error	9878.2	90	109.76		
total	63160.6	119			

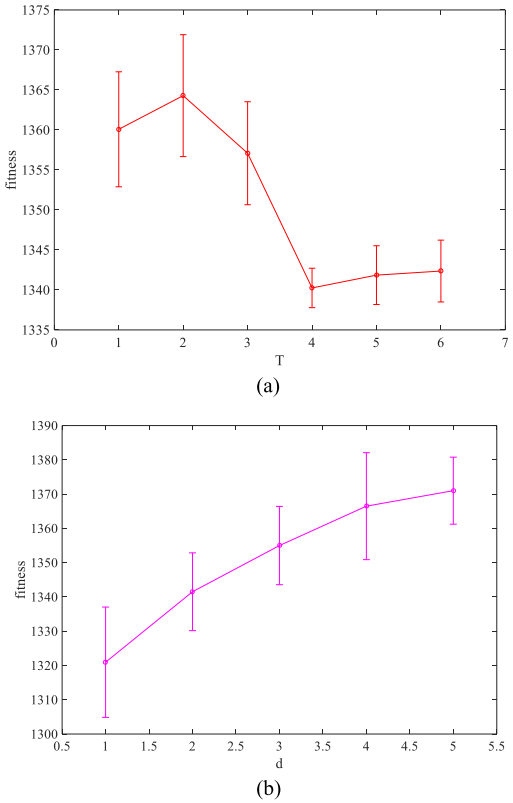

Fig. 7. 95% confidence interval for different selections of P_s and L_s. (a) 95% confidence interval for T. (b) 95% confidence interval for d.

The experiment was analyzed by a multifactor analysis of variance (ANOVA) technique. Table I lists the analysis results, which demonstrates that the two parameters are all of the significant factors (p-value <0.05). The parameter d, with an F-ratio of 90.11, significantly affects the performance of the proposed algorithm. However, the parameter T, with an F-ratio of 20.78, has a minor effect compared with d. Meanwhile, with a p-value of higher than 0.05, the factor interaction between the two parameters is nonsignificant.

Plots of the 95% confidence intervals for the fitness value under different selections of parameters T and d are shown in Fig. 7. According to Fig. 7(a), $T = 0.3$ yields a significantly better fitness value compared with other T values. According to Fig. 7(b), $d = 2$ yields a substantially better fitness value. Hence, on the basis of the results of this test, parameters T and d were set to 0.3 and 2, respectively.

In addition, through detailed experiments, we obtained that the maximum limit update iteration count $U_{\text{limit}} = 20$.

TABLE II
COMPARISONS OF FOUR CFJSP-I INSTANCES WITH GA-GSO-GTHS

Ins	fitness						RPI			
	Best		IGSA		GA-GSO-GTHS		IGSA		GA-GSO-GTHS	
	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1	63.0	434.6	63.0	434.6	75.6	493.0	**0.0**	**0.0**	20.0	13.5
2	74.1	517.6	74.1	517.6	87.8	539.8	**0.0**	**0.0**	18.5	4.3
3	74.0	512.9	74.0	512.9	80.8	554.1	**0.0**	**0.0**	9.2	8.0
4	79.0	549.0	79.0	549.0	90.9	602.2	**0.0**	**0.0**	15.1	9.7
Mean	72.5	503.5	72.5	503.5	83.8	547.3	**0.0**	**0.0**	15.7	8.9

C. Comparisons of the First Type of Instances

The first type of instances was collected from [26], wherein the manufacturing processes in a large cement equipment manufacturing company in Tianjin were investigated and simulated. The considered company mainly produced large complex cement equipment, and therefore, weighted and crane device energy consumption should be simultaneously considered. The detailed data and parameters of the machines and jobs can be found in the reference.

The comparison results between the proposed IGSA algorithm and GA-GSO-GTHS are given in Table II. The first column represents the instance number, and the second and third columns list the best results collected from the two compared algorithms. The fourth and fifth columns provide the results of the proposed IGSA algorithm, and the following two columns list the results of the GA-GSO-GTHS algorithm. The last four columns give the deviation values or RPI values of the two compared algorithms for the two objective values, respectively.

From Table II, the following conclusions can be drawn.

1) For solving the given four instances with different problem scales, the proposed IGSA algorithm obtained all the better results. For example, for the first instance, IGSA obtained the best solution with the two objectives, $f_1 = 63.0$ and $f_2 = 434.6$, which is obviously better than the results of the GA-GSO-GTHS algorithm with the two values 75.6 and 493.0, respectively.
2) From the last four columns, we can see that the proposed algorithm obtained better RPI values than the GA-GSO-GTHS algorithm, for all the four instances, especially in the first objective, i.e., the proposed algorithm performed better considering the makespan objective.
3) The average gap values of the two objectives Gap_1 and Gap_2 are calculated as follows:

$$\text{Gap}_1 = (\text{GA}-\text{GSO}-\text{GTHS}_{f1}-\text{IGSA}_{f1})/\text{IGSA}_{f1} \times 100 \tag{48}$$

$$\text{Gap}_2 = (\text{GA}-\text{GSO}-\text{GTHS}_{f2}-\text{IGSA}_{f2})/\text{IGSA}_{f2} \times 100. \tag{49}$$

From the last line, we can obtain the two gap values as $\text{Gap}_1 = 15.59\%$ and $\text{Gap}_2 = 8.70\%$, which shows the efficiency of the proposed algorithm.

TABLE III
COMPARISONS BETWEEN IGSA AND CPLEX SOLVER ($\omega = 0.9$)

Inst	Scale	Best	Fitness		RPI	
			IGSA(30s)	CPLEX(3h)	IGSA	CPLEX
1	5-2	111.10	111.10	111.10	**0.00**	**0.00**
2	5-3	72.10	72.10	72.10	**0.00**	**0.00**
3	5-4	54.80	54.80	54.80	**0.00**	**0.00**
4	6-2	122.80	122.80	122.80	**0.00**	**0.00**
5	6-3	124.12	124.12	125.00	**0.00**	0.71
6	6-4	84.30	84.30	84.30	**0.00**	**0.00**
7	7-2	141.50	141.50	141.50	**0.00**	**0.00**
8	7-3	113.05	113.05	115.60	**0.00**	2.26
9	7-4	86.20	86.20	86.20	**0.00**	**0.00**
10	8-2	172.48	172.48	172.48	**0.00**	**0.00**
11	8-3	131.10	131.10	131.10	**0.00**	**0.00**
12	8-4	107.10	107.10	108.80	**0.00**	1.59
13	9-2	205.50	205.50	205.50	**0.00**	**0.00**
14	9-3	139.40	139.40	139.40	**0.00**	**0.00**
15	9-4	113.30	113.30	113.50	**0.00**	**0.18**
16	10-2	281.30	281.30	281.30	**0.00**	**0.00**
17	10-3	153.90	153.90	153.90	**0.00**	**0.00**
18	10-4	127.60	127.60	130.30	**0.00**	2.12
Mean		130.09	130.09	130.54	**0.00**	0.34

D. Performance Compared With the Exact Solver CPLEX

To test the solution quality obtained by the proposed algorithm, we employed an exact solver IBM ILOG CPLEX 12.7 to calculate the MIP model for 18 small-scale instances. The settings of the exact solver are as follows: the maximum number of threads is 3 and the working memory is 4096 MB. For each instance, the CPLEX solver was performed five times. The CPU time limit was set to 3 h for each run.

Table III shows the comparison results between the proposed IGSA algorithm and the CPLEX solver. The first column gives the instance number, and the second column shows the problem scale with the numbers of jobs and machines. The best results achieved by the two methods are given in the third column. The weighted fitness values for each instance are shown in the following two columns for IGSA and CPLEX, respectively. The last two columns show the RPI values for the two algorithms. It can be observed from the table that: 1) for the given 18 small-scale instances, the proposed IGSA algorithm obtained higher quality solutions than the CPLEX solver; 2) for the instances with more number of available machines, the exact CPLEX solver performs slightly worse than IGSA; and 3) the gap of the average fitness values can be calculated from the last line, and the resultant gap value is 0.35%, which also shows the superior performance of the proposed IGSA algorithm over the exact CPLEX solver.

E. Efficiency of the Exploration Heuristic

To investigate the effectiveness of the exploration heuristic discussed in Section IV-G, we coded the two different types of IGSA algorithms, i.e., the IGSA-NE algorithm without the exploration heuristic and the IGSA algorithm with all the components discussed in Section IV. It should be noted that, in IGSA-NE, IG-based destruction and construction procedures will be performed at each iteration, while, in the IGSA algorithm, the destruction and construction procedures will be performed following the certain conditions discussed in Section IV-G. All the other components of the two compared algorithms are set the same.

To evaluate whether the difference between the two methods was significant, we performed a multifactor analysis of variance (ANOVA), in which the compared methods were considered as factors. Fig. 8(a) shows the means and 95% least-significant difference (LSD) intervals for the fitness values of the two compared methods. The p-value is close to zero; hence, there exist significant differences between the compared methods. It can be concluded that the proposed exploration strategy significantly improves the performance of the proposed algorithm.

F. Efficiency of the Construction Heuristic

To investigate the effectiveness of the construction heuristic discussed in Section IV-E, we coded the two different types of IGSA algorithms, i.e., the IGSA-NR algorithm without the construction heuristic considering Lemma 1 to omit the replicated positions and the IGSA algorithm with all the components discussed in Section IV. Fig. 8(b) shows the means and 95% LSD intervals for the fitness values of the two compared methods. It can be concluded that the proposed construction strategy significantly improves the performance of the proposed algorithm.

G. Comparisons With Efficient Algorithms

To further verify the performance of the proposed algorithm compared with other efficient algorithms, we select the following algorithms, namely, DABC, ICA, GA-GSO-GTHS, and MIG. The results obtained by all the compared algorithms after 30 independent runs are presented in Tables IV and V, where the RPI values for all the compared algorithms are provided, respectively.

Table IV presents the results for solving instances with the weight value set to 0.1. The best fitness values for each instance are list in the second column, while the RPI values obtained by the five compared algorithms are given in the last five columns, respectively. Table V shows the average results collected by all the nine weighted values, i.e., from 0.1 to 0.9. All the compared algorithms were run for CPU time = 30 s for each instance.

It can be observed from the two tables that: 1) considering that $\omega = 0.1$, IGSA obtained 24 satisfactory values out of the given 30 instances, while the other four compared algorithms performed worse; for example, the second-best algorithm MIG could obtain only six satisfactory values; 2) considering the average weighted values, the proposed algorithm also obtained 27 satisfactory values, which is obviously better than the other compared algorithms, especially for the problem with large scales; and 3) from the average RPI values listed in the two tables, the robust performance of the proposed IGSA algorithm can be observed. Fig. 8(c) shows the means and the 95% LSD intervals for the five compared algorithms. It can be concluded from Fig. 8(c) that the proposed IGSA algorithm performs significantly better than the other four algorithms.

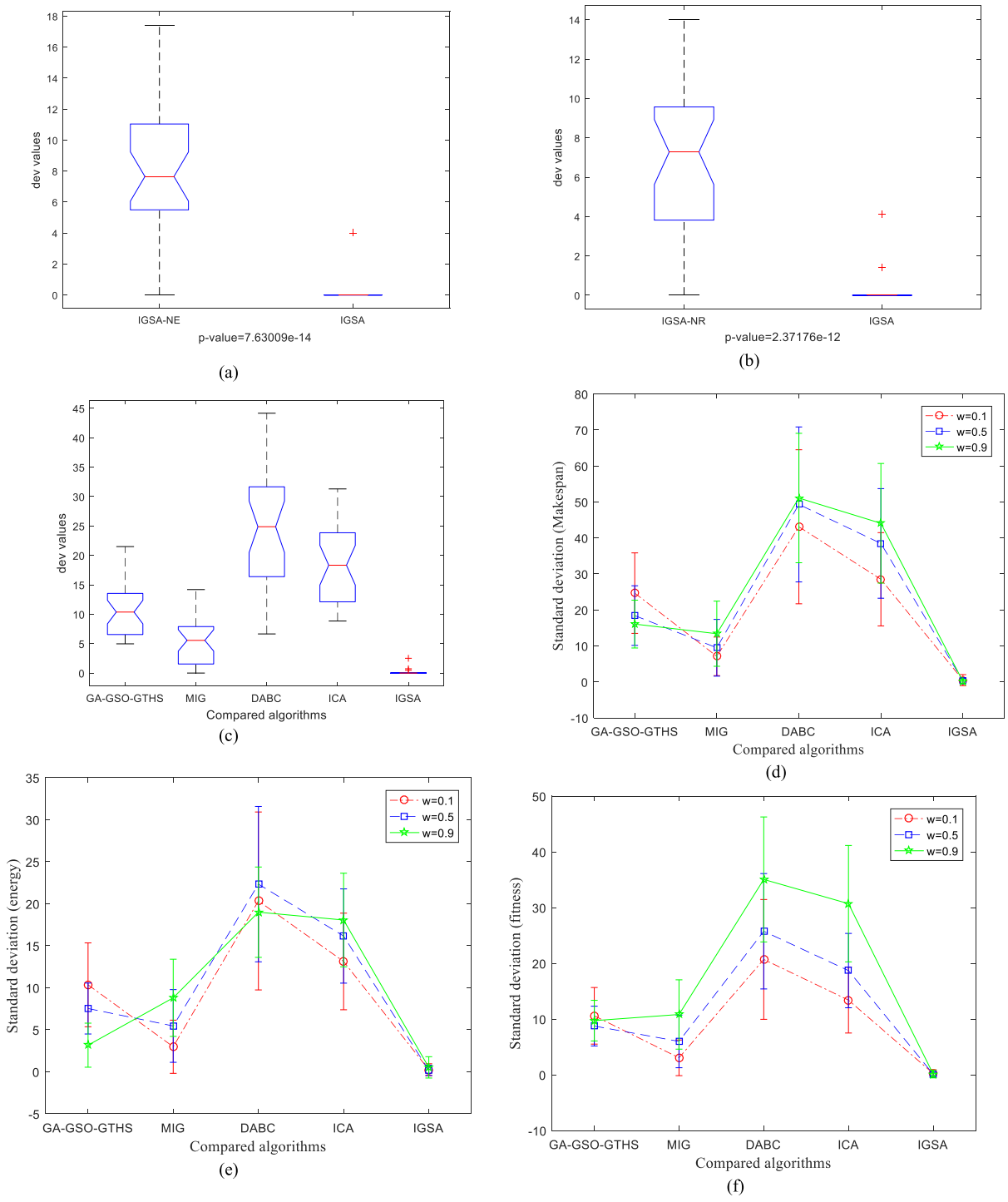

Fig. 8. ANOVA Comparison results. (a) Comparisons of the exploration heuristic. (b) Comparisons of the construction heuristic. (c) ANOVA Comparisons for the average weighted values. (d) Standard deviation comparisons for makespan. (e) Standard deviation comparisons for energy consumption. (f) Standard deviation comparisons for total fitness.

Furthermore, we made a detailed comparison for the standard deviation comparisons for makespan, energy consumption, and the weighted fitness values under three conditions, i.e., 0.1, 0.5, and 0.9, respectively. Fig. 8(d)–(f) shows the comparison results. It can be observed from these three subfigures that the proposed IGSA algorithm performs better with different weighted values. In addition, Fig. 9 shows the distribution of solutions with different weights optimized by the five compared algorithms. As observed from Fig. 9, IGSA shows competitive performance considering the makespan, energy consumption, and weighted fitness value.

Fig. 10 compares the convergence curves for "Inst5," where the convergence performance of the proposed algorithm can be verified. The convergence comparisons for other instances can

TABLE IV

COMPARISONS OF CFJSP-II INSTANCES ($\omega = 0.1$)

Inst	Scale	Best	DABC	ICA	GA-GSO-GTHS	MIG	IGSA
Inst1	20-6	1098.49	4.65	9.26	6.20	**0.00**	2.69
Inst2	20-7	1178.42	5.10	6.24	3.23	**0.00**	0.23
Inst3	20-8	1522.09	8.73	9.07	5.12	0.18	**0.00**
Inst4	20-9	1524.71	9.76	6.15	5.39	2.84	**0.00**
Inst5	20-10	2138.29	15.55	16.93	10.95	1.40	**0.00**
Inst6	30-6	1965.15	8.44	7.26	6.48	**0.00**	2.83
Inst7	30-7	2002.11	8.48	9.23	7.56	**0.00**	0.46
Inst8	30-8	2312.33	17.77	12.02	8.11	2.18	**0.00**
Inst9	30-9	2447.66	25.01	16.42	13.41	3.37	**0.00**
Inst10	30-10	3065.49	32.76	21.81	19.92	5.65	**0.00**
Inst11	40-6	2574.34	11.04	9.54	4.97	0.21	**0.00**
Inst12	40-7	2264.87	12.24	7.54	3.74	0.51	**0.00**
Inst13	40-8	3076.18	19.77	13.87	11.97	5.11	**0.00**
Inst14	40-9	3256.02	24.81	13.73	12.69	3.02	**0.00**
Inst15	40-10	3869.12	36.17	20.86	17.81	4.57	**0.00**
Inst16	50-6	3220.85	13.26	7.82	6.98	**0.00**	0.51
Inst17	50-7	3082.06	15.13	5.92	6.20	0.19	**0.00**
Inst18	50-8	4127.39	31.36	19.73	15.68	2.39	**0.00**
Inst19	50-9	3880.12	34.39	22.24	16.87	4.97	**0.00**
Inst20	50-10	4876.43	45.45	26.62	19.99	9.73	**0.00**
Inst21	80-6	5336.81	10.76	6.80	5.26	**0.00**	0.24
Inst22	80-7	4792.95	14.30	9.14	7.10	1.59	**0.00**
Inst23	80-8	6249.12	27.58	15.57	13.97	5.68	**0.00**
Inst24	80-9	6083.76	27.25	16.07	13.11	4.15	**0.00**
Inst25	80-10	8264.90	37.27	22.91	18.32	12.29	**0.00**
Inst26	100-6	6704.90	16.04	8.12	6.06	0.56	**0.00**
Inst27	100-7	6231.73	15.40	7.57	6.81	0.76	**0.00**
Inst28	100-8	8385.03	27.49	15.51	11.87	4.79	**0.00**
Inst29	100-9	8113.70	29.81	16.62	13.86	5.17	**0.00**
Inst30	100-10	10108.40	35.39	21.60	17.91	10.18	**0.00**
Mean		4125.11	20.71	13.40	10.58	3.05	**0.23**

TABLE V

COMPARISONS OF CFJSP-II INSTANCES (AVERAGE OF ALL THE NINE WEIGHTED VALUES)

Inst	Scale	Best	DABC	ICA	GA-GSO-GTH	MIG	IGSA
Inst1	20-6	719.64	6.65	9.48	5.85	**0.00**	2.5
Inst2	20-7	759.45	8.00	9.13	4.97	0.81	**0.0**
Inst3	20-8	971.13	11.3	12.9	5.35	1.97	**0.0**
Inst4	20-9	974.97	16.3	15.5	6.62	3.10	**0.0**
Inst5	20-10	1385.0	25.1	24.1	10.32	7.82	**0.0**
Inst6	30-6	1272.9	11.2	8.85	5.48	**0.00**	0.4
Inst7	30-7	1282.4	11.2	10.7	6.27	**0.00**	0.7
Inst8	30-8	1481.8	24.5	18.1	10.42	4.90	**0.0**
Inst9	30-9	1605.2	31.4	22.7	12.86	6.76	**0.0**
Inst10	30-10	1999.7	39.1	28.8	16.94	11.0	**0.0**
Inst11	40-6	1654.8	16.3	12.0	7.00	1.06	**0.0**
Inst12	40-7	1455.7	21.1	13.1	6.55	1.64	**0.0**
Inst13	40-8	1989.6	28.1	19.5	12.81	6.38	**0.0**
Inst14	40-9	2103.7	31.0	20.2	12.71	6.21	**0.0**
Inst15	40-10	2541.9	38.9	27.2	17.29	9.91	**0.0**
Inst16	50-6	2084.1	17.8	12.5	7.19	1.37	**0.0**
Inst17	50-7	1957.3	21.4	12.8	7.33	1.55	**0.0**
Inst18	50-8	2705.4	35.9	25.5	16.39	7.38	**0.0**
Inst19	50-9	2537.5	38.8	27.2	18.26	9.60	**0.0**
Inst20	50-10	3278.6	44.1	31.3	21.50	12.8	**0.0**
Inst21	80-6	3466.4	15.3	10.4	6.02	1.54	**0.0**
Inst22	80-7	3130.1	18.4	11.9	6.51	2.22	**0.0**
Inst23	80-8	4197.8	29.4	19.6	13.39	7.44	**0.0**
Inst24	80-9	4015.1	30.6	20.3	13.54	7.90	**0.0**
Inst25	80-10	5641.6	37.8	25.1	17.42	14.1	**0.0**
Inst26	100-6	4423.7	19.0	12.1	7.44	1.49	**0.0**
Inst27	100-7	4068.7	19.1	12.2	6.70	1.90	**0.0**
Inst28	100-8	5615.2	28.5	18.5	12.39	7.40	**0.0**
Inst29	100-9	5440.9	31.6	20.0	12.67	8.05	**0.0**
Inst30	100-1	6848.4	36.0	23.8	15.52	13.8	**0.0**
Mean		2720.3	24.8	17.8	10.79	5.34	**0.1**

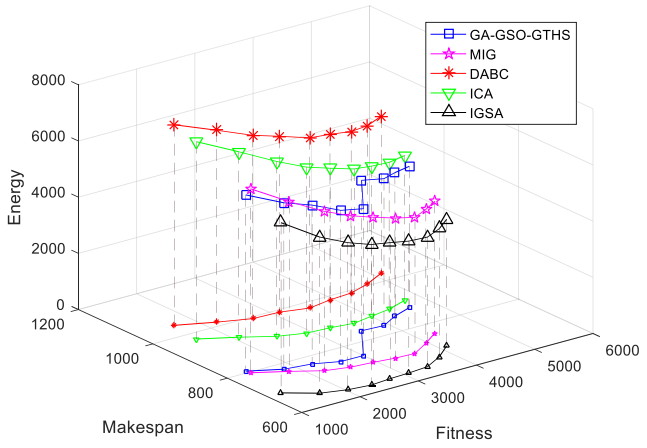

Fig. 9. Distribution of solutions with different weights.

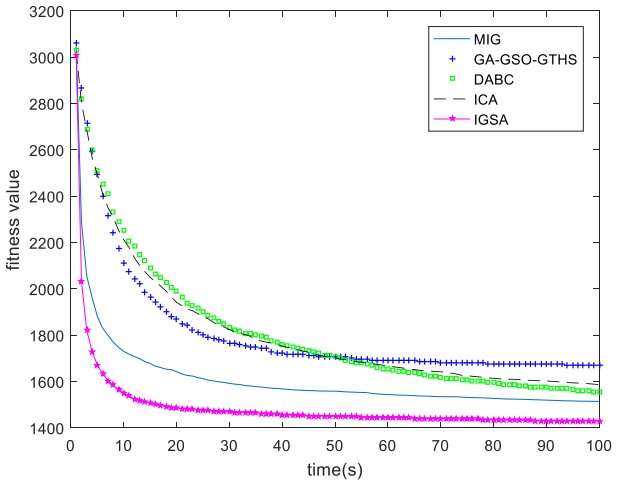

Fig. 10. Comparisons of the convergence curves for Inst5 (average of all the nine weight values).

be found in the Supplementary Material. The Gantt charts for the optimal solutions for the instance 20-6 are given in Fig. 11, wherein each operation is represented by a rectangle labeled with the job number and operation number. A different transport action is represented by rectangles filled with different colors. It can be also concluded from the figure that the proposed algorithm obtained a feasible solution with high quality.

H. Normalization Comparisons

It can be observed from Fig. 9 that, although the two objectives have been converted into the same unit, they have different ranges. In this section, normalization comparisons are conducted to verify the performance of the proposed algorithm, where each objective is first normalized before calculating the weighted objective value. The normalized fitness

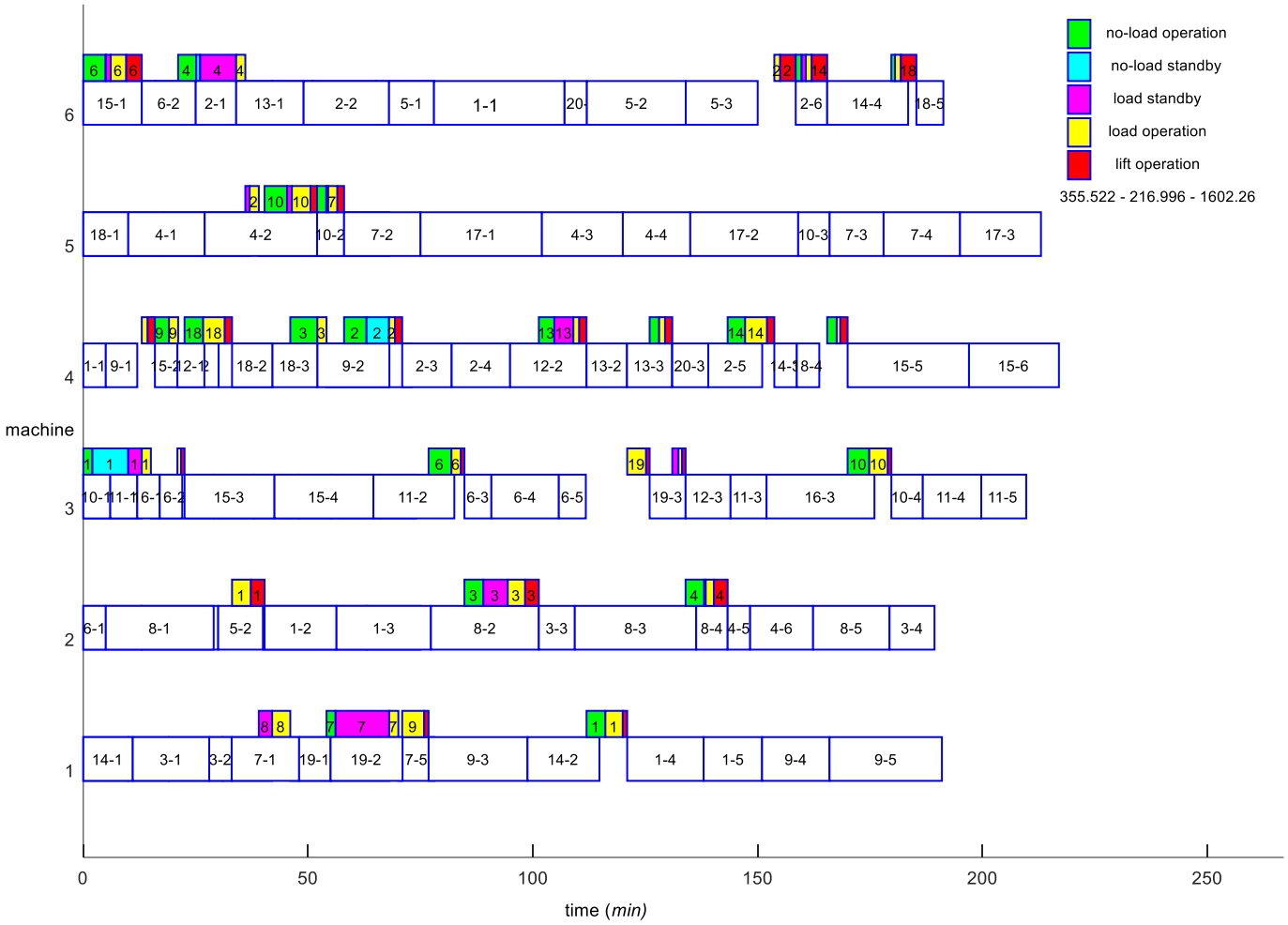

Fig. 11. Gantt chart for the optimal result for 20-6 ($f = 355.522$).

value is calculated as follows:

$$f_i' = \frac{f_i}{f_i^{\max}}, \quad i = 1, 2 \tag{50}$$

where f_i is the ith objective value, $f_i^{\max}$ is the maximum value of the ith objective in the initial population, and f_i' is the normalized objective value.

Table VI gives the average comparison results of the nine weighted values, i.e., from 0.1 to 0.9. It can be concluded from Table VI that: 1) considering the normalized objective comparisons, IGSA obtained 27 satisfactory values out of the given 30 instances, which is obviously better than the other compared algorithms, especially for the problem with large scales and 2) in a nutshell, the normalization comparisons also verify the efficient performance of the proposed algorithm.

VI. Conclusion

This study considered an FJSP with crane transportation, a typical industrial optimization problem. The objective was to minimize the makespan and energy consumptions. The main contributions of this study are as follows: 1) a mathematical model of the problem considering the crane transportation was first developed; 2) an effective algorithm that combines IG and SA algorithms was proposed; 3) several problem-specific heuristics were developed to enhance the exploitation abilities of the proposed algorithm; and 4) an efficient exploration heuristic and an SA-based acceptance method were developed to strengthen the exploration abilities of the IGSA algorithm.

A. Discussion of the Results

This study considered the extension instances of the manufacturing process in a large cement equipment manufacturing company in Tianjin. Three types of instances were tested.

The first type of instances was directly collected from [26]. Compared with the GA-GSO-GTHS algorithm, the gap values for the two objectives were $\text{Gap}_1 = 15.59\%$ and $\text{Gap}_2 = 8.70\%$, which verifies the efficiency of the proposed algorithm in solving small-scale realistic problems.

The second type of instance was used to test the performance bound of the proposed algorithm. Compared with the exact CPLEX solver, the resultant gap value of the average performance was 0.35%, which verified the performance bound of IGSA.

The third type of instances was used to make detailed comparisons of the proposed algorithm with several efficient algorithms, namely, DABC, ICA, GA-GSO-GTHS, and MIG. Considering the average performances for all the nine weighted values, it was observed that the proposed algorithm performed efficiently for instances of different scales.

TABLE VI
COMPARISONS OF CFJSP-II INSTANCES (AVERAGE OF ALL THE NINE WEIGHTED VALUES AND NORMALIZED OBJECTIVES)

Inst	Scale	DABC	ICA	GA-GSO-GTHS	MIG	IGSA
Inst1	20-6	3.58	0.52	**0.00**	5.84	4.45
Inst2	20-7	**0.00**	2.44	**0.00**	4.60	2.44
Inst3	20-8	3.09	3.74	1.56	3.02	**0.00**
Inst4	20-9	5.30	6.74	**0.00**	4.09	**0.00**
Inst5	20-10	17.82	16.22	10.05	1.41	**0.00**
Inst6	30-6	2.18	7.57	0.08	**0.00**	1.05
Inst7	30-7	8.35	7.22	**0.00**	1.87	**0.00**
Inst8	30-8	15.15	13.40	7.69	2.76	**0.00**
Inst9	30-9	19.83	15.62	15.63	**0.00**	**0.00**
Inst10	30-10	31.92	25.24	21.96	2.21	**0.00**
Inst11	40-6	8.72	3.96	4.31	2.68	**0.00**
Inst12	40-7	16.17	10.50	5.42	4.79	**0.00**
Inst13	40-8	27.93	15.09	14.37	1.80	**0.00**
Inst14	40-9	18.16	13.59	10.44	10.06	**0.00**
Inst15	40-10	35.73	22.70	20.71	1.61	**0.00**
Inst16	50-6	12.34	0.77	2.27	0.59	**0.00**
Inst17	50-7	18.72	7.93	5.93	3.51	**0.00**
Inst18	50-8	36.28	16.68	18.53	15.36	**0.00**
Inst19	50-9	48.48	27.17	27.84	17.66	**0.00**
Inst20	50-10	49.97	29.23	29.19	21.87	**0.00**
Inst21	80-6	31.89	7.31	2.42	3.08	**0.00**
Inst22	80-7	29.18	4.10	1.43	0.27	**0.00**
Inst23	80-8	52.03	15.83	15.48	12.13	**0.00**
Inst24	80-9	50.12	17.70	17.88	12.60	**0.00**
Inst25	80-10	62.77	20.53	24.53	17.41	**0.00**
Inst26	100-6	35.22	4.03	2.38	4.17	**0.00**
Inst27	100-7	47.57	12.15	9.11	7.86	**0.00**
Inst28	100-8	55.09	16.25	16.01	11.14	**0.00**
Inst29	100-9	58.24	16.46	16.80	12.09	**0.00**
Inst30	100-10	56.50	20.20	21.33	13.27	**0.00**
Mean		28.61	12.70	10.78	6.66	**0.26**

B. Discussion of the Sensitivity Analysis

A parameter sensitivity analysis was performed by using the DOE approach with a full factorial design. The experimental results of the parameter sensitivity analysis were studied by a multifactor ANOVA technique. Therefore, the proposed algorithm could adapt to CFJSP with different problem scales.

To make detailed comparisons of the proposed algorithm with DABC, ICA, GA-GSO-GTHS, and MIG, we performed the experiments with all the nine weighted values, i.e., from 0.1 to 0.9. Considering the average weighted values, the proposed algorithm obtained 27 satisfactory values out of the given 30 instances. The standard deviation comparisons for makespan, energy consumption, and weighted fitness values under three conditions, i.e., 0.1, 0.5, and 0.9, were also used to perform the sensitivity analysis of the performance of the compared algorithm.

C. Discussion of the Effectiveness of the Proposed Framework

From the experimental comparisons, it was observed that the proposed algorithm was efficient in solving CFJSP with different problem scales. The main advantages of IGSA are as follows.

1) For the IG component, an improved construction heuristic considering the problem features was proposed; the heuristic can decrease the number of insertion iterations. In addition, problem-specific exploration and exploitation heuristics were embedded in the proposed algorithm to balance the global and local search abilities.
2) An SA-based heuristic was embedded to further provide the algorithm with the ability to escape from the local optimal.
3) In a nutshell, the improved IG heuristic performs the exploration and exploitation tasks of the proposed algorithm, and the SA-based heuristic further enhances the global searchability.

D. Discussion of the Future Work

The limitation of the proposed algorithm is that multiple crane transportations have not been considered. Therefore, the future work mainly focuses on the following tasks.

1) Apply the proposed algorithm to solve FJSPs with multiple cranes. Notably, a CFJSP with multiple cranes will be different than that in the current work. For considering multiple crane constraints, the coding and decoding methods should be improved to record the crane transportation routes and crane assignment. In addition, the local search operators should be redesigned to adapt to the improved coding approach. The other components, including the construction and destruction heuristics, and the SA-based acceptance criterion can be embedded with slight improvements. Moreover, to adapt to the CJFP with multiple cranes, more problem-specific heuristics should be proposed, which are challenging future works.
2) Apply a Pareto-based multiobjective algorithm, and consider other realistic constraints and objectives into the considered problem.
3) Apply the interval multiobjective optimization methods to solve the considered problems under dynamic environments.

REFERENCES

[1] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 32, no. 1, pp. 1–13, Feb. 2002.
[2] L. Sun, L. Lin, M. Gen, and H. Li, "A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 5, pp. 1008–1022, May 2019.
[3] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019.
[4] J. Li, Z.-M. Liu, C. Li, and Z. Zheng, "Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem," *IEEE Trans. Fuzzy Syst.*, early access, Aug. 12, 2020, doi: 10.1109/TFUZZ.2020.3016225.
[5] M. Dawande, H. N. Geismar, S. P. Sethi, and C. Sriskandarajah, "Sequencing and scheduling in robotic cells: Recent developments," *J. Scheduling*, vol. 8, no. 5, pp. 387–426, Oct. 2005.
[6] S. P. Sethi, J. B. Sidney, and C. Sriskandarajah, "Scheduling in dual gripper robotic cells for productivity gains," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 324–341, Jun. 2001.
[7] V. Kats and E. Levner, "A faster algorithm for 2-cyclic robotic scheduling with a fixed robot route and interval processing times," *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 51–56, Feb. 2011.
[8] A. Che, V. Kats, and E. Levner, "An efficient bicriteria algorithm for stable robotic flow shop scheduling," *Eur. J. Oper. Res.*, vol. 260, no. 3, pp. 964–971, Aug. 2017.
[9] A. Che and C. Chu, "Multi-degree cyclic scheduling of two robots in a no-wait flowshop," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 2, pp. 173–183, Apr. 2005.
[10] D. Shabtay, K. Arviv, H. Stern, and Y. Edan, "A combined robot selection and scheduling problem for flow-shops with no-wait restrictions," *Omega*, vol. 43, pp. 96–107, Mar. 2014.

[11] D. Shabtay and K. Arviv, "Optimal robot scheduling to minimize the makespan in a three-machine flow-shop environment with job-independent processing times," *Appl. Math. Model.*, vol. 40, nos. 5–6, pp. 4231–4247, Mar. 2016.

[12] K. Arviv, H. Stern, and Y. Edan, "Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem," *Int. J. Prod. Res.*, vol. 54, no. 4, pp. 1196–1209, Feb. 2016.

[13] A. Elmi and S. Topaloglu, "Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization," *Comput. Oper. Res.*, vol. 73, pp. 67–83, Sep. 2016.

[14] S. Burnwal and S. Deb, "Scheduling optimization of flexible manufacturing system using cuckoo search-based approach," *Int. J. Adv. Manuf. Technol.*, vol. 64, nos. 5–8, pp. 951–959, Feb. 2013.

[15] A. Elmi and S. Topaloglu, "A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots," *Comput. Oper. Res.*, vol. 40, no. 10, pp. 2543–2555, Oct. 2013.

[16] C. Sangsawang, K. Sethanan, T. Fujimoto, and M. Gen, "Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2395–2410, Apr. 2015.

[17] G. D. Batur, S. Erol, and O. E. Karasan, "Robot move sequence determining and multiple part-type scheduling in hybrid flexible flow shop robotic cells," *Comput. Ind. Eng.*, vol. 100, pp. 72–87, Oct. 2016.

[18] S. S. Zabihzadeh and J. Rezaeian, "Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time," *Appl. Soft Comput.*, vol. 40, pp. 319–330, Mar. 2016.

[19] J. Hurink and S. Knust, "A tabu search algorithm for scheduling a single robot in a job-shop environment," *Discrete Appl. Math.*, vol. 119, nos. 1–2, pp. 181–203, Jun. 2002.

[20] J. Hurink and S. Knust, "Tabu search algorithms for job-shop problems with a single transport robot," *Eur. J. Oper. Res.*, vol. 162, no. 1, pp. 99–111, Apr. 2005.

[21] H. E. Nouri, O. B. Driss, and K. Ghédira, "Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment," *Appl. Intell.*, vol. 45, no. 3, pp. 808–828, Oct. 2016.

[22] S. Q. Liu and E. Kozan, "A hybrid metaheuristic algorithm to optimise a real-world robotic cell," *Comput. Oper. Res.*, vol. 84, pp. 188–194, Aug. 2017.

[23] A. Elmi and S. Topaloglu, "Cyclic job shop robotic cell scheduling problem: Ant colony optimization," *Comput. Ind. Eng.*, vol. 111, pp. 417–432, Sep. 2017.

[24] A. Ahmadi-Javid and P. Hooshangi-Tabrizi, "Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an anarchic society optimization algorithm," *Comput. Oper. Res.*, vol. 84, pp. 73–91, Aug. 2017.

[25] H. E. Nouri, O. B. Driss, and K. Ghédira, "Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model," *Comput. Ind. Eng.*, vol. 102, pp. 488–501, Dec. 2016.

[26] Z. Liu, S. Guo, and L. Wang, "Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption," *J. Cleaner Prod.*, vol. 211, pp. 765–786, Feb. 2019.

[27] S. Karimi, Z. Ardalan, B. Naderi, and M. Mohammadi, "Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm," *Appl. Math. Model.*, vol. 41, pp. 82–667, 2017.

[28] S.-S. Leu and S.-T. Hwang, "GA-based resource-constrained flow-shop scheduling model for mixed precast production," *Autom. Construct.*, vol. 11, no. 4, pp. 439–452, Jun. 2002.

[29] H. Li, J. Duan, and Q. Zhang, "Multi-objective integrated scheduling optimization of semi-combined marine crankshaft structure production workshop for green manufacturing," *Trans. Inst. Meas. Control*, vol. 43, no. 3, pp. 579–596, Feb. 2021, doi: 10.1177/0142331220945917.

[30] B. Zhou and X. Liao, "Particle filter and Levy flight-based decomposed multi-objective evolution hybridized particle swarm for flexible job shop greening scheduling with crane transportation," *Appl. Soft. Comput.*, vol. 91, pp. 1–18, Jun. 2020, doi: 10.1016/j.asoc.2020.106217.

[31] J. Irizarry and E. P. Karan, "Optimizing location of tower cranes on construction sites through GIS and BIM integration," *J. Inf. Technol. Construct.*, vol. 17, no. 23, pp. 351–366, 2012.

[32] M. Marzouk and A. Abubakr, "Decision support for tower crane selection with building information models and genetic algorithms," *Autom. Construct.*, vol. 61, no. 1, pp. 1–15, 2016.

[33] A. Khodabandelu, J. Park, and C. Arteaga, "Crane operation planning in overlapping areas through dynamic supply selection," *Automat. Construct.*, vol. 117, no. 9, 2020, Art. no. 103253, doi: 10.1016/j.autcon.2020.103253.

[34] K. Kawada, H. Sogo, T. Yamamoto, and Y. Mada, "Robust PD sway control of a lifted load for a mobile crane using a genetic algorithm," *IEEJ Trans. Sensors Micromach.*, vol. 123, no. 10, pp. 1181–1186, 2003.

[35] B. Peng, F. Flager, and J. Wu, "A method to optimize mobile crane and crew interactions to minimize construction cost and time," *Autom. Construct.*, vol. 95, no. 11, pp. 10–19, 2018.

[36] H. Taghaddos, A. Eslami, U. Hermann, S. AbouRizk, and Y. Mohamed, "Auction-based simulation for industrial crane operations," *Autom. Construct.*, vol. 104, no. 8, pp. 107–119, 2019.

[37] H. Nam and T. Lee, "A scheduling problem for a novel container transport system: A case of mobile harbor operation schedule," *Flex. Serv. Manuf. J.*, vol. 25, pp. 576–608, Dec. 2013.

[38] G. Zhao, J. Liu, L. Tang, R. Zhao, and Y. Dong, "Model and heuristic solutions for the multiple double-load crane scheduling problem in slab yards," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1307–1319, Jul. 2020.

[39] H. H. Tostani, H. Haleh, S. M. H. Molana, and F. M. Sobhani, "A bi-level bi-objective optimization model for the integrated storage classes and dual shuttle cranes scheduling in AS/RS with energy consumption, workload balance and time windows," *J. Cleaner Prod.*, vol. 257, Jun. 2020, Art. no. 120409, doi: 10.1016/j.jclepro.2020.120409.

[40] L. Bukata, P. Šůcha, and Z. Hanzálek, "Optimizing energy consumption of robotic cells by a branch & bound algorithm," *Comput. Oper. Res.*, vol. 102, pp. 52–66, Feb. 2019.

[41] Y. Chen, X. Mao, S. Yang, and Q. Wang, "Cost-efficient inter-robot delivery for resource-constrained and interdependent multi-robot schedules," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 1, Jan. 2019, Art. no. 172988141982804.

[42] S. Gürel, H. Gultekin, and V. E. Akhlaghi, "Energy conscious scheduling of a material handling robot in a manufacturing cell," *Robot. Comput.-Integr. Manuf.*, vol. 58, pp. 97–108, Aug. 2019.

[43] V. Kats and E. Levner, "On the existence of dominating 6-cyclic schedules in four-machine robotic cells," *Eur. J. Oper. Res.*, vol. 268, no. 2, pp. 755–759, Jul. 2018.

[44] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop with scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, 2007.

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

[46] R. Ruiz, Q.-K. Pan, and B. Naderi, "Iterated greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, pp. 213–222, Mar. 2019.

[47] G. Al Aqel, X. Li, and L. Gao, "A modified iterated greedy algorithm for flexible job shop scheduling problem," *Chin. J. Mech. Eng.*, vol. 32, no. 1, p. 21, Dec. 2019.

[48] J. E. C. Arroyo, J. Y.-T. Leung, and R. G. Tavares, "An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 239–254, Jan. 2019.

[49] J.-Q. Li *et al.*, "Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2425–2439, Jun. 2020.

[50] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Appl. Math. Model.*, vol. 38, no. 3, pp. 1111–1132, Feb. 2014.

[51] J.-Q. Li *et al.*, "Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems," *J. Cleaner Prod.*, vol. 250, Mar. 2020, Art. no. 119464.

[52] D. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multi-objective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 109–1097, Feb. 2019.

[53] Y. Du, J.-Q. Li, C. Luo, and L.-L. Meng, "A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100861, doi: 10.1016/j.swevo.2021.100861.

[54] J. Sun, Z. Miao, D. Gong, X.-J. Zeng, J. Li, and G. Wang, "Interval multiobjective optimization with memetic algorithms," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3444–3457, Aug. 2020, doi: 10.1109/TCYB.2019.2908485.

[55] J.-Q. Li *et al.*, "Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots," *Swarm Evol. Comput.*, vol. 52, Feb. 2020, Art. no. 100600, doi: 10.1016/j.swevo.2019.100600.

[56] Q.-K. Pan, L. Gao, and L. Wang, "A multi-objective hot-rolling scheduling problem in the compact strip production," *Appl. Math. Model.*, vol. 73, pp. 327–348, Sep. 2019.