Improved artificial immune algorithm for the flexible job shop problem with transportation time

Xiao-long Chen¹, Jun-qing Li¹,², Yu-yan Han² and Hong-yan Sang²

Abstract
The flexible job shop problem (FJSP), as one branch of the job shop scheduling, has been studied during recent years. However, several realistic constraints including the transportation time between machines and energy consumptions are generally ignored. To fill this gap, this study investigated a FJSP considering energy consumption and transportation time constraints. A sequence-based mixed integer linear programming (MILP) model based on the problem is established, and the weighted sum of maximum completion time and energy consumption is optimized. Then, we present a combinational meta-heuristic algorithm based on a simulated annealing (SA) algorithm and an artificial immune algorithm (AIA) for this problem. In the proposed algorithm, the AIA with an information entropy strategy is utilized for global optimization. In addition, the SA algorithm is embedded to enhance the local search abilities. Eventually, the Taguchi method is used to evaluate various parameters. Computational comparison with the other meta-heuristic algorithms shows that the improved artificial immune algorithm (IAIA) is more efficient for solving FJSP with different problem scales.

Keywords
Flexible job shop, transportation time, energy consumption, mixed integer linear programming model, improved artificial immune algorithm

Introduction
In recent years, increasing numbers of enterprises have been paying more attention to the problem of how to optimize the scheduling of workshop to maximize their efficiency. As a result, the job-shop problem (JSP) has become one of most popular research topics in the literature owing to its potential to dramatically decrease costs and increase throughput. This has created a need for research on the scheduling problem. Traditional JSP can be concluded as follows: there are a set of jobs to be carried out on a set of machines, where each job involves multiple operations and each operation can be carried out by only one machine. In addition, each machine can process only one job at a time. However, the canonical JSP is restricted by factors related to having a rigid model, while flexible job shop scheduling resembles actual job shop circumstances more accurately.¹ Compared to the traditional JSP, the flexible job shop problem (FJSP) is a practically useful extension of the JSP in which each operation should be assigned to a set of alternative machines, with the result that the FJSP is much more complicated than the traditional JSP.² Because the JSP is strongly NP-hard, the FJSP is strongly NP-hard as well.³

In view of the fact that FJSP can effectively improve the production efficiency of the job shop and shorten the manufacturing time, multiple heuristic or meta-heuristic algorithms have been developed to optimize the problem. For example, Pezzella et al.⁴ introduced a genetic algorithm (GA) framework integrating different strategies for developing efficient algorithms for the problem effectively. Girish and Jawahar⁵ described a powerful particle swarm optimization (PSO)-based heuristic algorithm for FJSP to minimize the makespan. Li et al.⁶ proposed a novel tabu search (TS) algorithm by combining an effective neighborhood structure and local search methods based on the critical block theory. Aiming for flexible job shop scheduling with the constraint of preventive maintenance activity,

¹School of Information Science and Engineering, Shandong Normal University, Jinan, PR China
²School of Computer, Liaocheng University, Liaocheng, PR China

Corresponding author:
Jun-qing Li, School of Information Science and Engineering, Shandong Normal University, Jinan 250014, PR China.
Email: lijunjing@lcu-cs.com

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
et al. formulated a multi-objective optimization problem, and environmental pollution problem, Wu et al. (B&B). With the consideration of the deterioration effect and proposed a parallel branch and bound algorithm. Most researchers have considered simultaneously in the FJSP with scheduling-layout problem. Liu et al. developed a combinational algorithm based on glowworm swarm optimization (GSO) algorithm and GA, hybrid with a heuristic strategy of green transmission that effectively solves the FJSP with crane transportation. Piroozfard et al. considered both the total late work criterion and the total emitted carbon footprint bi-objectives, proposed an improved multi-objective evolutionary algorithm (MOEA), and verified its efficiency in comparison with two other multi-objective algorithms. Li et al. combined several heuristic strategies and a TS algorithm, constructed a discrete artificial bee colony (DABC) algorithm, and addressed a scheduling problem that simultaneously minimizes makespan, total work load, and maximal workload. Soto et al. studied the multi-objective FJSP with the aim of minimizing the total workload, maximal workload and makespan, and proposed a parallel branch and bound algorithm (B&B). With the consideration of the deterioration effect and environmental pollution problem, Wu et al. formulated a multi-objective optimization model based on the energy consumption and step-deterioration effect model. Then, a multi-objective hybrid pigeon-inspired optimization (PIO) and a SA algorithm are developed to solve this problem. Ebrahimi et al. investigated an energy-aware model to minimize the energy consumption and tardiness penalty simultaneously in the FJSP with scheduling-layout problem, and four meta-heuristic algorithms are introduced in the study.

Literature review shows that several assumptions in canonical FJSP being considered are unreasonable, for example, each operation can be processed after it is finished on the previous machine. Most researchers have omitted the intermediate process of transportation time between two machines while the transportation of jobs really requires the participation of cranes, automatic guided vehicles or other tools, which is an essential characteristic of actual industrial scheduling. Few studies have conducted such investigations. Yu et al. presented an imperialist competition algorithm for FJSP in physical examination system. Rossi and Dini introduced the artificial immune algorithm (AIA), inspired by biological immune system, was proposed as a new intelligent approach by Castro et al. Compared with other heuristic algorithms, the AIA makes use of the advantages of generating diverse solutions to maintain diversity of the population and an immune memory mechanism, thereby overcoming the inevitable premature problem of other heuristic algorithms in the process of optimization in order to obtain the optimal solution. The AIA has been widely used owing to its efficient global search capability, and many hybrid meta-heuristic algorithms have also appeared. Bagheri et al. integrated several initial strategies and mutation operators of antibodies, successfully applying the AIA to solve the FJSP. Lin and Ying introduced a hybrid algorithm of AIA and SA to effectively solve the blocking flow shop scheduling problem. Savsani et al. presented four variants of hybrid metaheuristic optimization algorithms that combine the characteristics of biogeography-based optimization (BBO) with AIA and ACO, and verified their effectiveness on many benchmark problems. Zeng and Wang embedded PSO and SA algorithm into the framework of AIA and proposed a hybrid meta-heuristic algorithm to solve the problem with multiple objectives. Roshanaei et al. constructed two mathematical models including sequence-based and position-based, and proposed a hybrid meta-heuristic algorithm based on the of AIA and SA. With the consideration of fuzzy processing time of realistic system, Li et al. proposed an improved version of AIA, designed four initialization
heuristic algorithms, and embedded SA to enhance the exploitation ability.

At present, there are mainly three approaches to solve the multi-objective optimization problems.\textsuperscript{52} Multi-objective evolutionary algorithms based on decomposition (MOEA/D)\textsuperscript{33,34} including the algorithms based on weight decomposition,\textsuperscript{6,35,36} Chebyshev approach,\textsuperscript{24} etc. The non-pareto approach, in which different operators are used in a separated way. The Pareto approach, a method that is based on the pareto dominance relation, in which solutions converging to the Pareto front.\textsuperscript{37–39} In this study, an approach based on the weight decomposition is used to optimize this problem in order to transform the multi-objective problem into a single-objective problem and reduce the complexity of the algorithm.

Among the modern meta-heuristic algorithms, the AIA and SA algorithms have become an effective and efficient algorithm to solve combinational optimization problems.\textsuperscript{53} In this study, a hybrid meta-heuristic algorithm is proposed to solve the current problem, using a combination of the AIA and a SA algorithm, taking the maximum completion time (makespan) and energy consumption as the objective function, and assigning different weight coefficients to the objectives. In the proposed combinational meta-heuristic algorithm, an information entropy theory is applied to obtain better antibodies and some potential high-quality antibodies are searched from the point of view of global search. At the same time, four kinds of mutation operators which based on machine selection and operation sequence parts are utilized to perform further local searches in the neighborhood structure, and the Taguchi method are used to calibrate parameters.\textsuperscript{40} To testify the availability of the model, the proposed MILP model is constructed by CPLEX.

The outline of this paper is organized as follows. The FJSP considering transportation time and energy consumption and a sequence-based mixed integer linear model are introduced in section “Problem description and mathematical modeling.” In section “IAIA,” an improved artificial immune algorithm (IAIA) is proposed to solve this problem. The parameters and experimental results are shown in section “Experiment analyses.” The section “Conclusion and future research” draws conclusions and describes directions for future work.

Problem description and mathematical modeling

Problem description

The FJSP-T model is improved over the canonical FJSP, adding constraints of transportation time and total energy consumption generated during the scheduling process. Accordingly, two objectives are of concern: (1) minimize the makespan; and (2) minimize the energy consumption. The weighted sum of makespan and total energy consumption is used as the target of optimization. To take the transportation time of jobs from assembly line to processing machine into consideration, we assumed an additional virtual machine which called machine zero in place of the assembly line. The FJSP-T must satisfy the following assumptions:

- Machines cannot process multiple jobs simultaneously.
- All of the jobs are ready when the process is started.
- When entire process begins, operations cannot be interrupted until it is completed.
- Each operation of a job is handled by only one machine.
- Sufficiently many cranes are available to transfer jobs from one machine to another machine.
- Each machine has a sufficient buffer to store jobs.
- The energy consumption of cranes is ignored.

Mathematical modeling

In this paper, we propose a mathematical model based on the FJSP with energy consumption and transportation time, which can be defined as follows. There is a set of jobs \( \{I_1, I_2, \ldots, I_n\} \) to be executed on a set of machines \( \{M_1, M_2, \ldots, M_m\} \) in which each job \( I_i \) consists of a sequence of \( \text{op}_{ij} \) operations and each operation \( O_{ij} \) is applied to on one of a subset of eligible machines \( M_{ij} \subseteq M \). In terms of transportation time, every operation \( O_{ij} \) of each job \( I_i \) has its own transportation time \( t_{i,k,k'} \) of transferring from machine \( k' \) to another machine \( k \). In addition, the energy consumption generated when jobs are processed on machines is taken into account. When jobs are being processed on machines, they produce disparate energy consumption \( e_{ci,j} \) per unit time. It worth noting that all the machines are assumed to be the same, and variables given in the mathematical model, that is, processing time of each operation, transportation time and unit energy consumption, are fixed and not affected by the state of machines. The parameters defined in this model are showed as follows.

\[
\begin{align*}
\text{min } & \alpha \cdot C_{\text{max}} + (1-\alpha) \cdot TEC \\
TEC &= \sum_{i=1}^{n} \sum_{j=1}^{\text{op}_i} \sum_{k=1}^{m} \sum_{k'=1}^{m} Y_{i,j,k,k'} \times p_{i,j,k} \times e_{ci,k} \\
&\text{subject to:} \\
&\sum_{k=1}^{m} Y_{i,j,k,k'} = 1 \quad \forall i,j > 1 \\
&\sum_{k=1}^{m} Y_{i,1,k,0} = 1 \quad \forall i \\
&\sum_{k'=1}^{m} Y_{i,j,k,k'} \leq e_{i,j,k} \quad \forall i,j > 1, k \\
&Y_{i,1,k,0} \leq e_{i,1,k} \quad \forall i,j 
\end{align*}
\]
\begin{equation}
Y_{i,j,k,k'} \leq \sum_{k''=1}^{m} Y_{i,j-1,k'',k'} \quad \forall i,j > 2, k, k' \tag{6}
\end{equation}

\begin{equation}
Y_{i,2,k,k'} \leq Y_{i,1,k',0} \quad \forall i, k, k' \tag{7}
\end{equation}

\begin{equation}
C_{i,j} \geq C_{i,j-1} + \sum_{k=1}^{m} \sum_{k'=1}^{m} Y_{i,j,k,k'} (p_{i,j,k} + u_{i,k',k}) \quad \forall i, j > 1 \tag{8}
\end{equation}

\begin{equation}
C_{i,1} \geq \sum_{k=1}^{m} Y_{i,1,k,0} (p_{i,1,k} + u_{i,0,0}) \quad \forall i \tag{9}
\end{equation}

\begin{equation}
C_{i,j} \geq C_{i,f} + p_{i,j,k} - B(1 - X_{i,j,f,f}) \tag{10}
\end{equation}

\begin{equation}
C_{i,f} \geq C_{i,1} + p_{i,1,k} - B(1 - X_{i,j,k',1}) \tag{11}
\end{equation}

\begin{equation}
\forall i \neq i', j > 1, f > 1, k \tag{12}
\end{equation}

\begin{equation}
C_{i,j} \geq C_{i,f} + p_{i,j,k} - B(1 - X_{i,j,k',1}) \tag{13}
\end{equation}

\begin{equation}
C_{i,f} \geq C_{i,1} + p_{i,f,k} - B(X_{i,j,f,f}) \tag{14}
\end{equation}

\begin{equation}
\forall i \neq i', j > 1, f > 1, k \tag{15}
\end{equation}

\begin{equation}
Y_{i,j,k,k'} \leq Y_{i,j-1,k'',k'} \quad \forall i,j > 2, k, k' \tag{6}
\end{equation}

\begin{equation}
Y_{i,2,k,k'} \leq Y_{i,1,k',0} \quad \forall i, k, k' \tag{7}
\end{equation}

\begin{table}
\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
Jobs & Operations & Machines & \\
\hline
& & $M_1$ & $M_2$ & $M_3$ \\
\hline
\text{Job1} & $O_{1,1}$ & 13 & 9 & - \\
& $O_{1,2}$ & 15 & - & 8 \\
& $O_{1,3}$ & 7 & 4 & 13 \\
\hline
\text{Job2} & $O_{2,1}$ & - & 12 & 9 \\
& $O_{2,2}$ & 9 & - & 12 \\
& $O_{2,3}$ & 15 & 10 & 6 \\
\hline
\text{Job3} & $O_{3,1}$ & - & 10 & - \\
& $O_{3,2}$ & 8 & 12 & - \\
& $O_{3,3}$ & 14 & - & 9 \\
\hline
\end{tabular}
\end{center}
\caption{Processing time when operations are processed in different machines.}
\end{table}

\textbf{Example of the FJSP-T model}

There gives a typical example to illustrate the FJSP with the constraint of transportation time, the processing time is shown in Table 1, in which the symbol “-” indicates that the machine does not have ability to process the job. The unit energy consumption generated by each job during the processing of each machine is listed in Table 2. Because the $M_0$ machine is a virtual machine that represents the assembly line, the energy consumption generated by $M_0$ is zero, as is the related transportation time. The transportation time required for each job to be moved from one machine to another is listed in Table 3, as shown in Figure 1, where $O_{1,i}$ is the first operation of the first job. Once the process starts, $O_{1,i}$ is first transferred to $M_1$ and then transferred to $M_2$ for further processing. Since $O_{2,i}$ has finished processing at this time, some idle time appears. Other operations follow the identical process. The result is an optimal solution in which once an operation is completed, the job
begins to be transported immediately to another machine in prepare for further processing, as illustrated in Figure 1.

Table 2. Unit energy consumption when jobs are processed in each machine.

| Jobs | Machines |
|------|-----------|
|      | $M_0$ | $M_1$ | $M_2$ | $M_3$ |
| job1 | 0     | 0.5   | 0.8   | 0.7   |
| job2 | 0     | 0.6   | 1.0   | 0.9   |
| job3 | 0     | 0.7   | 0.5   | 0.8   |

Table 3. Transportation time between different machines.

| Jobs | Machines | Machines |
|------|----------|----------|
|      | $M_0$ | $M_1$ | $M_2$ | $M_3$ |
| job1 | $M_0$ | 0     | 4     | 5     | 1     |
|      | $M_1$ | 0     | 5     | 2     | 3     |
|      | $M_2$ | 0     | 2     | 3     | 0     |
|      | $M_3$ | 0     | 1     | 0     | 3     |
| job2 | $M_0$ | 0     | 6     | 2     | 5     |
|      | $M_1$ | 0     | 3     | 1     | 2     |
|      | $M_2$ | 0     | 3     | 0     | 5     |
|      | $M_3$ | 0     | 3     | 6     | 0     |
| job3 | $M_0$ | 0     | 2     | 3     | 4     |
|      | $M_1$ | 0     | 0     | 3     | 6     |
|      | $M_2$ | 0     | 5     | 0     | 2     |
|      | $M_3$ | 0     | 3     | 6     | 0     |
the $O_{2,1}$. There are two available machines that can implement the operation $O_{2,2}$ and the first machine is selected, resulting in the second gene position of the MS part to be one. Other numbers have the respective same meanings. In the OS part, each job $I_i$ appears $opi$ times, where $opi$ indicates the number of operations of job $I_i$, and the order in which each job emerges determines the operation it is. The processing sequence illustrated in Figure 2 is $O_{3,1}$–$O_{1,1}$–$O_{2,1}$–$O_{3,2}$–$O_{2,2}$–$O_{2,3}$–$O_{1,2}$–$O_{1,3}$–$O_{3,3}$. The total length of the chromosome is $2^3 P_{N_j=1} op_j$.

To maintain the diversity of the population and enable coding scheme to obtain the optimal solution quickly, global selection (GS), location selection (LS), and random selection (RS) are applied to the generated MS part. These three encoding schemes, all designed by Karimi et al., require no repair mechanism thereby substantially reducing the time complexity. First, each job $I_i$ is assigned a different integer in the interval $[1, N]$ representing the priority of each job to be processed. Then, each operation is allocated to a feasible machine in turn. Meanwhile, the operation-selection part is initialized using RS. For the MS part, all of the jobs are assigned randomly, and each operation is arranged successively to a feasible machine respectively, with number $k$ indicating that the $k$th available machine is assigned to corresponding operation. For the OS part, all of the operations are disorganized and inserted into the chromosome.

**Affinity calculation**

In this study, evaluation criteria of antibodies are determined by affinity between antibody and antigen ($aff$) and individual concentration ($den$). After evaluation, only antibodies with higher expected reproduction probability ($ERP$) will be subject to immune operation, where $ERP$ is the probability that a selected antibody being selected. The formula for calculating the $ERP$ is as follows:

$$ERP = \beta \frac{aff_p}{\sum aff_p} + (1 - \beta) \frac{den_p}{\sum den_p} \quad (21)$$

where $aff_p$ is the affinity between antibody $p$ and the antigen, $den_p$ is the individual concentration of the antibody $p$, and the variable $\beta$ is used to determine their respective weights. After normalization, antibodies with higher affinity and lower concentration can obtain better values of $ERP$.

**Affinity of antigen.** Each antibody has its own affinity for the antigen, which is related to the maximum completion time ($C_{max}$). To calculate the affinity between antibody and antigen, the following equation is used:

$$aff_p = \frac{1}{C_{max}(P)} \quad (22)$$

This equation, makes it clear that the affinity of each antibody is determined by the makespan and that a greater affinity of the antibody corresponds to a lower makespan.

**Antibody concentration.** To decrease the number of the same antibodies and increase the diversity of the population, the proposed algorithm introduces affinity between antibody and other antibodies in the population. We calculate the particular affinity based on the immunity and entropy theory presented by Cui et al. For each antibody, the number of antibodies that resemble another antibody divided by the total number of antibodies is the concentration of an antibody. More similar antibodies in the population, results in a higher concentration of an antibody and the more expected reproduction probability of an antibody, resulting in a degrading of the $ERP$. The calculation of antibody concentration is detailed as follows:
Step 1: Information entropy theory of antibodies

The evaluation of antibody concentration through information entropy is essentially to calculate the number of antibodies with the value being two in general, representing average information entropy of the two antibodies. Assuming that there are \( L \) genes in a chromosome, the information entropy of each gene is calculated using equation (23):

\[
H_{a,b}(\eta) = \frac{1}{L} \sum_{k=1}^{L} H_{a,b,k}(\eta)
\]

(23)

where \( \eta \) is the number of antibodies with its value being one in general, representing average information entropy of the two antibodies. Assuming that there are \( L \) genes in a chromosome, the information entropy of each gene is calculated using equation (24):

\[
H_{a,b,k}(\eta) = \sum_{n=1}^{S} p_{n,k} \ln(p_{n,k})
\]

(24)

where \( p_{n,k} \) is the ratio that the \( n \)th antibody emerges at the \( k \)th locus, with \( p_{n,k} = \text{(total number of occurrences of the antibody at the }k\text{th position among individuals)}/\eta \).

Step 2: Similar extent of antibodies

The similar extent of each antibody \( (\Pi_{a,b}) \) represents the similarity between individual \( a \) and \( b \) and it determined by the average information entropy between two antibodies. The similarity of an antibody is calculated as follows:

\[
\Pi_{a,b} = \frac{1}{1 + H_{a,b}(2)}
\]

(25)

\[
d_{a,b} = \begin{cases} 1 & \Pi_{a,b} \geq \lambda \\ 0 & \text{otherwise} \end{cases}
\]

(26)

where \( H_{a,b}(2) \) is the common case in which \( \eta \) equals two, representing the average information entropy between individual \( a \) and individual \( b \). To calculate the antibody similarity, an intermediate variable \( \Pi_{a,b} \) and a binary variable \( d_{a,b} \) are introduced. A greater \( H_{a,b}(2) \) value between two antibodies, corresponds to a smaller \( \Pi_{a,b} \) value between them. \( \Pi_{a,b} \) ranges from zero to one, and \( d_{a,b} \) is equals to one if the \( \Pi_{a,b} \) value exceeds the threshold \( \lambda \); otherwise, it is zero.

Step 3: Density of antibody

The density of an antibody, denoted by \( den_p \), is the rate of the number of antibodies that similar to antibody \( p \) in the population, which is defined as follows:

\[
den_p = \frac{1}{N} \sum_{k=1}^{N} d_{p,k}
\]

(27)

where \( d_{p,k} \) denotes the summary between the antibody \( p \) and the other antibodies, and \( N \) denotes the size of population.

Selecting and cloning

To search better antibodies, \( N_c \) antibodies with the highest affinity are selected in the initial population to undergo the clone operation. The clone number of each selected antibody can be defined as: \( N_c = k + 1 \), in which \( k \) indicates that this antibody is the \( k \)th antibody with the highest affinity. Accordingly, \( N_c (N_c + 1) \) clones are generated in preparation for next step.

Mutation

Mutation aims to search its neighboring solution by altering one or more genes on the chromosome. The chromosome is composed of MS part and OS part. Thus, we apply different mutation operators in each part of each clone antibodies.

For the MS part, the random rule is used to change the sequence of the part. Two genes are randomly selected and replaced with other available machines. The new genes are also the serial numbers in the collection of available machines, rather than the number of the current machine. Considering the example in Figure 2, the MS mutation operator is shown in Figure 3(a), with the gene at the third position, as the third operation of \( J_2 \), is replaced by three in the collection of available machines that can process \( O_{2,3} \).

For another OS, three different canonical mutation operators are used to search other optimal solutions in the neighbor field.

(1) Swap operator: To form a new neighborhood solution, two different gene positions are selected randomly in the current solution to be exchanged at the selection positions.

(2) Randomly insert operator: A gene position is randomly selected and shifted it into a previous position.

(3) Reordering operator: We randomly select two different gene positions on the current chromosome and reorder the genes between the two selected positions.

To clarify the process, three OS mutation operators are shown in the Figure 3(b)–(d).

Update antibodies population

The clonal suppression operator selects the cloned and mutated antibodies, with lower-affinity antibodies are inhibited, leaving high affinity antibodies in a new antibody population. In this study, after the cloning and mutation processes, the selected high-quality antibodies are mixed with the cloned antibodies into a temporary antibody population. Then, the population is arranged in descending order according to the criterion of ERP, in which the \( N_c \) antibodies with the highest affinity is selected to replace the antibody with the lowest affinity in the initial population.

However, to avoid being trapped in local optimization, \( N_c \) antibodies with the highest affinity in the
The population are mutated through SA. The SA algorithm is an efficient heuristic algorithm. With its strong local search ability, it has been effectively applied to kinds of job shop scheduling problems combined with multiple optimization algorithms. In this study, on the basis of the original SA algorithm, a memory function is added to save the best solution obtained during search process and change the annealing rate. For each antibody, a gene in the MS part is selected randomly selected and replaced with another available machine, and two genes randomly selected in the OS part are exchanged with each other. The mutated antibody replaces the previous one directly if the mutated antibody is better, and whether the best antibody needs to be updated is determined. Otherwise, the solution is accepted with a random probability, following the Metropolis criterion described in equation (28):

\[ P = \begin{cases} 
1, & F(S_{\text{new}} < S_{\text{old}}) \\
\exp(-\Delta(T)), & F(S_{\text{new}} > S_{\text{old}}) 
\end{cases} \]

in which the probability of replacement is gradually reduced by temperature decline. When the system is at a higher temperature, the acceptance of the inferior solution is close to one, but the system no longer receives any inferior solutions when the temperature approaches \( T_0 \).

Compared to the traditional SA algorithm, in which the annealing rate is typically a fixed constant ranging from 0.75 to 0.95, therefore, the search of the algorithm at low temperatures is very slow and insufficient. Dai et al.\(^{44}\) improved the decline rate in accordance with the Hill function, after the completion of each isothermal search, the annealing using the following calculation criterion:

\[ T_t = T_{t-1} \times k_{\text{new}} \]

in which the \( t \) is the number of the annealing operation, and \( n \) represents is the Hill coefficient, whose value is typically greater than one. Consequently, the SA can be searched fully in the space of the solution to achieve the optimal effect.

**Experiment analyses**

In this chapter, the proposed IAIA is applied to solve the FJSP-T using C++. To testify the efficiency of the algorithm, the experimental tests are conducted on a PC with a 2.40-GHz Intel Core i7 CPU and 8 GB of RAM. Weight decomposition has been proved to be an effective way to optimize multi-objective problems. In this study, the proposed algorithm based on weight decomposition is used to optimize the FJSP-T model. In the part of proving the effectiveness of information entropy theory and SA algorithm, the coefficient of makespan \( \alpha \) is set to 0.8, and the coefficient of total energy consumption is set to 0.2 because we prefer to take the completion time as the principal optimization objective. Then, in the section of multi-algorithm comparison, the performance of proposed algorithm is analyzed under three sets of weight coefficients, that is, 0.2, 0.5, and 0.8.

**Instances**

We randomly generated 30 instances based on the actual processing circumstances of a factory. The instances are divided into two parts, of which the first represents the job scale with values ranging from 10 to 100 and the number of jobs being \( n \in \{10, 20, 30, 50, 100\} \), and the second represents the total number of machines, with the value \( m \in \{3, 5, 6, 7, 8, 10\} \).
According to the characteristics of the FJSP, the number of operations contains a degree of randomness, with values randomly ranging from \(m/2\) to \(m\) when the number of machines exceeds five. For example, the instance 10-5 involves 10 jobs processed on five machines, with the number of operations of jobs evenly valued within the interval \([m/2, m]\). In addition, the processing time of each operation is set to the interval \([10, 20]\), and the transportation time between two machines is set to \([5, 20]\), and the unit energy consumption when machines are processing is set to \([0.5, 1]\). In actual production, for each operation, machines can be out of order or in preventive maintenance. Consequently, we randomly set up some machines with no processing capability with the value ranging from one to half number of the machines for each operation.

### Parameter selection of the algorithm

Some key parameters in IAIA are evaluated because the final experimental effect will be affected by parameter selection, which involves some significant parameters. The selection of parameters determines the convergence speed and effect of the algorithm. Accordingly, four of the more critical parameters are selected to process the parameters adjustment experiment, with other parameters being selected from previous experience or other papers, as follows:

1. Population size \(\text{Popsize} = 100\)
2. Ratio of initial assignment with GS: 60%
3. Ratio of initial assignment with LS: 30%
4. Ratio of initial assignment with RS: 10%
5. Original temperature \(T_0 = n 	imes m\)
6. Final temperature \(T_f = 10\)
7. Iteration number of each temperature \(L = 100\)
8. Hill function \(t = 2\)
9. Selected probability of each operator \(P_m = 1/3\)

There are four significant parameters we considered in this study, that is, the number of selected antibodies with higher affinity \((N_c)\), the weight of the objective \((\beta)\), the concentration threshold \((\lambda)\), and the annealing rate \((K)\). The Taguchi method of Design of Experiment (DOE) was applied to verify the effect of these four parameters on the performance of the algorithm. For each parameter, we considered four levels, as shown in Table 4. A set of orthogonal arrays \(L_{16}(4^5)\) was constructed to combine the parameters, and for each combination, which runs 30 times independently, the average value obtained is taken as the response value \((\text{RV})\). Table 5 shows all of the combinations of the four parameters and their corresponding RV values.

The instance 50-8 was used to evaluate each parameter of each combination. We show the factor level trend of each parameter in the Figure 4. According to this analysis, we can infer that \(N_c = 6\), \(\beta = 0.8\), \(\lambda = 0.90\), \(K = 0.99\) comprise the best set of parameters.

### Comparison with the exact solver CPLEX

To certify the validity of the FJSP-T model, we coded the problem formula in IBM ILOG CPLEX 12.7 and

| Variables | Values |
|-----------|--------|
| \(N_c\)   | 4      | 6      | 8      | 10     |
| \(\beta\)  | 0.5    | 0.6    | 0.7    | 0.8    |
| \(\lambda\)| 0.90   | 0.91   | 0.92   | 0.93   |
| \(K\)      | 0.9    | 0.95   | 0.97   | 0.99   |

| Number | Control factor level | RV       |
|--------|----------------------|----------|
| 1      | 1                    | 1        | 1       | 1230.92 |
| 2      | 1                    | 1        | 2       | 1232.77 |
| 3      | 1                    | 2        | 2       | 1232.89 |
| 4      | 1                    | 2        | 3       | 1235.67 |
| 5      | 2                    | 1        | 1       | 1229.07 |
| 6      | 2                    | 1        | 2       | 1227.66 |
| 7      | 2                    | 3        | 1       | 1236.97 |
| 8      | 2                    | 4        | 2       | 1228.71 |
| 9      | 3                    | 1        | 3       | 1230.93 |
| 10     | 3                    | 2        | 2       | 1247.56 |
| 11     | 3                    | 3        | 2       | 1229.52 |
| 12     | 3                    | 4        | 1       | 1233.04 |
| 13     | 4                    | 1        | 2       | 1245.69 |
| 14     | 4                    | 2        | 2       | 1237.24 |
| 15     | 4                    | 3        | 2       | 1240.68 |
| 16     | 4                    | 4        | 1       | 1232.59 |

| Algorithm 1. Improved artificial immune algorithm (IAIA) |
|--------------------------------------------------------|
| **Input:** FJSP data set, AIA parameters               |
| **Output:** a near-optimal solution                     |
| **Begin**                                             |
| **Initialize**                                        |
| a) Initialize the parameters                          |
| b) Initialize the population with the MSOS chromosome representation |
| **Evaluate** each antibody in the current population by calculating the its affinity with the antigen |
| **While** (not termination condition) **do**           |
| Evaluate the affinity of each antibody                 |
| Select the best \(N_c\) antibodies with the highest affinity |
| Clone \(N_c\) antibodies from the selected antibodies |
| Exert mutation on the \(N_c(N_c+1)\) cloning antibody   |
| Calculate the ERP of the mutated antibodies and sort them with the cloning antibodies in descending order |
| Update the original population through the suppression process |
| Mutate the best \(N_c\) antibodies with the highest affinity in the population using SA algorithm |
| **End**                                               |
| Output a near-optimal solution                         |
| **End**                                               |

To certify the validity of the FJSP-T model, we coded the problem formula in IBM ILOG CPLEX 12.7 and
ran various small-scale instances. Meanwhile, to verify the efficiency of the IAIA further, CPLEX was used as the benchmark for performing a comparison with the IAIA. Because CPLEX uses an exact algorithm based on branch and bound, calculating an accurate result requires substantial time. Consequently, the total computing time of CPLEX is limited to 1 h, and the number of threads is set to three. Each algorithm calculates the relative percentage increase (RPI) based on the best value, using the following equation:

$$RPI = \frac{(f_{Current} - f_{Best})}{f_{Best}} \times 100$$  \hspace{1cm} (31)$$

where $f_{Current}$ is the fitness value of the current algorithm that we compared to, and $f_{Best}$ is the best fitness value obtained from all of the given algorithms.

As shown in Table 6, the scale of each instance is listed in the first column, in which all the instances are generated randomly following the above generation rule. The best values selected from two algorithms are listed in the second column. The following two columns display the fitness values of two comparison algorithms, and the RPI values are displayed in the last two columns. In the illustrated 15 small instances, the proposed IAIA is able to achieve better results in most instances.

**Efficiency of the local search**

In this subsection, the SA algorithm is verified as to whether it can optimize the solution effectively. Then, the experiment is tested between the IAIA and the IAIA without the SA algorithm, denoted as IAIA-NSA.

As shown in Table 7, the scale of instances is listed in the first column, with the best value obtained from the comparison between two algorithms is displayed in the second column. The average fitness value after 30 independent runs and the RPI value obtained from two

---

**Algorithm 2. Simulated annealing (SA) algorithm**

**Input:** an initial solution,

**Output:** a near optimal solution

Begin
- Initialize the original temperature $T_0$, the final temperature $T_f$, the iteration number of each temperature $L$, and the annealing rate $K$.
- Set $T = T_0$
- While ($T > T_f$) do
  - For $i = 1$ to $L$ do
    - Generate the neighborhood solution $S_{new}$ from $S_{old}$
    - Evaluate the improvement of performance criterion
      - $\text{function delta} = F(S_{new}) - F(S_{old})$
      - if ($\text{random}(0,1) < e^{-\text{delta}/T}$) then
        - $S_{old} = S_{new}$
      - else
        - $S_{old} = S_{new}$
    - End if
  - Update the new annealing rate
- End while
End

---

| Instance | Best | Fitness IAIA | Fitness CPLEX | RPI IAIA | RPI CPLEX |
|----------|------|--------------|---------------|---------|-----------|
| 5-2      | 148.74 | 148.74       | 148.74        | 0.00    | 0.00      |
| 5-3      | 118.96 | 118.96       | 118.96        | 0.00    | 0.00      |
| 5-4      | 163.10 | 163.1        | 163.74        | 0.00    | 0.39      |
| 6-2      | 168.62 | 168.62       | 168.62        | 0.00    | 0.00      |
| 6-3      | 161.86 | 161.86       | 161.86        | 0.00    | 0.20      |
| 7-2      | 192.52 | 192.52       | 193.32        | 0.00    | 0.42      |
| 7-3      | 178.78 | 178.78       | 178.78        | 0.00    | 0.00      |
| 7-4      | 193.06 | 193.06       | 193.48        | 0.00    | 0.22      |
| 8-2      | 200.04 | 200.04       | 200.04        | 0.00    | 0.00      |
| 8-3      | 176.92 | 176.92       | 176.92        | 0.21    | 0.00      |
| 8-4      | 202.58 | 202.58       | 203.6         | 0.00    | 0.50      |
| 9-2      | 255.88 | 255.88       | 255.88        | 0.00    | 0.00      |
| 9-3      | 229.26 | 229.26       | 229.26        | 0.00    | 0.00      |
| 9-4      | 224.42 | 224.42       | 224.74        | 0.00    | 0.14      |
| Average  | 185.00 | 185.02       | 185.23        | 0.01    | 0.13      |

The bold emphasis means the better values.

---

Figure 4. Factor level trend of parameters.
algorithms are displayed in other columns. It can be observed that: (1) in the illustrated 30 instances, the IAIA obtained 22 optimal values and performed better than the IAIA-NSA, and (2) from the last row, the average RPI value obtained by IAIA is less than that of IAIA without the SA algorithm, which proves the efficiency of this improvement.

To testify the efficiency of the IAIA, a multifactor analysis of variance (ANOVA) is applied to describe the significant difference between two algorithms. The means and the 95% least-significant difference (LSD) interval are shown in Figure 5. The \( p \)-value, 1.1794e-07, is much less than 0.05, and that shows there are significant differences between two algorithms. Therefore, we can conclude that the proposed IAIA is significantly improved over the IAIA-NSA. The proposed algorithm has been greatly improved as a result of the utilization of the improved SA algorithm.

### Efficiency of the information entropy strategy

To testify the efficiency of the information entropy strategy, a method not using the information entropy strategy, which denoted as IAIA-NS, is described in this section. In this method, the antibody population of cloned antibodies and selected \( N_c \) antibodies is sorted according to affinity value between antibodies and antigens, rather than the ERP. Both the IAIA and IAIA-NS algorithms were implemented 30 times independently for 30 s each time. The minimum value obtained after 30 runs and the RPI values between the two algorithms are shown in the Table 8.

| Instance | Best Fitness | RPI | Best Fitness | RPI |
|----------|--------------|-----|--------------|-----|
| 10-3     | 152.44       | 0.00 | 152.44       | 0.00 |
| 10-5     | 175.70       | 1.35 | 175.70       | 1.35 |
| 10-6     | 204.96       | 0.30 | 204.96       | 0.30 |
| 10-7     | 238.14       | 3.46 | 238.14       | 3.46 |
| 10-8     | 276.86       | 0.59 | 276.86       | 0.59 |
| 10-10    | 363.80       | 0.46 | 363.80       | 0.46 |
| 20-3     | 240.22       | 0.64 | 240.22       | 0.64 |
| 20-5     | 330.22       | 2.08 | 330.22       | 2.08 |
| 20-6     | 416.30       | 0.12 | 416.30       | 0.12 |
| 20-7     | 390.40       | 2.04 | 390.40       | 2.04 |
| 20-8     | 470.04       | 1.15 | 470.04       | 1.15 |
| 20-10    | 612.10       | 3.05 | 612.10       | 3.05 |
| 30-3     | 371.18       | 0.00 | 371.18       | 0.00 |
| 30-5     | 459.42       | 0.20 | 459.42       | 0.20 |
| 30-6     | 642.84       | 1.44 | 642.84       | 1.44 |
| 30-7     | 561.66       | 2.50 | 561.66       | 2.50 |
| 30-8     | 704.28       | 3.18 | 704.28       | 3.18 |
| 30-10    | 848.20       | 0.00 | 848.20       | 0.00 |

The bold emphasis means the better values.
The illustrated 30 instances shown in Table 8 shows that (1) the IAIA obtained 21 better solutions while IAIA-NS obtained only nine better solutions out of the 30 given instances; and (2) the average RPI value obtained by IAIA is less than that of IAIA without the strategy of information entropy, which proves the efficiency of the strategy.

Meanwhile, ANOVA was conducted with the confidence interval shown in Figure 6, with the obtained p-value is 0.0009, which is far less than 0.05. In conclusion, the application of the information entropy strategy performs better in solving this problem.

**Comparison with other efficient algorithms**

To verify the proposed algorithm has a great improvement, three other algorithms were selected to compare with the IAIA (1) the imperialist competitive algorithm,16 (2) the enhanced GA,19 and (3) the variable neighborhood search algorithm.45 The first and second comparison algorithms have been proved to have significant advantages in solving this problem, and the variable neighborhood search algorithm is also widely accepted because of its effective local search ability. Consequently, the advantages and competitive performance are better highlighted. Each algorithm runs 30 times independently and 30 s for each time, and the average value is used for comparison. In order to verify the performance of algorithms fairly, all the algorithms are implemented in accordance with the parameter, strategies and framework mentioned in the references, and the performance of each algorithm is compared.

![Figure 6. Means and 95% LSD interval for IAIA-NSA and IAIA.](image)

The bold emphasis means the better values.

**Table 9. Comparison with other meta-heuristic algorithms under $\alpha = 0.8$.**

| Instance | Best Fitness | ICA + LS | EGA | VNS | IAIA |
|----------|--------------|---------|-----|-----|------|
| 10-3     | 151.13       | 151.13  | 153.15 | 151.83 | 152.55 |
| 10-5     | 179.17       | 181.25  | 186.63 | 179.17 | 179.49 |
| 10-6     | 206.78       | 211.05  | 217.87 | 210.59 | 206.78 |
| 10-7     | 246.90       | 252.72  | 261.99 | 259.95 | 246.90 |
| 10-8     | 278.73       | 288.52  | 299.39 | 296.18 | 278.73 |
| 10-10    | 360.05       | 381.69  | 401.46 | 400.36 | 360.05 |
| 20-3     | 241.80       | 241.81  | 244.54 | 241.80 | 242.55 |
| 20-5     | 333.71       | 342.68  | 356.46 | 343.97 | 333.71 |
| 20-6     | 411.49       | 434.67  | 450.51 | 441.36 | 411.49 |
| 20-7     | 384.35       | 409.66  | 427.92 | 421.30 | 384.35 |
| 20-8     | 448.63       | 490.67  | 511.13 | 503.14 | 448.63 |
| 20-10    | 582.99       | 640.55  | 667.03 | 655.78 | 582.99 |
| 30-3     | 371.54       | 374.48  | 376.21 | 371.54 | 373.60 |
| 30-5     | 457.51       | 469.58  | 486.12 | 464.35 | 457.51 |
| 30-6     | 640.00       | 677.05  | 702.44 | 678.60 | 640.00 |
| 30-7     | 550.59       | 585.43  | 610.72 | 597.99 | 550.59 |
| 30-8     | 677.83       | 739.58  | 771.37 | 752.18 | 677.83 |
| 30-10    | 810.50       | 882.49  | 905.97 | 886.57 | 810.50 |
| 50-3     | 582.53       | 601.05  | 598.56 | 582.53 | 586.15 |
| 50-5     | 747.41       | 763.98  | 783.79 | 757.58 | 747.41 |
| 50-6     | 979.18       | 1014.15 | 1039.07 | 1009.34 | 979.18 |
| 50-7     | 914.09       | 957.76  | 978.49 | 955.33 | 914.09 |
| 50-8     | 1164.66      | 1211.70 | 1241.20 | 1204.16 | 1164.66 |
| 50-10    | 1371.29      | 1424.66 | 1455.96 | 1410.93 | 1371.29 |
| 100-3    | 1098.01      | 1127.33 | 1135.77 | 1098.01 | 1110.22 |
| 100-5    | 1647.20      | 1663.82 | 1681.23 | 1647.20 | 1654.89 |
| 100-6    | 2137.72      | 2180.51 | 2204.13 | 2139.63 | 2137.72 |
| 100-7    | 1816.18      | 1845.26 | 1870.51 | 1826.68 | 1816.18 |
| 100-8    | 2379.81      | 2426.75 | 2465.67 | 2379.81 | 2383.35 |
| 100-10   | 2640.70      | 2688.32 | 2733.59 | 2640.70 | 2641.24 |
| Average  | 827.08       | 855.34  | 873.96 | 850.29 | 828.15 |

The bold emphasis means the better values.
under different weight coefficient. The experiment results are illustrated in the Tables 9 to 11, the average value for 30 runs for each instance of the four algorithms are displayed and the RPI values are obtained using the equation (31). As shown in Tables 9 to 11, we can observe that the proposed IAIA obtains the optimal values in most instances, obtaining RPI value equal to zero, with an absolutely advantage over the other algorithms. This shows the most effective performance in solving the FJSP-T model.

In order to visualize the results of the comparison, ANOVA is applied to evaluate the performance of each algorithm. As shown in Figure 7(a)–(c), which represents different ANOVA diagram under three sets of weight coefficients, that is, 0.8, 0.2, and 0.5, and the $p$-value under three different sets of weight coefficients are all far less than 0.05, showing that the four compared algorithms have significant differences. Eight instances are selected randomly, and their corresponding convergence curves are showed in the Figure 8(a)–(h). In the illustrated convergence curves, the initial solution of IAIA is superior to the other three algorithms through the application of three initialization strategies. Meanwhile, the IAIA shows effective global search and local search abilities. Figure 9 illustrates a Gantt chart of an optimal solution of the instance 20-8, and this solution optimized by IAIA is feasible and efficient.

We analyzed and summarized the reasons that the performance of the proposed algorithm is better than other three efficient algorithms as follows:

(1) we applied three initial strategies to ensure that we can obtain effective solution, and these initial strategies do not require the repair mechanism.

(2) the AIA is essentially an extension of the traditional GA and that not only has a powerful global search ability which but also improves the selection operator. To accelerate the convergence and avoid to fall in to local optimization, we introduced an operator of ERP that contains two parts, the calculation of fitness between antibodies and antigens, and the calculation of concentration between an antibody and other antibodies.

(3) to avoid to fall into the local optimization, the SA algorithm is applied to local search in each neighborhood space of solutions.

In conclusion, the proposed algorithm achieved great improvement for optimizing the FJSP regardless of the solution accuracy and the computational time.

### Table 10. Comparison with other meta-heuristic algorithms under $\alpha = 0.2$.

| Instance | Best Fitness | ICA + LS | EGA | VNS | IAIA | RPI |
|----------|--------------|---------|-----|-----|------|-----|
|          |              |         |     |     |      |     |
| 10-3     | 210.04       | 210.05  | 210.17 | 210.04 | 210.07 |
| 10-5     | 297.47       | 297.57  | 299.61 | 297.47 | 298.11 |
| 10-6     | 377.26       | 377.52  | 380.81 | 377.26 | 377.76 |
| 10-7     | 438.35       | 438.35  | 444.71 | 439.13 | 438.68 |
| 10-8     | 495.66       | 497.76  | 510.59 | 500.34 | 495.66 |
| 10-10    | 640.62       | 649.20  | 679.10 | 651.46 | 640.62 |
| 20-3     | 347.44       | 347.45  | 348.88 | 347.44 | 347.59 |
| 20-5     | 597.34       | 597.57  | 602.82 | 597.66 | 597.34 |
| 20-6     | 757.91       | 764.66  | 779.27 | 763.90 | 757.91 |
| 20-7     | 771.26       | 779.73  | 804.31 | 780.58 | 771.26 |
| 20-8     | 893.26       | 923.88  | 958.32 | 914.87 | 893.26 |
| 20-10    | 1163.51      | 1242.45 | 1292.42 | 1230.76 | 1163.51 |
| 30-3     | 575.20       | 575.20  | 575.42 | 575.20 | 575.20 |
| 30-5     | 843.15       | 846.51  | 855.45 | 843.98 | 843.15 |
| 30-6     | 1220.02      | 1246.10 | 1270.45 | 1233.33 | 1220.02 |
| 30-7     | 1059.06      | 1099.16 | 1137.65 | 1087.44 | 1059.06 |
| 30-9     | 1360.03      | 1448.41 | 1499.31 | 1413.17 | 1360.03 |
| 30-10    | 1642.19      | 1848.68 | 1900.91 | 1770.78 | 1642.19 |
| 50-3     | 882.87       | 883.73  | 884.68 | 882.87 | 882.96 |
| 50-5     | 1406.62      | 1431.41 | 1446.98 | 1413.01 | 1406.62 |
| 50-6     | 1822.82      | 1936.84 | 1963.43 | 1867.60 | 1822.82 |
| 50-7     | 1773.98      | 1938.17 | 1966.61 | 1840.64 | 1773.98 |
| 50-8     | 2306.79      | 2574.35 | 2615.38 | 2455.89 | 2306.79 |
| 100-3    | 2791.71      | 3168.18 | 3200.48 | 3036.74 | 2791.71 |
| 100-5    | 1657.71      | 1699.55 | 1668.83 | 1657.71 | 1658.41 |
| 100-6    | 2960.36      | 3186.95 | 3196.45 | 3025.99 | 2960.36 |
| 100-7    | 3998.18      | 4502.78 | 4487.97 | 4212.32 | 3998.18 |
| 100-8    | 3547.00      | 3981.59 | 4008.35 | 3796.74 | 3547.00 |
| 100-10   | 4916.04      | 5429.57 | 5452.52 | 5238.69 | 4916.04 |
| Average  | 1725.31      | 1859.60 | 1878.44 | 1803.88 | 1725.39 |

The bold emphasis means the better values.
Table 11. Comparison with other meta-heuristic algorithms under $\alpha = 0.5$.

| Instance | Best Fitness | RPI |
|----------|--------------|-----|
|          | ICA + LS | EGA | VNS | IAIA | ICA + LS | EGA | VNS | IAIA |
| 10-3     | 182.99    | 182.99 | 183.38 | 183.00 | 183.41 | 0.00 | 0.21 | 0.01 | 0.23 |
| 10-5     | 242.38    | 243.40 | 249.01 | 242.38 | 244.38 | 0.42 | 2.66 | 0.00 | 0.82 |
| 10-6     | 296.43    | 298.66 | 305.87 | 296.43 | 296.49 | 0.75 | 3.09 | 0.00 | 0.02 |
| 10-7     | 345.91    | 350.27 | 364.21 | 351.29 | 345.91 | 1.24 | 5.03 | 1.53 | 0.00 |
| 10-8     | 392.99    | 397.28 | 416.04 | 404.06 | 392.99 | 1.08 | 5.54 | 2.74 | 0.00 |
| 10-10    | 508.20    | 524.21 | 559.02 | 541.39 | 508.20 | 3.05 | 9.09 | 6.13 | 0.00 |
| 20-3     | 296.26    | 296.26 | 297.26 | 296.29 | 296.71 | 0.00 | 0.34 | 0.01 | 0.15 |
| 20-5     | 467.64    | 476.26 | 486.90 | 471.23 | 467.64 | 1.81 | 3.96 | 0.76 | 0.00 |
| 20-6     | 590.46    | 610.20 | 634.82 | 615.98 | 590.46 | 3.24 | 6.99 | 4.14 | 0.00 |
| 20-7     | 580.70    | 602.66 | 633.07 | 616.61 | 580.70 | 3.64 | 8.27 | 5.82 | 0.00 |
| 20-8     | 676.94    | 725.49 | 763.33 | 732.36 | 676.94 | 6.69 | 11.32 | 7.57 | 0.00 |
| 20-10    | 872.25    | 967.67 | 1011.55 | 986.43 | 872.25 | 9.86 | 13.77 | 11.57 | 0.00 |
| 30-3     | 473.49    | 475.11 | 476.79 | 473.49 | 475.12 | 0.34 | 0.69 | 0.00 | 0.34 |
| 30-5     | 647.62    | 663.40 | 680.15 | 656.20 | 647.62 | 2.38 | 4.78 | 1.31 | 0.00 |
| 30-6     | 926.32    | 974.11 | 1012.33 | 973.55 | 926.32 | 4.91 | 8.50 | 4.85 | 0.00 |
| 30-7     | 805.00    | 864.09 | 903.69 | 867.12 | 805.00 | 6.84 | 10.92 | 7.16 | 0.00 |
| 30-8     | 1019.08   | 1122.54 | 1174.56 | 1123.51 | 1019.08 | 9.22 | 13.24 | 9.29 | 0.00 |
| 30-10    | 1212.46   | 1393.90 | 1443.16 | 1402.20 | 1212.46 | 13.02 | 15.99 | 13.53 | 0.00 |
| 50-3     | 737.42    | 746.44 | 747.39 | 737.42 | 737.57 | 1.21 | 1.33 | 0.00 | 0.02 |
| 50-5     | 1076.32   | 1113.18 | 1137.75 | 1097.07 | 1076.32 | 3.31 | 5.40 | 1.89 | 0.00 |
| 50-6     | 1382.47   | 1509.56 | 1541.53 | 1477.59 | 1382.47 | 8.42 | 10.32 | 6.44 | 0.00 |
| 50-7     | 1334.23   | 1483.64 | 1521.70 | 1457.69 | 1334.23 | 10.07 | 12.32 | 8.47 | 0.00 |
| 50-8     | 1726.51   | 1928.91 | 1968.78 | 1910.14 | 1726.51 | 10.49 | 12.31 | 9.61 | 0.00 |
| 50-10    | 2071.06   | 2328.77 | 2368.72 | 2306.78 | 2071.06 | 11.07 | 12.57 | 10.22 | 0.00 |
| 100-3    | 1382.03   | 1415.09 | 1412.37 | 1382.03 | 1384.33 | 2.34 | 2.15 | 0.00 | 0.17 |
| 100-5    | 2299.06   | 2461.99 | 2487.24 | 2395.19 | 2299.06 | 6.62 | 7.57 | 4.01 | 0.00 |
| 100-6    | 3064.86   | 3374.44 | 3405.92 | 3302.44 | 3064.86 | 9.17 | 10.01 | 7.19 | 0.00 |
| 100-7    | 2689.59   | 2950.13 | 2971.90 | 2901.17 | 2689.59 | 8.83 | 9.50 | 7.29 | 0.00 |
| 100-8    | 3664.66   | 3960.40 | 3995.94 | 3905.27 | 3664.66 | 7.47 | 8.29 | 6.16 | 0.00 |
| 100-10   | 4276.66   | 4560.34 | 4588.66 | 4509.30 | 4276.66 | 6.22 | 6.80 | 5.16 | 0.00 |
| Average  | 1307.05   | 1405.22 | 1430.05 | 1391.13 | 1307.28 | 5.16 | 7.41 | 4.77 | 0.06 |

The bold emphasis means the better values.

Figure 7. Means and 95% LSD interval for the compared algorithms under different coefficients.
Figure 8. Convergence curve for instances under $\alpha = 0.8$: (a) convergence curve for instance 10-5, (b) convergence curve for instance 10-8, (c) convergence curve for instance 20-6, (d) convergence curve for instance 30-5, (e) convergence curve for instance 30-10, (f) convergence curve for instance 50-7, (g) convergence curve for instance 50-10, and (h) convergence curve for instance 100-6.
Conclusion and future research

Scheduling has always been one of the hot topics in manufacturing and production fields, and FJSP, as an extension of JSP, has received widespread attention because of its flexibility and realistic. In this study, we considered transportation time between machines, which are typically ignored in most literatures. We first modeled the FJSP with the transportation time as an extension of the canonical FJSP, denoting it as FJSP-T. Two objectives were considered simultaneously in the mathematical model, that is, the maximal completion time and the total energy consumption, and the weight decomposition is used to optimized the objectives. To solve this problem, a hybrid meta-heuristic algorithm based on the AIA and SA algorithm are proposed. In this algorithm, each solution is represented by two components, that is machine selection vector and operation sequence vector. Then, three initial strategies are applied to initial populations. To maintain the diversity of the population during iterations, an information entropy theory is embedded into the AIA for better global optimization. Meanwhile, the SA algorithm is proposed for further local search. Finally, a set of optimal parameters is obtained by using Taguchi method. To verify the validity of the FJSP-T model, the FJSP-T model is constructed by CPLEX, it is verified that the results obtained by CPLEX are not as good as that of proposed algorithm. In addition, we also carried out weight experiments under different coefficients, and analyzed the effectivenss of the algorithm under different coefficients. The multifactor ANOVA results and convergence curve comparisons also verified the competitiveness of the proposed algorithm. According to comparison with other meta-heuristic algorithms, the efficiency of the proposed algorithm is further proved.

However, there is still work to be considered in the future, we could (1) consider combining the flexible job shop scheduling problem with other complicated constraints that are typically ignored in this field to deploy our research to actual production, for example, fuzzy processing time, setup time, and distributed multifactories; (2) consider factors that we have not considered thoroughly enough, for example, the energy consumption generated by cranes in the process of handling and transferring, the energy consumption generated by machines in idle time; (3) consider the normalization of multiple objectives; (4) consider combining algorithms with neural model; and (5) consider applying the proposed algorithms to other scheduling fields.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research is partially supported by National Science Foundation of China under Grant 61773192, 61803192, 61773246, Shandong Province Higher Educational Science and Technology Program (J17KZ005).

ORCID iD

Jun-qing Li https://orcid.org/0000-0002-3617-6708

References

1. Nouiri M, Bekrar A, Jemai A, et al. An effective and distributed particle swarm optimization algorithm for
1. Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. *Ann Oper Res* 1993; 41(3): 157–183.
2. Garey MR, Johnson DS and Sethi R. The complexity of flowshop and job shop scheduling. *Math Oper Res* 1976; 1(2): 117–129.
3. Pezzella F, Morganti G and Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. *Comput Oper Res* 2008; 35(10): 3202–3212.
4. Girish BS and Jawahar N. A particle swarm optimization algorithm for flexible job shop scheduling problem. In: 2009 IEEE international conference on automation science and engineering, Bangalore, India, 22–25 August 2009, pp.298–303. IEEE.
5. Li JQ, Pan QK and Liang YC. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Comput Ind Eng* 2010; 59(4): 647–662.
6. Gao KZ, Suganthan PN, Pan QK, et al. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowl-Based Syst* 2016; 109: 1–16.
7. Wang L, Zhou G, Xu Y, et al. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *Int J Adv Manuf Technol* 2012; 60(1): 303–315.
8. Li X and Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int J Prod Econ* 2016; 174: 93–110.
9. Liu Z, Guo S and Wang L. Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption. *J Cleaner Prod* 2019; 211: 765–786.
10. Proorxfard H, Wong KY and Wong WP. Minimizing total carbon footprint and total late work in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resour Conserv Recyc* 2018; 128: 267–283.
11. Li JQ, Pan QK and Tasgetiren MF. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Appl Math Model* 2014; 38(3): 1111–1132.
12. Soto C, Dorronsoro B, Fraire HH, et al. Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. *Swarm Evol Comput* 2020.
13. Wu X, Shen X and Li C. The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously. *Comput Ind Eng* 2019; 1004–1024.
14. Ebrahimi A, Jeon HW, Lee S, et al. Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: a comparison of four metaheuristic algorithms. *Comput Ind Eng* 2020; 141: 106295.
15. Karimi S, Ardalan Z, Naderi B, et al. Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Appl Math Model* 2017; 41: 667–682.
16. Yu H, Li J, Zhang L, et al. An imperialist competition algorithm using a global search strategy for physical examination scheduling. *Appl Intell* 2020. DOI: 10.1007/s10489-020-01975-y.
35. Gao KZ, Suganthan PN, Pan QK, et al. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. J Intell Manuf 2016; 27(2): 363–374.
36. Meng L, Zhang C, Ren Y, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Comput Ind Eng 2020; 142: 106347.
37. Zitzler E and Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 1999; 3(4): 257–271.
38. Li JQ, Pan QK, Duan PY, et al. Solving multi-area environmental/economic dispatch by a Pareto-based chemical-reaction optimization algorithm. IEEE/CAA J Autom Sinica 2019; 6(5): 1240–1250.
39. Lu C, Gao L, Pan Q, et al. A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution. Appl Soft Comput 2019; 728–749.
40. Montgomery DC. Design and analysis of experiments, John Wiley & Sons, Arizona. Comput Ind Eng 2005; 56(4): 1494–1501.
41. Cui X, Li M and Fang T. Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principles. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546). IEEE, Seoul, South Korea, 2001, Vol. 2, pp. 1316–1321.
42. Li X and Gao L. Effective methods for integrated process planning and scheduling. Berlin: Springer, 2018.
43. Li X, Gao L, Pan Q, et al. An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. IEEE Trans Syst Man Cybern Syst 2018; 49(10): 1933–1945.
44. Dai M, Zhang Z, Giret A, et al. An enhanced estimation of distribution algorithm for energy-efficient job-shop scheduling problems with transportation constraints. Sustainability 2019; 11(11): 3085.
45. Amiri M, Zandieh M, Yazdani M, et al. A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. Int J Prod Res 2010; 48(19): 5671–5689.
46. Li JQ, Liu ZM, Li CD, et al. Improved artificial immune system algorithm for Type-2 fuzzy flexible job shop scheduling problem. IEEE Trans Fuzzy Syst 2020. DOI: 10.1109/TFUZZ.2020.3016225.
47. Zhang GH, Hu YF, Sun JH, et al. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. Swarm Evol Comput 2020. DOI: 10.1016/j.swevo.2020.100664.
48. Tao XR, Li JQ, Huang TH, et al. Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption. Complex Intell Syst 2020. DOI: 10.1007/s40747-020-00193-w.
49. Peng K, Pan QK, Gao L, et al. A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. Swarm Evol Comput 2019; 45: 92–112.
50. Wen L, Li X and Gao L. A transfer convolutional neural network for fault diagnosis based on ResNet-30. Neural Comput Appl 2020; 32: 6111–6124.
51. Wen L, Ye X and Gao L. A new automatic machine learning based hyperparameter optimization for workpiece quality prediction. Meas Control 2020. DOI: 10.1177/0020294020932347.
52. Li JQ, Han YQ, Duan PY, et al. Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems. J Cleaner Prod 2019. DOI: 10.1016/j.jclepro.2019.119464.
53. Li JQ and Han Y. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in Cloud computing system. Cluster Comput 2020. DOI: 10.1007/s10586-019-03022-z.
54. Song MX, Li JQ, Han YQ, et al. Metaheuristics for solving the vehicle routing problem with the time windows and energy consumption in cold chain logistics. Appl Soft Comput 2020; 95: 106561. DOI: 10.1016/j.asoc.2020.106561.