Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm

Sajad Karimi, Zaniar Ardalan, B. Naderi*, M. Mohammadi

Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran

ARTICLE INFO

Article history:
Received 28 December 2013
Revised 22 August 2016
Accepted 15 September 2016
Available online 30 September 2016

Keywords:
Scheduling
Flexible job shops
Transportation times
Mathematical models
Hybrid imperialist competitive algorithm

ABSTRACT

After the completion of a job on a machine, it needs to be transported to the next machine, actually taking some time. However, the transportation times are commonly neglected in the literature. This paper incorporates the transportation times between the machines into the flexible job-shop scheduling problem. We mathematically formulate the problem by two mixed integer linear programming models. Since the problem is NP-hard, we propose an adaptation of the imperialist competitive algorithm hybridized by a simulated annealing-based local search to solve the problem. Various operators and parameters of the algorithm are calibrated using the Taguchi method. The presented algorithm is assessed by comparing it against two other competitive algorithms in the literature. The computational results show that this algorithm has an outstanding performance in solving the problem.

1. Introduction

Scheduling is an integral part of the operation research which helps the industries to plan their manufacturing activities appropriately. Three factors of the workshop circumstances, the technical constraints of the processes, and the performance indexes compose a scheduling problem [1]. One famous problem in this area is the job-shop scheduling problem (JSSP). In the conventional JSSP, m different machines should process n different jobs. The aim of JSSP is to find a job sequence on each machine regarding a given objective function. The most common objective function is to minimize the makespan (the maximum completion time of all jobs). The JSSP is an NP-hard problem [2].

As a result of the progress in technology, factories utilize flexible machines that are able to perform more than one type of operation. In this regard, a variant of the JSSP, named flexible job-shop scheduling problem (FJSSP), is defined where machines are flexible. Hence, each operation can be carried out by different machines. In the FJSSP, there are two sub-problems: assigning operations to the machines and sequencing the operations on them. Note that the FJSSP is also NP-hard.

In the classical FJSSP, several basic assumptions, partially non-realistic, are considered. One such assumption is that each job can be processed on the next machine forthwith after its completion on the previous machine. That is, the transportation times are overlooked. Yet in real cases, jobs should be first transported between the machines by transportation systems. This drawback motivates us to incorporate the transportation times between the machines into the FJSSP. There are two types of transportation systems: single-transporter and multi-transporter systems. Here we consider the multi-transporter system. In this system, several devices are deployed to perform the transportation of the jobs from one machine to another.

* Corresponding author.
E-mail addresses: bahman.naderi@aut.ac.ir, bahman_naderi62@yahoo.com (B. Naderi).

http://dx.doi.org/10.1016/j.apm.2016.09.022
0307-904X/© 2016 Elsevier Inc. All rights reserved.

Since the number of the transporters is considered to be infinite, jobs can be transported without any delay. We also consider job-dependent transportation times. That is, the magnitude of the transportation time depends on both the distance among the machines and the job to be transported.

The major contribution of this paper is to consider the transportation times in the FJSSP. It provides two mathematical models: sequence-based and position-based mixed integer linear programming (MILP) models. Furthermore, since the FJSSP is NP-hard, we develop a new imperialist competitive algorithm to solve this problem. This method utilizes the assimilation policy, the imperialist competition mechanisms, and a simulated annealing-like local search. The algorithm is evaluated by comparing it with two mathematical models and two available algorithms in the relevant literature: the genetic algorithm [14] and the chemical reaction optimization [18].

The remainder of this paper is organized as follows. The literature review is discussed in Section 2. The MILP models are presented in Section 3. The developed metaheuristic algorithm is described in Section 4. The computational results are presented in Section 5. The conclusion and future research are mentioned in Section 6.

2. Literature review

The paper by Bruker and Schlie [3] is the first work to address the FJSSP. They developed a polynomial algorithm to solve the FJSSP. Afterwards, to solve this problem, various approaches have been deployed in the literature. These approaches can be divided into two groups: integrated and hierarchical methods [4]. In the first approach, the assignment and sequencing decisions are determined interactively. In the second approach, these two decisions are taken independently.

Choi and Choi [5] proposed a mixed integer linear programming (MILP) model to solve the FJSSP. They considered alternative operations and sequence-dependent setup times. Gao et al. [6] studied the FJSSP with machine availability constraints and presented a mixed integer non-linear programming (MINLP) formulation. Their solution procedure was a hybridized genetic algorithm. Imanipour and Zegordi [7] studied the FJSSP. They also presented an MINLP model for the FJSSP with sequence-dependent setup times and developed a Tabu search. Fattahi et al. [8] proposed the first position-based MILP model and also developed six heuristics for the FJSSP. To obtain the better solutions, a mathematical formulation is deployed in small-sized instances. The heuristics contain both integrated and hierarchical approaches. In another research, Fattahi et al. [9] considered the FJSSP with overlap between the operations. A hierarchical approach was developed based on simulated annealing. Ozguven et al. [10] dealt with two problems: the FJSSP with routing and sequencing sub-problems, and the FJSSP with process plan flexibility. Ozguven et al. [11] studied an advanced form of the FJSSP with sequence-dependent setup times as well as routing flexibility. They proposed two MILP formulations for the problem. Roshanaei et al. [41] developed new solution methodologies for the FJSSP. They proposed two MILP models to mathematically formulate the problem: position-based and sequence-based.

Several other meta-heuristics have been proposed as follows: a hierarchical Tabu search by Brandimarte [12], genetic algorithms by Chen et al. [13] and Zhang et al. [14], a local search technique by Mastrolilli and Gambardella [15], parallel variable neighborhood search by Yazdani et al. [16], a hybrid particle swarm optimization by Xia and Wu [17], a discrete chemical-reaction optimization by Li and Pan [18], an artificial immune algorithm by Bagheri et al. [32], and an evolutionary algorithm by Rahmati et al. [34].

The literature of scheduling with transportation times is as follows. Strusevich [19] considered the time lags in the open-shop problems. He assumed those time lags as the times required to transport a job to the next machines. Hurink and Knust [20] studied this problem with a single-transporter. Langston [21] considered two-stage hybrid flow-shops, and introduced several solving procedures. Naderi et al. [22] incorporated the sequence-dependent transportation times with a single-transporter system into the flow-shop. Their solution procedure was an adaptation of the simulated annealing algorithm. Boudhar and Haned [23] studied scheduling preemptive jobs on the identical parallel machines. Several heuristic algorithms were presented and a high-quality lower-bound one was obtained for the makespan-minimizing problems. Naderi et al. [44] considered the transportation times in the permutation flow-shops. They proposed six different models for the problem.

3. Problem description and mathematical modeling

This section first illustrates the problem by a numerical instance of the FJSSP. The processing and transportation times are respectively shown in Tables 1 and 2.

The optimal solutions of this problem with and without the transportation times are shown as Gantt charts in Figs. 1 and 2. As shown, the optimal solutions of the two problems are different.

Later on, the FJSSP with the transportation times is mathematically formulated. Note that the decision variables in the model of the FJSSP should determine both the assignment of the operation to the machines and the sequence of the operations on each machine. To solve this problem, two types of MILP models are presented: sequence-based and position-based. The proposed models are based on the model presented by Roshanaei et al. [41]. In the sequence-based model the binary variables are explained as follows. For each pair of the operations (j, h), there is a binary variable which shows whether operation j is processed after operation h or not. In the position-based model, binary variables determine the positions of the operations in the sequence.

Table 1
The processing times of each operation on each machine.

Jobs	Operations	Machines		
		M1	M2	M3
Job 1	$O_{1,1}$	4	–	2
	$O_{1,2}$	4	2	–
	$O_{1,3}$	5	7	2
Job 2	$O_{2,1}$	-	3	5
	$O_{2,2}$	3	2	4

Table 2
The transportation times between the machines for each job.

jobs	Machines	Machines		
		M1	M2	M3
Job 1	M0	2	3	2
	M1	3	2	1
	M2	4	1	3
	M3	1	8	5
Job 2	M0	5	1	3
	M1	1	3	2
	M2	5	5	1
	M3	3	4	4

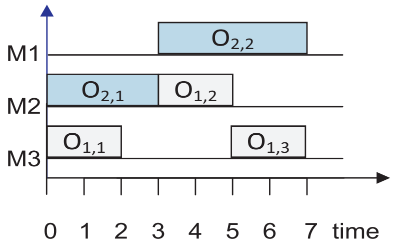

Fig. 1. The Gantt chart of the FJSSP without the transportation times.

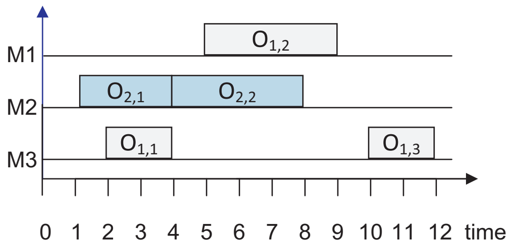

Fig. 2. The Gantt chart of the FJSSP with the transportation times.

3.1. Assumptions and parameters

In both models, the transportation times between the machines is considered as follows. There is an unlimited number of transporters. Thus, no delay occurs while transporting jobs to the next machines. Machine failure is not considered. Buffers are infinite. Machines cannot process more than one job simultaneously. Each operation should be allocated to one machine. Every machine and every job is ready at time 0. Each job has a fixed processing sequence; if the process of an operation is started, it should be finished without any interruption. In order to consider the transportation times between the input warehouse and the first machine, a dummy machine with zero processing time, say machine 0, is defined. The parameters and the indexes are presented in Table 3.

3.2. Model 1 (sequence-based model)

In this subsection, the sequence-based mathematical model for the FJSSP is presented. It is noteworthy that the index range of the machine is from 0 to m. The zero in the range denotes a dummy machine. The model variables are:

Table 3
The parameters and the indexes.

Para.	Definition
N	The number of the jobs
m	The number of the machines
j,h	The indices for the jobs where $\{1,2,\ldots,n\}$
i	The indices for the machines where $\{1,2,\ldots,m\}$
n_j	The number of the operations of job j.
$O_{\ j,l}$	l_{th} operation of job j
$P_{j,l,i}$	The processing time of $O_{j,l}$ on machine i.
l,z	The indices for the operations of job j $\{1,2,\ldots,n_j\}$
$t_{j,k,i}$	The transportation time of job j from machine k to machine i.
f,k	The indices for the positions in machine i where $\{1,2,\ldots,r_i\}$
$e_{j,l,I}$	The parameter that takes value 1 if $O_{j,l}$ is eligible to be processed on machine i; otherwise 0.
r_i	The number of the operations eligible to be processed on machine i (i.e., $r_i = \sum_{j=1}^{n}\sum_{l=1}^{n_j} e_{j.l.i}$)
M	A large positive number

$X_{j,l,h,z}$: A binary variable taking value 1 if $O_{j,l}$ is processed after $O_{h,z}$ and 0 otherwise. $J=\{1,2,\ldots,n-1\},h>j$
$Y_{j,l,i,k}$: A binary variable taking value 1 if $O_{j,l}$ is processed on machine i and $O_{j,l-1}$ on machine k.
$C_{j,l}$: A continuous variable for the completion time of $O_{j,l}$

Minimize C_{max}
Subject to:

$$\sum_{i=1}^{m}\sum_{k=1}^{m} Y_{j,l,i,k} = 1 \quad \forall_{j,l>1} \tag{1}$$

$$\sum_{i=1}^{m} Y_{j,1,i,0} = 1 \qquad \forall_{j} \tag{2}$$

$$\sum_{k=1}^{m} Y_{j,l,i,k} \le e_{j,l,i} \qquad \forall_{j,l>1,i} \tag{3}$$

$$Y_{j,1,i,0} \le e_{j,1,i} \qquad \forall_{j,i} \tag{4}$$

$$Y_{j,l,i,k} \le \sum_{f=1}^{m} Y_{j,l-1,k,f} \qquad \forall_{j,l>2,i,k} \tag{5}$$

$$Y_{j,2,i,k} \le Y_{j,1,k,0} \qquad \forall_{j,i,k} \tag{6}$$

$$C_{j,l} \ge C_{j,l-1} + \sum_{i=1}^{m}\sum_{k=1}^{m} Y_{j,l,i,k}\big(p_{j,l,i} + \ t_{j,k,i}\big) \qquad \forall_{j,l>1} \tag{7}$$

$$C_{j,1} \ge \sum_{i=1}^{m} Y_{j,1,i,0}\cdot\big(p_{j,1,i} + \ t_{j,0,i}\big) \quad \forall_{j} \tag{8}$$

$$C_{j,l} \ge C_{h,z} + p_{j,l,i} - M\big(1 - X_{j,l,h,z}\big) - M\left(2 - \left(\sum_{k=1}^{m} Y_{j,l,i,k}\right) - \left(\sum_{k=1}^{m} Y_{h,z,i,k}\right)\right) \quad \forall_{j\langle n,l\rangle 1;h>j,z>1;i} \tag{9}$$

$$C_{j,1} \ge C_{h,z} + p_{j,1,i} - M\big(1 - X_{j,1,h,z}\big) - M\left(2 - Y_{j,1,i,0} - \left(\sum_{k=1}^{m} Y_{h,z,i,k}\right)\right) \qquad \forall_{j\langle n;h\rangle j,z>1;i} \tag{10}$$

$$C_{j,l} \ge C_{h,1} + p_{j,l,i} - M\big(1 - X_{j,l,h,1}\big) - M\left(2 - \left(\sum_{k=1}^{m} Y_{j,l,i,k}\right) - Y_{h,1,i,0}\right) \qquad \forall_{j\langle n,l\rangle 1;h>j,i} \tag{11}$$

$$C_{j,1} \ge C_{h,1} + p_{j,1,i} - M\big(1 - X_{j,1,h,1}\big) - M\big(2 - Y_{j,1,i,0} - Y_{h,1,i,0}\big) \qquad \forall_{j\langle n,h\rangle j,i} \tag{12}$$

$$C_{h,z} \ge C_{j,l} + p_{h,z,i} - M\big(X_{j,l,h,z}\big) - M\left(2 - \left(\sum_{k=1}^{m} Y_{j,l,i,k}\right) - \left(\sum_{k=1}^{m} Y_{h,z,i,k}\right)\right) \qquad \forall_{j\langle n,l\rangle 1;h>j,z>1;i} \tag{13}$$

$$C_{h,z} \ge C_{j,1} + p_{h,z,i} - M\big(X_{j,1,h,z}\big) - \ M\left(- \ M(2 - Y_{j,1,i,0} - \left(\sum_{k=1}^{m} Y_{h,z,i,k}\right)\right) \qquad \forall_{j\langle n;h\rangle j,z>1;i} \tag{14}$$

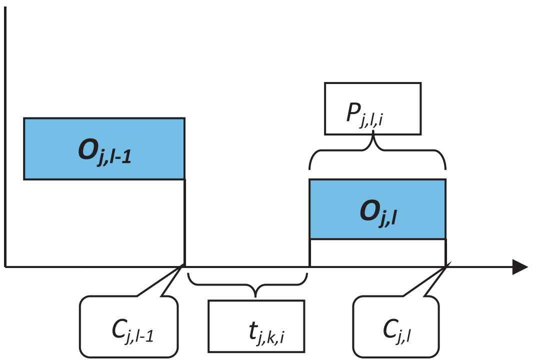

Fig. 3. The schematic Gantt chart to illustrate the constraints.

$$C_{h,1} \geq C_{j,l} + p_{h,1,i} - M\left(X_{j,l,h,1}\right) - M\left(2 - \left(\sum_{k=1}^{m} Y_{j,l,i,k}\right) - Y_{h,1,i,0}\right) \qquad \forall_{j\langle n,l\rangle 1;h>j,i} \tag{15}$$

$$C_{h,1} \geq C_{j,1} + p_{h,1,i} - M\left(X_{j,1,h,1}\right) - M\left(2 - Y_{j,1,i,0} - Y_{h,1,i,0}\right) \qquad \forall_{j\langle n,h\rangle j,i}, \tag{16}$$

$$C_{max} \geq C_{j,n_j} \qquad \forall_j \tag{17}$$

$$C_{j,l} \geq 0 \qquad \forall_{j,l} \tag{18}$$

$$X_{j,l,h,z}, Y_{j,l,i,k} \in \{0,\ 1\}. \tag{19}$$

The objective function is minimizing the maximum makespan. Constraint set (1) guarantees that every operation (except the first operation) should be allocated to only one machine. Constraint set (2) represents that the first operation of every job is exactly assigned to one position of one available machine.

For every operation, there is a set of machines that are feasible to process this operation, and this set of feasible machines (alternative machines) does not necessarily include all the machines for each operation. So, we define constraint set (3) in order to make sure that the machine for each $O_{j,l}$ (except $O_{j,1}$) is selected from the eligible alternative machines for $O_{j,l}$. Constraint set (4) enforces that the first operation of each job should be operated on the feasible machines for this operation.

According to the definition of variable $Y_{j,l,i,k}$, $O_{j,l-1}$ should be processed on machine k, if $O_{j,l}$ is processed on machine i. Therefore, constraint sets (5) and (6) ensure that if $O_{j,l}$ is operated on machine i, $O_{j,l-1}$ is processed on machine k.

It is noteworthy that constraint sets (7)–(16) are sequencing constraints. As shown in Fig. 3, the operation $O_{j,l}$ should be completed after the completion of $O_{j,i-1}$, the transportation of the related job, and the processing of the $O_{j,l}$. Hence, constraint sets (7) and (8) enforce that $O_{j,l}$ starts just after the completion of job $O_{j,l-1}$ and the transportation of job j to machine i. These constraints consider the transportation times in the FJSSP model. Constraint sets (9) to (16) force every machine to operate only one operation at a certain time. Constraint set (17) determines the value of the makespan by considering the completion time of the last operation of all the jobs. Constraint set (18) enforces that the continuous variables take positive values, and constraint set (19) shows that those related variables are binary.

3.3. Model 2 (the position-based model)

The position-based mathematical model is presented here. The binary variables locate the jobs in the sequence. Like the sequence-based model, the machine index begins from 0. The variables are as follows.

$X_{j,l,i,f,k}$ Take value 1 if $O_{j,l}$ is processed in position f of machine i after the processing of job j on machine k, and 0 otherwise.
$C_{j,l}$ The completion time of $O_{j,l}$.

Minimize C_{max}
Subject to:

$$\sum_{i=1}^{m} \sum_{f=1}^{r_i} \sum_{k=1}^{m} X_{j,l,i,f,k} = 1 \qquad \forall_{j,l>1}, \tag{20}$$

$$\sum_{i=1}^{m} \sum_{f=1}^{r_i} X_{j,1,i,f,0} = 1 \qquad \forall_j, \tag{21}$$

$$\sum_{j=1}^{n}\sum_{l=2}^{n_j}\sum_{k=1}^{m} X_{j,l,i,f,k} + \sum_{j=1}^{n} X_{j,1,i,f,0} \le 1 \qquad \forall_{i,f \in R_i}, \tag{22}$$

$$\sum_{f=1}^{r_i}\sum_{k=1}^{m} X_{j,l,i,f,k} \le e_{j,l,i} \qquad \forall_{j,l>1,i}, \tag{23}$$

$$\sum_{f=1}^{r_i} X_{j,1,i,f,0} \le e_{j,1,i} \qquad \forall_{j,i}, \tag{24}$$

$$\sum_{f=1}^{r_i} X_{j,l,i,f,k} \le \sum_{f=1}^{r_k}\sum_{t=1}^{m} X_{j,l-1,k,f,t} \qquad \forall_{j,l>2,i,k}, \tag{25}$$

$$\sum_{f=1}^{r_i} X_{j,2,i,f,k} \le \sum_{f=1}^{r_k} X_{j,1,k,f,0} \qquad \forall_{j,i,k}, \tag{26}$$

$$C_{j,l} \ge C_{j,l-1} + \sum_{i=1}^{m}\sum_{f=1}^{r_i}\sum_{k=1}^{m} X_{j,l,i,f,k}\left(p_{j,l,i} + t_{j,k,i}\right)\forall_{j,l>1} \tag{27}$$

$$C_{j,1} \ge \sum_{i=1}^{m}\sum_{f=1}^{r_i} X_{j,1,i,f,0}\left(p_{j,1,i} + t_{j,0,i}\right) \quad \forall_{j}, \tag{28}$$

$$C_{j,l} \ge C_{h,z} + p_{j,l,i} - M\left(1 - \sum_{k=1}^{m} X_{j,l,i,f,k}\right) - M\left(1 - \sum_{t=1}^{f-1}\sum_{k=1}^{m} X_{h,z,i,t,k}\right) \quad \forall_{j,l>1,h,z>1,j\ne h,i,f\epsilon R_i>1}, \tag{29}$$

$$C_{j,1} \ge C_{h,z} + p_{j,1,i} - M\left(1 - X_{j,1,i,f,0}\right) - M\left(1 - \sum_{t=1}^{f-1}\sum_{k=1}^{m} X_{h,z,i,t,k}\right) \qquad \forall_{j,h,z>1,j\ne h,i,f\epsilon R_i>1}, \tag{30}$$

$$C_{j,l} \ge C_{h,1} + p_{j,l,i} - M\left(1 - \sum_{k=1}^{m} X_{j,l,i,f,k}\right) - M\left(1 - \sum_{t=1}^{f-1} X_{h,1,i,t,0}\right) \quad \forall_{j,l>1,h,j\ne h,i,f\epsilon R_i>1}, \tag{31}$$

$$C_{j,1} \ge C_{h,1} + p_{j,1,i} - M\left(1 - X_{j,1,i,f,0}\right) - M\left(1 - \sum_{t=1}^{f-1} X_{h,1,i,t,0}\right) \quad \forall_{j,h,j\ne h,i,f\epsilon R_i>1}, \tag{32}$$

$$C_{max} \ge C_{j,n_j} \qquad \forall_j \tag{33}$$

$$C_{j,l} \ge 0 \tag{34}$$

$$X_{j,l,i,f,k} \in \{0,\ 1\} \tag{35}$$

The objective function is to minimize the makespan. Constraint set (20) ensures that each operation (except the first operation) uniquely occupies one position on one machine among all of the available machines. Constraint set (21) ensures that the first operation of each job is assigned to one position of one available machine. Constraint set (22) guarantees that each position of each machine is occupied once. When l is equal to 1, index k will be equal to 0 because $O_{j,1}$ is the first operation in each job. Constraint set (23) enforces that $O_{j,l}$ (except $O_{j,1}$) must be assigned to an eligible machine that can process $O_{j,l}$. Constraint set (24) enforces that the first operation of each job should be operated on its respective and eligible machines. According to the definition of variable $X_{j,l,i,f,k}$, if the operation l of one job is operated on machine i, O_{l-1} should be operated on machine k. Thus, we define Constraint sets (25) and (26) to ensure that $O_{j,l-1}$ is processed on machine k when $O_{j,l}$ is processed on machine i. Constraint sets (27) to (32) are sequencing constraints. Constraint sets (27) and (28) are logical precedence constraints among the operations of a job and the transportation times between the machines considered in these constraints. According to Fig. 3, the processing of $O_{j,l}$ can be completed after the completion of its previous operation ($O_{j,l-1}$), and the transportation of the job from machine k to machine i, and the processing of $O_{j,1}$ on machine i. Constraint sets (27) and (28) guarantee this issue. Constraint sets (29)–(32) ensure that every machine processes only one operation at a time. Constraint set (33) defines the makespan. Constraint sets (34) and (35) define the decision variables.

```
Procedure: the proposed imperialist competitive algorithm

Initialize population (by GS, LS and RS)
Do empires formation
While the stopping criterion is not met do
        For i=1 to Nimp do
              Mutate the imperialist
              Assimilate colonies by mutated imperialist
        Endfor
        Update the imperialists
        Do imperialist competition strategy
        For i=1 to Nimp do
           Do imperialist development plans mechanism (IDP)
        Endfor
        Replace similar colonies with a new solution producing by initialization procedure
        Apply Local Search on best imperialist
Endwhile
```

Fig. 4. The Pseudo-code of the proposed imperialist competitive algorithm.

Country=

machine-selection (MS)					operation-sequence(OS)				
2	1	3	1	2	1	2	2	1	1

Fig. 5. The structure of the solution representation in the imperialist competitive algorithm.

4. The proposed imperialist competitive algorithm

Since the FJSSP is NP-hard, solving the large-scale problems with the proposed mathematical models is practically impossible. Therefore, we develop an imperialist competitive algorithm (ICA). This algorithm has been proposed for various optimization problems [24,25]. It is a population-based algorithm in which every element of the population are called a country. Each country denotes an encoded solution to the problem. The better countries are chosen to become the imperialists. The other countries are divided among the imperialists as colonies. The whole set of an imperialist and its colonies is called an empire. After seizing the colonies, the imperialists try to penetrate into their colonies by making them similar to themselves. Then the imperialists compete with each other to seize more colonies and gain more power. This process makes some imperialists more powerful and some weaker. The weak empires eventually collapse. The ICA has been employed in several areas such as the non-convex dynamic economic power-dispatching problem [35], the project scheduling problem [36], the hub location problem [37], the supply chain network design [38], the power flow problem with non-smooth cost functions, [39] and the complicated image-matching problems [40].

4.1. The proposed ICA procedure

This paper proposes a well-organized ICA for the FJSSP. In this regard, some effective changes are made in the conventional ICA as follows. As mentioned above, the imperialist countries assimilate their colonies by making them similar to themselves. This procedure is called the assimilation policy. This procedure lessens the diversity of the algorithm. Thus, in order to prevent this similarity and to have more powerful colonies, we mutate empires by a mutation strategy and then the colonies are assimilated into these mutated empires. In the real world cases, imperialist countries have several strategies to develop their power and domination, but the classical ICA does not develop the imperialists' power, obviously it is a drawback of the classical ICA. Hence, in order to make up for that, a strategy called the imperialists' development plan is applied in the algorithm. This helps the algorithm to give better results. Furthermore, to enhance the algorithm diversification, a replacing strategy is implemented and also a local search is deployed to increase the intensification. The pseudo-code of the ICA is shown in Fig. 4.

4.2. Encoding scheme, initialization, and empires' formation

In the literature of the FJSSP, several solution representations exist [13,14,45]. Each one has its own pros and cons. In this paper, we use an encoding scheme proposed by Zhang et al. [14]. This solution representation does not require repair mechanism. Therefore, the decoding procedure takes less time throughout the iterations. This encoding scheme includes two parts; the first part is machine-selection (MS) and the second part is operation-sequence (OS). They are named MSOS (Fig. 5). According to the example mentioned in Section 2, one possible encoding can be as follows.

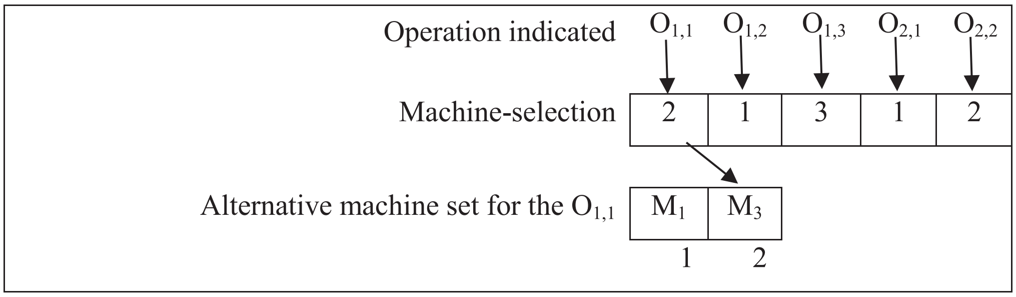

Fig. 6. Machine-selection, part of the solution representation.

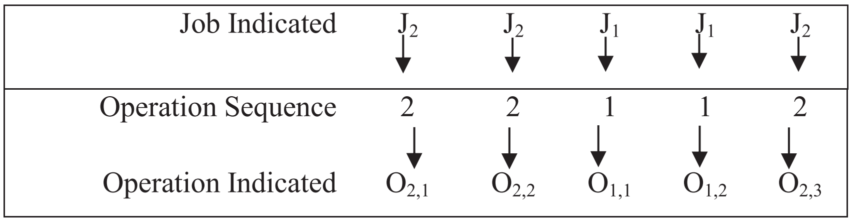

Fig. 7. Operation sequence, part of the solution representation.

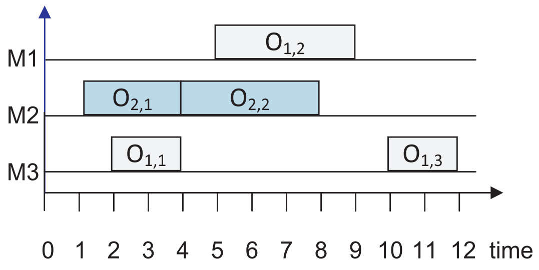

Fig. 8. The Gantt chart of the encoding scheme.

The value placed in the machine cell for the first operation is equal to 2. This means that 2th machine in the alternative machine set is selected for the first operation. (See Fig. 6).

In the second part, the indicator of each job is duplicated n_j times, that n_j is the number of the operations of job j. Also, the sequence of the operations is indicated by this part of encoding. It is illustrated in Fig. 7.

The sequence of the operations in Fig. 6 is $O_{2,1}$ – $O_{2,2}$ –$O_{1,1}$– $O_{1,2}$– $O_{2,3}$. The decoding of the encoding in Fig. 4 is shown as a Gantt chart in Fig. 8.

Generating the initial countries is conducted through two steps. In the first step, we generate the initial machine-selection part. For this part, three approaches, already presented by Zhang et al. [14], are adapted: global selection (GS), local selection (LS), and random selection (RS). The adaptation of the first two approaches is applied by adding the transportation time to the processing time of the corresponding machine. In the second part of the solution representation, the sequence of all the operations is randomly determined. The major advantage of the presented encoding scheme is that it does not produce infeasible solutions.

In the ICA, some countries become imperialist and others are the colonies of these imperialists. In order to select the imperialists, the powerful ones (countries with better objective functions) are selected. The rest form their colonies. The more powerful an imperialist is, the more colonies it seizes. The population size (POP) includes several imperialist countries (N_{imp}) and their colonies (N_{col}).

$$\text{Pop} = N_{imp} + N_{col}.$$

To assign colonies to the imperialists, the roulette wheel selection procedure is used. The power of each imperialist is calculated by:

$$Power\ (k) = \frac{1}{objective\ \ function\ \ (k)},$$

where *objective function (k)* is the makespan value corresponding to the kth imperialist. After the calculation of the power of the imperialists, we normalize them as follows.

$$p_k\ \frac{power\ \ (k)}{\sum_{k=1}^{Nimp} power\ \ (k)},$$

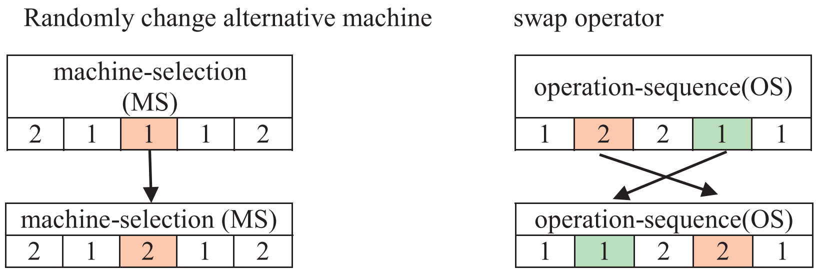

Fig. 9. The mutation procedure before the assimilation.

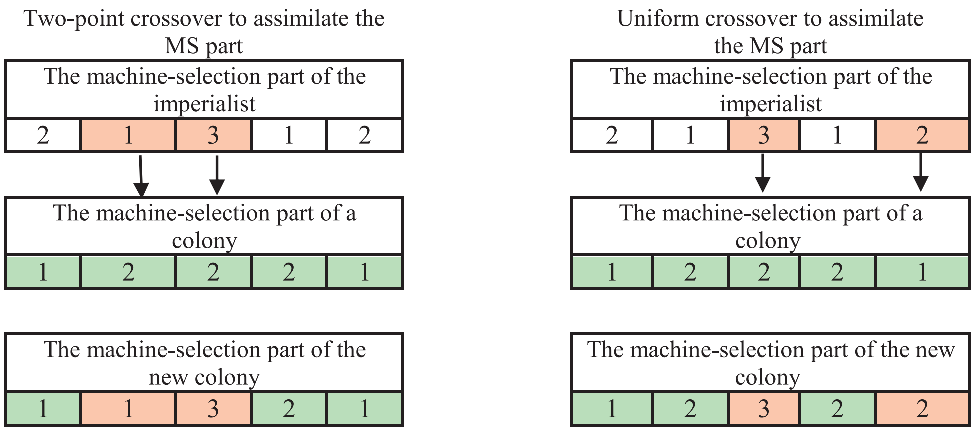
Two-point crossover to assimilate the MS part
The machine-selection part of the imperialist
2 1 3 1 2
The machine-selection part of a colony
1 2 2 2 1
The machine-selection part of the new colony
1 1 3 2 1
Uniform crossover to assimilate the MS part
The machine-selection part of the imperialist
2 1 3 1 2
The machine-selection part of a colony
1 2 2 2 1
The machine-selection part of the new colony
1 2 3 2 2

Fig. 10. The assimilation procedure for the machine-selection part.

where p_k is the normalized power of the kth imperialist and N_{imp} indicates the number of the imperialists. Now p_k is the probability of assigning each colony to the kth imperialist.

4.3. The assimilation policy mechanism

In the real world, imperialist countries desire to make their colonies similar to themselves. Each colony has some features such as language, religion, culture, economy, etc. These features are affected by the imperialist in order to assimilate the colonies. Thus, in the ICA, the assimilation policy mechanism denotes this imperialist effort.

In this paper, the encoding scheme has two independent parts. Hence, we employ different operators in each part. For the machine-selection part, two-point and uniform crossovers are deployed. In the operation-sequence part, OX and POX crossovers are utilized. It is noteworthy that if all the colonies of an imperialist assimilate into the imperialist, after some iteration, many repetitive colonies are obtained. To prevent this from happening, the imperialist is firstly mutated, and then the colonies are assimilated into the mutated imperialist.

For the first part of the encoding, mutation is done by changing the alternative machine of the related operation randomly. For the second part, a swap operator is used to make a mutation (see Fig. 9).

In the two-point crossover [26] for the first part, two points of the imperialist are randomly selected, and the cells between these two points are inserted into the corresponding cells of the colony. Then, a new colony is made. In the uniform crossover [27], each cell of the imperialist is selected with a constant probability value, and inserted into its related cell in the colony (see Fig. 10).

In the OX crossover [28], two points of the operation-sequence part of the imperialist are randomly selected, and the cells between those two points are inserted into the new colony. The other cells of the new colony are selected from the older colony with respect to the order in the older colony (Fig. 11). In the POX crossover [27], some jobs are randomly selected, and the related cells from the imperialist are inserted into the new colony. The remaining jobs are inserted into the new colony according to the order in the older colony (Fig. 12).

4.4. The imperialists' updating and the competition mechanism

Through the iterations, colonies may obtain greater power (better objective functions) than their imperialist. In this case, the country with the greatest power becomes the new imperialist. The imperialists compete with each other to obtain more colonies and extend their power. When an imperialist gradually loses its power, a more powerful one may capture its

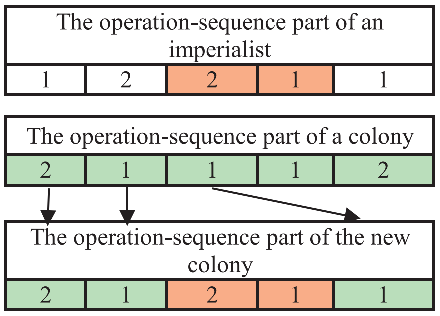
The operation-sequence part of an imperialist
1 2 2 1 1
The operation-sequence part of a colony
2 1 1 1 2
The operation-sequence part of the new colony
2 1 2 1 1

Fig. 11. The OX crossover to assimilate the operation-sequence part.

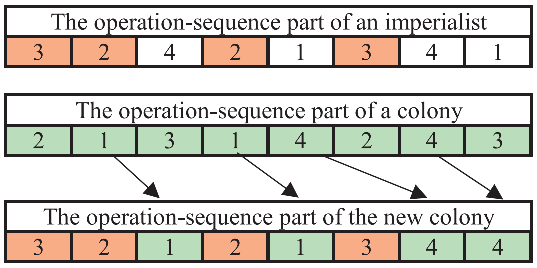
The operation-sequence part of an imperialist
3 2 4 2 1 3 4 1
The operation-sequence part of a colony
2 1 3 1 4 2 4 3
The operation-sequence part of the new colony
3 2 1 2 1 3 4 4

Fig. 12. The POX crossover to assimilate the operation-sequence part.

colonies. Therefore, in each iteration of the algorithm, the colony with the least power under the imperialist with the least power is separated from it and annexed to another imperialist with respect to the power of the imperialists.

4.5. The imperialists' development plan mechanism and local search

Ignoring the imperialists' behavior in the classical ICA causes the colonies to become assimilated into the less powerful imperialist. In order to eliminate this shortcoming, we use the imperialists' development plan mechanism (IDP). This mechanism improves the performance of the ICA.

In the IDP, for each imperialist, N_{idp} development plans (new solutions) are randomly produced. Then, the best plan (best solution) is selected. The development plans are designed based on the current situation. Thus, in order to design a plan based on the current situation, we should make a slight change to the imperialist. According to the fact that the encoding scheme has two parts, an operator is chosen to implement this change to each part. First, in the machine-selection part, one cell is randomly selected, and an alternative machine is randomly changed. Second, for the operation-sequence part, a shift operator is selected. In the shift operator, one cell is randomly selected and transported to another random position in the encoding scheme.

In evolutionary algorithms, similar solutions decrease the diversity of the algorithms. This deteriorates the algorithm performance. Here, similar colonies are replaced by new colonies produced by the initialization phase. Moreover, to improve the performance by enhancing the intensification of the algorithm, a local search based on the simulated annealing (SA), is utilized. The best imperialist is improved by this local search. The machine-selection part and the operation-sequence part of the solution are nested and improved in the local search (see Fig. 13). The first part is improved by the internal loop, and the external loop tries to find a better operation-sequence part. The procedure is inspired by SA. Some changes are made to the current solution. Then, if a better solution is obtained, it is accepted. Otherwise, this change is accepted with a low probability. To change the machine-selection part, one cell is selected, and its alternative machine is substituted. For the second part of the encoding, inversion operator [29] is used.

5. Numerical experiments

This section first evaluates and compares the performance of two presented MILP formulations based on the size and computational complexities. Then, the proposed metaheuristic algorithm is calibrated and compared with both models on small instances. Later, we evaluate the metaheuristic algorithm on large instances by comparing it with two best performing algorithms available on the FJSSP literature (genetic algorithm by Zhang et al. [14] and chemical reaction optimization by Li and Pan [18]). These two algorithms overcome other competitive algorithms in the literature [14,18].

The MILP models are solved by GAMS 22.2 and the algorithms are implemented by MATLAB 7.12. They are run on a system with a 2.1 GHz Intel core i3 processor and a 4 GB RAM memory. To compare the algorithms, we need a suitable

```
Procedure: local search
For i=1 to Nimp do
    For i=1 to ITERseq (# iteration of operation-sequencepart) do
        Change operation-sequencepart
        For j=1 to ITERassg (# iteration of machine-selection part) do
            Change machine-selection part
            If new objective < current objective then
                Accept new solution
            Else
                Accept new solution with a low probability
            Endif
        Endfor
        If new objective < current objective then
            Accept new solution
        Else
            Accept new solution with a low probability
        Endif
    Endfor
Endfor
```

Fig. 13. The Pseudo-code of the local search.

Table 4
The size complexity comparison between the models.

	Sequence-based model	Position-based model
# Constraint	$2.L^2(m+1)(n-1)!+m^2n(n-1)$ $+3mnL-2mn+4nL$	$L^2(n^2-n)(m+1)(F-1)+m^2n(L-1)$ $+mn(3L-2)+4Ln+mF+F$
# Binary variables	$(nL)^2+nL(m+1)^2$	$nL(m+1)^2F$
# Continues variables	nL	nL
# Total variables	$(nL)^2+nL(m+1)^2+nL$	$nL(m+1)2F+ nL$

performance index. In this regard, we use the relative percentage deviation (RPD) [28]. The RPD is calculated as follows.

$$RPD_{i,j} = \frac{X_{i,j} - X_{min,j}}{X_{min,j}} \times 100,$$

where $X_{i,\,j}$ is the average obtained from the objective functions from ith trail of jth instance.

5.1. Models' evaluation

The models are compared based on the size and computational complexities. In the size complexity, the comparison is conducted based on the number of the variables and the constraints. Table 4 shows the results of the comparison. Parameter L is the maximum number of the jobs' operations, and parameter F is the maximum number of the positions on the machines.

To evaluate the computational complexity of the MILP models, we produce a set of small instances. As there is no dataset containing the transportation times in the literature of the FJSSP, we generate instances based upon Barnes and Brandimarte datasets [42,43]. The number of the machines (m), the number of the jobs (n) and the average number of the operations in each job (o) are as follows. $m = \{2, 4\}$, $n = \{2, 3, 4, 5, 6\}$, o= {3, 5}. We generate 20 instances from the combinations of m, n and o. To generate the processing times, a uniform distribution between [1,99] is used and the transportation times between the machines are generated according to a uniform distribution between [1,30]. The instances can be emailed upon the request.

We set the stop criterion for the MILP models at 1000 seconds of the computational time. The results of the comparison are presented in Table 5. In this table, the GAP is the relative gap obtained from the GAMS. As reported in Table 5, the sequence-based model has a better performance (time and Gap) in comparison with the position-based model.

5.2. Parameter tuning

Before evaluating the proposed algorithm, we calibrate the algorithm, i.e. different parameters of the algorithm are analyzed using the Taguchi method. The stopping criterion of the tested algorithms is the elapsed computational time of $0.05\times m\times n\times o$ seconds. The Initial tests show that the suitable computational time to reach the desirable result depends on the number of the machines, jobs, and the average operations of each job.

The critical part of adapting an efficient evolutionary algorithm is the selection of the values of the parameters, which is called parameter tuning. We evaluate the impacts of the parameters on the performance of the algorithms. Several methods

Table 5
The results of the computational complexity comparison between the models.

Instance	size (n,o,m)	Sequence-based model					Position-based model				
		C_{max}	GAP	CPU time (s)	# variables	# constraints	C_{max}	GAP	CPU time (s)	# variables	# constraints
1	2,3,2	202	0	0.45	264	336	202	0	0.2	300	812
2	2,3,4	129	0	0.19	456	568	129	0	0.36	972	2896
3	2,5,2	247	0	0.16	600	800	247	0	0.81	500	2076
4	2,5,4	209	0	0.17	920	1336	209	0	0.63	1620	7632
5	3,3,2	218	0	0.18	999	1254	218	0	2	999	7662
6	3,3,4	118	0	0.3	1431	2124	118	0	0.53	3267	27,582
7	3,5,2	372	0	0.26	2475	3162	372	10%	1000	1665	20,874
8	3,5,4	214	0	0.23	3195	5292	214	0	43	5445	75,678
9	4,3,2	212	0	0.21	2784	5696	212	0	10.31	2352	36,952
10	4,3,4	158	0	0.31	3552	9568	158	0	426	7728	131,120
11	4,5,2	437	0	2	7200	15,232	447	25.5%	1000	3920	101,912
12	4,5,4	223	0	0.78	8480	25,440	223	10.3	1000	12,880	362,544
13	5,3,2	269	0	0.28	6375	33,210	269	23%	1000	4575	122,540
14	5,3,4	137	0	0.3	7575	55,480	137	2%	1000	15,075	429,880
15	5,5,2	632	23%	1000	16,875	91,310	–	–	–	7625	339,240
16	5,5,4	255	0	61	18,875	152,280	–	–	–	25,125	1,191,480
17	6,3,2	283	0	103	12,744	234,456	–	–	–	7884	322,260
18	6,3,4	185	0	4	14,472	390,960	–	–	–	26,028	1,121,232
19	6,5,2	887	32%	1000	34,200	649,896	–	–	–	13,140	893,508
20	6,5,4	303	1.30%	1000	37,080	1,083,312	–	–	–	43,380	3,110,736

Table 6
Different levels of the Taguchi factor.

Symbols	Level	Type
$N_{country}$	3	50
		100
		150
N_{imp}	3	5
		10
		20
$ITER_{seq}$	3	$N_{allop}/4$
		$N_{allop}/2$
		N_{allop}
$ITER_{assg}$	3	$N_{allop}/4$
		$N_{allop}/2$
		N_{allop}

Table 7
The orthogonal array L9.

trial	Control factor level			
	$N_{country}$	N_{imp}	$ITER_{seq}$	$ITER_{assg}$
1	$N_{country}(1)$	$N_{imp}(1)$	$ITER_{seq}(1)$	$ITER_{assg}(1)$
2	$N_{country}(1)$	$N_{imp}(2)$	$ITER_{seq}(3)$	$ITER_{assg}(2)$
3	$N_{country}(1)$	$N_{imp}(3)$	$ITER_{seq}(2)$	$ITER_{assg}(3)$
4	$N_{country}(1)$	$N_{imp}(1)$	$ITER_{seq}(3)$	$ITER_{assg}(3)$
5	$N_{country}(2)$	$N_{imp}(2)$	$ITER_{seq}(2)$	$ITER_{assg}(1)$
6	$N_{country}(2)$	$N_{imp}(3)$	$ITER_{seq}(1)$	$ITER_{assg}(2)$
7	$N_{country}(3)$	$N_{imp}(1)$	$ITER_{seq}(2)$	$ITER_{assg}(2)$
8	$N_{country}(3)$	$N_{imp}(2)$	$ITER_{seq}(1)$	$ITER_{assg}(3)$
9	$N_{country}(3)$	$N_{imp}(3)$	$ITER_{seq}(3)$	$ITER_{assg}(1)$

can be used for parameter tuning including full factorial [31] and Taguchi experiments [33]. Here, we use the Taguchi experiments to reduce the number of the tests and the time. We also consider the following control factors: the number of the countries ($N_{country}$), the imperialists (N_{imp}), the number of local search iterations for the operation-sequence part ($ITER_{seq}$), and the number of the local search iterations for the machine-selection part ($ITER_{assg}$). Different levels of these factors are shown in Table 6.

Where N_{allop} is the total operations of all the jobs. Since L9 in the orthogonal array meets our minimum requirements, we select it. Table 7 shows the orthogonal array L9.

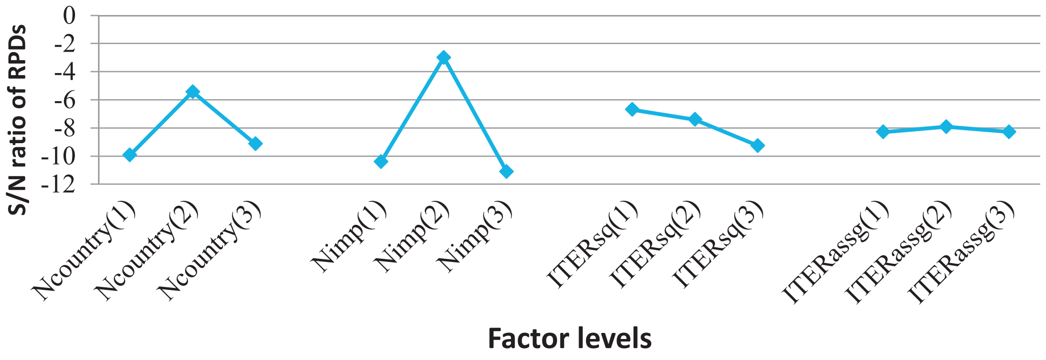

Fig. 14. The average S/N plot for every factor level.

Table 8
The results of the comparison of the algorithm with the models.

Instance	Size (n,o,m)	Sequence-based model			Position-based model			ICA	
		C_{max}	GAP	CPU time(s)	C_{max}	GAP	CPU time (s)	C_{max}	CPU time (s)
1	2,3,2	202	0	0.45	202	0	0.2	202*	1.2
2	2,3,4	129	0	0.19	129	0	0.36	129*	2.4
3	2,5,2	247	0	0.16	247	0	0.81	247*	2
4	2,5,4	209	0	0.17	209	0	0.63	209*	4
5	3,3,2	218	0	0.18	218	0	2	218*	1.8
6	3,3,4	118	0	0.3	118	0	0.53	118*	3.6
7	3,5,2	372	0	0.26	372	10%	1000	372*	3
8	3,5,4	214	0	0.23	214	0	43	214*	6
9	4,3,2	212	0	0.21	212	0	10.31	212*	2.4
10	4,3,4	158	0	0.31	158	0	426	158*	4.8
11	4,5,2	437	0	2	447	25.5%	1000	437*	4
12	4,5,4	223	0	0.78	223	10.3	1000	223*	8
13	5,3,2	269	0	0.28	269	23%	1000	269*	3
14	5,3,4	137	0	0.3	137	2%	1000	137*	6
15	5,5,2	632	23%	1000	–	–	–	632	5
16	5,5,4	255	0	61	–	–	–	381	10
17	6,3,2	283	0	103	–	–	–	283*	3.6
18	6,3,4	185	0	4	–	–	–	185*	7.2
19	6,5,2	887	32%	1000	–	–	–	761	6
20	6,5,4	303	1.3%	1000	–	–	–	289	12

A set of 60 instances is generated: 5 instances for each one of the following 12 combinations

$$n = \{5,\ 10,\ 20\},\ m = \{5,\ 10\},\ o = \{5,\ 10\}.$$

After conducting the Taguchi method, all the RPDs are converted into S/N ratios for each trial [30]. S/N of the RPDs are calculated as follows.

$$S/N = -10 \times \log_{10}\left(\mathrm{RPD}^2\right).$$

S/N results are shown in Fig. 14.

The selected levels of the factors according to both the RPD and S/N results are: $N_{country}$=100, N_{imp}=10, $ITER_{seq}$=N_{allop}/4, $ITER_{assg}$=N_{allop}/2.

5.3. Algorithms' evaluation

In this subsection, the proposed metaheuristic is compared with the MILP models in the small instances and two high-performing existing metaheuristics in large instances. To evaluate the general performance of the algorithm, the small-sized instances already solved by the models, are also solved by the ICA. The results of the algorithm in small-sized problems are shown in Table 8.

As it can be seen, the ICA obtains the optimal solution in 16 out of 20 instances (remarks with *). It is remarkable that the elapsed time of the ICA is much less than that of the models in larger-sized instances. These results prove the effectiveness of ICA in the FJSSP.

Now, the proposed ICA is compared with two existing algorithms presented for the FJSSP: the Genetic Algorithm (GA) presented by Zhang et al. [14] and the chemical reaction optimization (CRO) proposed by Li and Pan [18]. We generate 36 instances with the combination of *n, m, o* where *n*= {10, 15, 20, 25}, *m* = {5, 10, 15} *and O* = {10, 15, 20} and solve them

Table 9
The average RPDs of the algorithms presented for the instances.

Instance	Size (n,o,m)	Algorithms		
		GA	CRO	ICA
1	10,10,5	4.91	3.18	1.86
2	10,10,10	11.43	10.13	5.30
3	10,10,15	36.60	12.20	6.10
4	10,15,5	10.40	5.20	2.60
5	10,15,10	11.20	13.50	4.50
6	10,15,15	15.60	17.80	8.90
7	10,20,5	6.56	5.71	3.55
8	10,20,10	5.66	7.18	2.33
9	10,20,15	20.15	13.40	10.11
10	15,10,5	7.80	6.67	3.63
11	15,10,10	7.30	7.60	3.11
12	15,10,15	13.50	6.60	4.20
13	15,15,5	13.20	14.00	4.60
14	15,15,10	14.20	13.70	7.15
15	15,15,15	44.14	34.50	12.10
16	15,20,5	13.20	5.27	1.90
17	15,20,10	24.00	17.51	8.30
18	15,20,15	39.00	19.80	6.60
19	20,10,5	11.51	9.57	5.82
20	20,10,10	23.20	13.60	5.07
21	20,10,15	22.30	9.30	6.11
22	20,15,5	19.20	7.90	3.20
23	20,15,10	13.20	11.40	4.09
24	20,15,15	25.20	19.89	6.30
25	20,20,5	38.00	26.20	5.70
26	20,20,10	33.30	8.70	7.63
27	20,20,15	41.06	39.90	13.30
28	25,10,5	21.00	7.30	3.50
29	25,10,10	37.80	12.50	4.13
30	25,10,15	28.20	6.40	4.70
31	25,15,5	35.70	24.60	8.70
32	25,15,10	43.00	24.40	9.30
33	25,15,15	45.20	25.40	11.30
34	25,20,5	58.80	44.70	14.90
35	25,20,10	62.15	44.00	9.00
36	25,20,15	46.20	15.40	7.70
Average		25.11	15.70	6.31

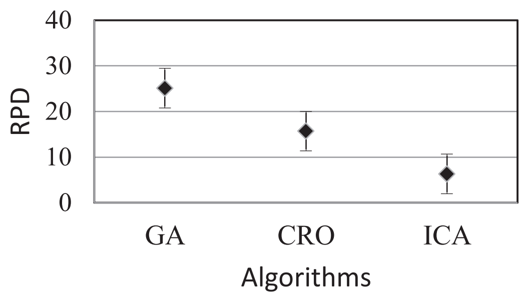

Fig. 15. The means plot and the LSD intervals for the algorithms on the generated instances.

by the algorithms with the same time termination. Each instance has been solved five times and the average RPDs of them are calculated. The average RPDs of the algorithms are shown in Table 9.

As it can be seen, the performance of the ICA is much better than those of both the GA and the CRO. The average RPD of the ICA is less than that of other algorithms. The average RPD of the ICA is 6.31% while the average RPD of the GA and the CRO are 25.11% and 15.7%, respectively. To acknowledge the better performance of the ICA, the means plot and the LSD intervals for the algorithms are obtained and shown in Fig. 15.

As the statistic results confirm, the ICA outperforms the other methods. To show whether the algorithm is robust or not, we analyze the potential impacts of the problem size (the number of the jobs and the number of the machines) on the algorithms' performance. A means plot for the interaction between the factors, the type of the method, and the number

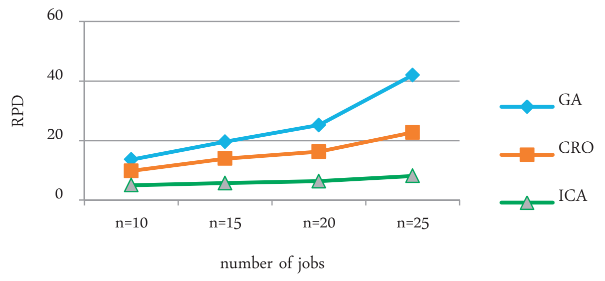

Fig. 16. The means plot for the interaction between the algorithms and the number of the jobs.

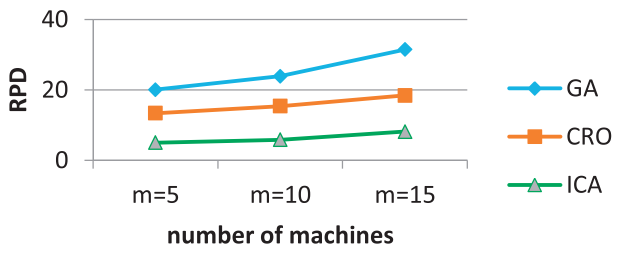

Fig. 17. The means plot for the interaction between the algorithms and the number of the machines.

of the jobs and the machines are shown in Figs. 16 and 17. The ICA shows more robustness in all instances of every size compared with other methods.

6. Conclusion and future research

Although the FJSSP is a well-studied problem, the available papers in the literature consider an unrealistic assumption of ignoring the transportation times of the jobs between the machines. This paper considered the transportation times in the FJSSP. Two MILP models were presented for this problem: sequence-based and position-based. These models were deployed to solve small-sized instances using the GAMS software. Their results disclosed that the sequence-based model has a better performance than the position-based model. As the FJSSP is NP-hard, we developed a metaheuristic algorithm (ICA) to solve the problems. The ICA has simple and efficient structures, making the algorithm more favorable. Recently, the ICA has been used by many researchers in various fields. Because of these advantages of the ICA, we developed an adaptation of the imperialist competitive algorithm hybridized by a local search to solve the problem. After tuning the parameters by the Taguchi method, the algorithm was compared with the MILP models on small instances. Finally, the ICA was compared with two existing evolutionary algorithms on large instances. The results demonstrated that the ICA provides a much better performance than the other algorithms.

The following directions are recommended for future research. Since the problem under the consideration assumes an unlimited number of transporters, modeling the problem with limited transporters can be interesting research. Besides, the other assumption of our models is that there is no machine failure. Hence, it is recommended to incorporate the machine failure into the model. We can also consider due time-oriented models such as minimizing the penalties for total tardiness and the machine's idle times. Also, since the ICA has shown excellent performance in different problems, we strongly recommend deploying this method in other scheduling problems.

Reference

[1] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice-Hall, Englewood cliffs, NJ, 1995.
[2] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow-shop and job shop scheduling, Math. Oper. Res. 1 (2) (1976) 117–129.
[3] P. Bruker, R. Schlie, Job-shop scheduling with multi-purpose machines, Computing 45 (1990) 369–375.
[4] W.J. Xia, Z.M. Wu, An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, Comput. Ind. Eng. 48 (2005) 409–425.
[5] I.C. Choi, D.S. Choi, A local search algorithm for job-shop scheduling problems with alternative operations and sequence-dependent setups, Comput. Ind. Eng. 42 (1) (2002) 43–58.
[6] J. Gao, M. Gen, L. Sun, Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, J. Intell. Manuf. 17 (4) (2006) 493–507.
[7] N. Imanipour, S.H. Zegordi, A heuristic approach based on Tabu Search for early/tardy flexible job-shop problems, Sci. Iran. 13 (1) (2006) 1–13.
[8] P. Fattahi, M.S. Mehrabad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, J. Intell. Manuf. 18 (3) (2007) 331–342.
[9] P. Fattahi, F. Jolai, J. Arkat, Flexible job shop scheduling with overlapping in operations, Appl. Math. Model. 33 (7) (2009) 3076–3087.
[10] C. Ozguven, L. Ozbakir, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Appl. Math. Model. 34 (6) (2010) 1539–1548.
[11] C. Ozguven, Y. Yavuz, L. Ozbakir, Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence-dependent setup times, Appl. Math. Model. 36 (2) (2012) 846–858.

[12] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, Ann. Oper. Res. 41 (1-4) (1993) 157–183.
[13] H. Chen, J. Ihlow, C. Lehmann, A genetic algorithm for flexible job-shop scheduling, in: IEEE International Conference on Robotics And Automation, 2, Detroit, 1999, pp. 1120–1125.
[14] G. Zhang, L. Gao, Y. Shi, An effective genetic algorithm for the flexible job-shop scheduling problem, Expert Syst. Appl. 38 (2011) 3563–3573.
[15] M. Mastrolilli, L.M. Gambardella, Effective neighbourhood functions for the flexible job shop problem, J. Sched. 3 (1) (2000) 3–20.
[16] M. Yazdani, M. Amiri, M. Zandieh, Flexible job-shop scheduling with parallel variable neighborhood search algorithm, Expert Syst. Appl. 37 (2010) 678–687.
[17] W. Xia, Z. Wu, An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, Comput. Ind. Eng. 48 (2) (2005) 409–425.
[18] J. Li, Q. Pan, Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity, Appl. Soft. Comput. 12 (2012) 2896–2912.
[19] V.A. Strusevich, A heuristic for the two-machine open-shop scheduling problem with transportation times, Discrete Appl. Math. 93 (1999) 287–304.
[20] J. Hurink, S. Knust, Tabu search algorithms for job-shop problems with a single transport robot, Eur. J. Oper. Res. 162 (2005) 99–111.
[21] M.A. Langston, Interstage transportation planning in the deterministic flow-shop environment, Oper. Res. 35 (1987) 556–564.
[22] B. Naderi, M. Zandieh, A. Khaleghi Ghoshe Balagh, V. Roshanaei, An improved simulated annealing for hybrid flow-shops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness, Expert Syst. Appl. 36 (2009) 9625–9633.
[23] M. Boudhar, A. Haned, Preemptive scheduling in the presence of transportation times, Comput. Oper. Res. 36 (2009) 2387–2393.
[24] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: CEC IEEE Congress on Evolutionary Computation, 2007.
[25] Z. Ardalan, S. Karimi, O. Poursabzi, B. Naderi, A novel imperialist competitive algorithm for generalized traveling salesman problems, Appl. Soft. Comput 26 (2015) 546–555.
[26] M. Watanabe, K. Ida, M. Gen, A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem, Comput. Ind. Eng. 48 (2005) 743–752.
[27] K.M. Lee, T. Yamakawa, K.M. Lee, A genetic algorithm for general machine scheduling problems, Int. J. Knowl. Based Electron. 2 (1998) 60–66.
[28] L. Davis, Applying adaptive algorithms to epistatic domains, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1985, pp. 162–164.
[29] K.S. Amirthagadeswaran, V.P. Arunachalam, enhancement of performance of genetic algorithm for job shop scheduling problems through inversion operator, Int. J. Adv. Manuf. Technol. 32 (2007) 780–786.
[30] B. Naderi, S.M.T. Fatemi Ghomi, M. Aminnayeri, A high performing meta-heuristic for job-shop scheduling with sequence-dependent setup times, Appl. Soft. Comput. 10 (2010) 703–710.
[31] D.C. Montgomery, Design and Analysis of Experiments, 5th, Wiley, New York, 2000.
[32] A. Bagheri, M. Zandieh, Iraj Mahdavi, M. Yazdani, An artificial immune algorithm for the flexible job-shop scheduling problem, Future Gener. Comput. Syst. 26 (4) (2010) 533–541.
[33] G. Taguchi, Introduction to Quality Engineering, White Plains: Asian Productivity Organization/UNIPUB, 2000.
[34] S.H.A. Rahmati, M. Zandieh, M. Yazdani, Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem, Int. J. Adv. Manu. Tech. 8 (2013) 5–8 64.
[35] B. Mohammadi-ivatloo, A. Rabiee, A. Soroudi, M. Ehsan, Imperialist competitive algorithm for solving non-convex dynamic economic power dispatch, Energy 44 (2012) 228–240.
[36] A. Rahimi, H. Karimi, B. Afshar-Nadjafi, Using meta-heuristics for project scheduling under mode identity constraints, Appl. Soft Comput. 13 (4) (2013) 2124–2135.
[37] M. Mohammadi, S.A. Torabi, R. Tavakkoli-Moghaddam, Sustainable hub location under mixed uncertainty, Transport. Res. E 62 (2014) 89–115.
[38] K. Devika, A. Jafarian, V. Nourbakhsh, Designing a sustainable closed-loop supply chain network based on triple bottom line approach: a comparison of meta-heuristic s hybridization techniques, Eur. J. Oper. Res. 235 (3) (2014) 594–615.
[39] M. Ghasemi, S. Ghavidel, S. Rahmani, A. Roosta, H. Falah, A novel hybrid algorithm of imperialist competitive algorithm and teaching learning algorithm for optimal power flow problem with non-smooth cost functions, Eng. Appl. Artif. Intell. 29 (2014) 54–69.
[40] H. Linzhi, D. Haibin, W. Yin, Hybrid bio-inspired lateral inhibition and imperialist competitive algorithm for complicated image matching, Opt. Int. J. Light Electron Opt. 125 (2014) 414–418.
[41] V. Roshanaei, H.ElMaraghy, Ahmed Azab, Mathematical modelling and a meta-heuristic for flexible job shop scheduling, Int. J. Prod. Res (2013) 6247–6274.
[42] Barnes J. W, Cambers. J. B, Flexible job shop scheduling by Tabu search, graduate program in operations research and industrial engineering, (1996), the University of Texas in at Austin, Technical report series, ORP96-09.
[43] P. Brandimarte, Routing and scheduling in a flexible job shop by Tabu search, Ann. Oper. Res. 22 (1993) 158–184.
[44] B. Naderi, A. Ahmadi Javid, F. Jolai, Permutation flow-shops with transportation times: mathematical models and solution methods, Int. J. Adv. Manuf. Technol. 46 (2010) 631–647.
[45] N.B. Ho, J.C. Tay, M. Edmund, K. Lai, An effective architecture for learning and evolving flexible job-shop schedules, Eur. J. Oper. Res. 179 (2007) 316–333.