*Journal of Quality Engineering and Production Optimization*

*Vol. 8, No. 1, Winter & Spring 2023*

*http://jqepo.shahed.ac.ir*

*Research Paper*

DOI: 10.22070/jqepo.2023.16882.1246

# Efficient scheduling of a no-wait flexible job shop with periodic maintenance activities and processing constraints

**Kasra Mahdavi [1], Mohammad Mohammadi [1*], Fardin Ahmadizar [2]**

[1] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*

[2] *Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran*

**** Corresponding Author:*** *Mohammad Mohammadi (Email: Mohammadi@khu.ac.ir)*

***Abstract*** *–Flexible job-shop scheduling problem (F-JSP) is an expansion of the job shop scheduling problem (JSP) which allows an operation to be fulfilled by any machine among a set of accessible machines at each stage. This paper investigates a no-wait F-JSP (NW-F-JSP) with machines accessibility restrictions for maintenance activities and machines processing capability to minimize total weighted tardiness. The study is organized in two phases. Firstly, a novel nonlinear mathematical model is developed for the supposed problem, and then it is converted into a linear mathematical model using techniques found in the literature. Since the structure of the problem is NP-hard, an imperialist competitive algorithm is proposed in the second phase to solve large instances of the problem. In the proposed algorithm, an effective solution representation with an efficient and greedy decoding methodology is adopted to reduce the search space. Numerical experiments are used to appraise the performance of the developed algorithm. It is inferred that in small instances, solving the mathematical model by GAMS leads to the optimal solution. Still, with an increased instance size, this method loses its efficiency and the ICA approach performs better under these conditions.*

***Keywords****– Flexible job shop, no-wait, maintenance activities, Imperialist competitive algorithm.*

## I. INTRODUCTION

Job shop scheduling problem (JSP) is a type of scheduling used in various production environments. The JSP was first introduced by Manne (1960). In this study, it is proven that JSP, known as the NP-hard optimization problem in production scheduling literature, is highly complex. To the best of our knowledge, there is no methodology in the literature that is able to solve large instances in a reasonable amount of time.

Due to the restrictions and special conditions in each production environment and their specific constraints, production scheduling requires consideration of these constraints, which complicates the scheduling problem. One of the manufacturing industries that has special conditions and restrictions is the perishable products industry. Delays during the production of perishable products can be very destructive. As a result, perishables are produced without delay, packaged and stored immediately. If the production system is in the form of a job shop, the scheduling problem will become a no-wait job shop scheduling problem (NW-JSP). In addition, due to the possible machines malfunction, it is imperative to incorporate machines maintenance activities into the model. Therefore, it can be stated that in the

*Manuscript Received:10-October-2022 & Revised: 14-December-2022 & Accepted: 14-February-2023*
*ISSN: 2423-3781*

production of perishable products, if the production system is in the form of a job shop, various restrictions must be considered.

In this paper, a no-wait flexible job shop scheduling problem (NW-F-JSP) with processing constraints is investigated. This scheduling problem is of special importance both from a theoretical and practical point of view. From a theoretical point of view, considering the machines capability and machines maintenance activities in NW-F-JSP shows that a number of feasible solutions, albeit limited, are available. In situations where there are such constraints in a flexible job shop scheduling problem (F-JSP), such solutions can be applied. From a practical perspective, one of the production environments, where conditions are very similar to the problem discussed in this study, is the perishable food manufacturing industry, where the production process is predominately carried out in the form of JSP. In such industries, to prevent food spoilage, it is necessary to eliminate the waiting time during production. On the other hand, due to the probability of machine malfunction, the consideration of machines maintenance activities and machines capabilities constraints in any production environment is inevitable. Therefore, the applicability of this problem with the considered constraints is clear in the perishable food manufacturing industry.

To the best of our knowledge, this is the first study on the F-JSP where machines processing capability, machines periodic maintenance activities, and no-wait constraints are simultaneously considered. The contributions are described as follows:

Investigating an NP-hard scheduling problem that is widely used in the perishable food manufacturing industry.

- A non-linear mathematical model based on the precedence variable is established for the NW-F-JSP with processing constraints.
- The proposed model is linearized by techniques in the literature to be solved by linear solvers.
- An Imperialist competitive algorithm (ICA approach) is customized to solve the NW-F-JSP.

The other sections of the paper are organized as follows: In Section II, the latest and related works are presented. Section III presents the mathematical model and its linearization process for the problem. In Section IV, the ICA approach is proposed to solve large instances of the problem. Computational results are discussed in Section V. Finally, conclusions and future research are presented in Section VI.

## II. LITERATURE REVIEW

In recent years, many researchers have studied JSP with various constraints. In this section, the newest related works are reviewed. El Khoukhi et al. (2017) investigated a F-JSP with machines accessibility restrictions to minimize the greatest termination time. A mathematical model for this problem was proposed, and due to its complexity, a new optimization algorithm relying on the ant nest algorithm was developed. The F-JSP with machines accessibility restrictions to minimize the greatest termination time was studied by Zandieh et al. (2017), and an ameliorated ICA approach for large instances was proposed. A study was conducted by Yazdani et al. (2017) on a JSP to minimize the sum of maximum tardiness and maximum earliness. A mathematical model was developed, and a new optimization approach was proposed relying on the ICA approach. A study was undertaken by Lu et al. (2017) on a multi-objective F-JSP with controllable processing times. In this study, minimizing the sum of consuming resources and the greatest termination time were designated as objective functions. A new multi-objective meta-heuristic algorithm called MODVOA was developed. A study was conducted by Benttaleb et al. (2018) on a JSP with two machines where one of the machines was out of reach in a certain period. In this study, the objective function was minimizing the greatest termination time. The optimality of Jackson's algorithm was investigated, and a heuristic algorithm was designed using Jackson's law. Subsequently, they proposed a branch and bound algorithm for the problem. A new cyclic algorithm relying on Tabu search was proposed by Fattahi et al. (2018) to improve the exploration and exploitation powers of certain solution encodings suggested in the literature. The effectiveness of the proposed solution representation was demonstrated in this research through the solution of several instances. Bürgy and Bülbül (2018) studied a JSP with the irregular objective function of minimizing the sum of convex costs depending on the operations start time and proposed a Tabu search algorithm for it. A mathematical model was proposed for a F-JSP with sequence-dependent setup times to minimize the greatest termination time. Additionally, a Tabu search algorithm was developed based on new neighborhood search functions and various structures by Shen et al. (2018). A JSP with the objective function of minimizing the greatest termination time, in which machines are not always available and become unavailable at intervals, was studied by Tamssaouet et al. (2018). In this study, simulated annealing and Tabu search algorithms with neighborhood functions for real-size instances of the problem were developed. García-León et al. (2019) proposed a local search approach for the multi-objective F-JSP to obtain Pareto solutions for any combination of regular functions. An ameliorated Jaya algorithm for an F-JSP with the objective function of minimizing the greatest termination time was developed by Caldeira and Gnanavelbabu (2019). In this problem, machines' setup time and transfer time between machines were considered. An ameliorated multi-objective Genetic algorithm for the problem of minimizing the greatest termination time and energy consumption in a multi-objective F-JSP with energy consumption and transportation restrictions was developed by Dai et al. (2019). Shen et al. (2019) investigated a F-JSP, in which the jobs' processing time is variable and depends on the start time of their processing. In this study, the objective function is minimizing the greatest termination time and the amount of energy consumed by machines. For this problem, a hybrid multi-objective algorithm called MOHPIOSA was presented by them. Samarghandi (2019) studied a NW-JSP with delivery deadline limitations and the objective function of minimizing the greatest termination time. The problem was transformed into another problem, and a mathematical model for both of them was presented. Subsequently, a genetic algorithm was developed to solve large instances. Miyata et al. (2019) studied a no-wait flow shop scheduling problem (NW-FSP) with dependent sequenced setup times and machines preventive maintenance to minimize the greatest termination time. In this problem, a new policy for preventive maintenance was postulated, with its parameters based on the Weibull distribution. In this study, constructive heuristics were developed for the proposed problem. A study was conducted by Samarghandi and Firouzi Jahantigh (2019) on a NW-FSP with due date constraints aimed at minimizing the greatest termination time. Two mathematical models were proposed, and a Constraint Programming Model was employed.

Zhang et al. (2020) investigated a NW-FSP with the objective function of minimizing the greatest termination time. The Discrete Migratory Bird Optimization (MBO) algorithm was developed by them to achieve high-quality solutions. A JSP was studied by Ahmadian et al. (2020) in which delivery date is considered for each job. In this problem, any discrepancy between the job's completion time and its delivery date is designated as a penalty, and the objective function is to minimize the sum of earliness and tardiness. A Variable Neighborhood Search (VNS) algorithm was developed for this problem. An ameliorated Genetic algorithm for a multi-objective F-JSP was developed by Zhang et al. (2020). In this study, machines setup time and jobs transfer time between machines were considered. Li et al. (2020) developed an ameliorated Jaya algorithm for a F-JSP, in which machine setup times and jobs transfer time between machines are considered. Ying and Lin (2020) studied a NW-JSP to minimize the greatest termination time. The MSA-BST algorithm, which is based on the Simulated Annealing algorithm, was developed by them. A F-JSP was studied by Zhu and Zhou (2020b) with job priority restrictions in which the processing time of jobs was expressed as an interval and the objective function was to minimize the interval length that is obtained for greatest termination time. A new optimization algorithm called SLHO was developed by them for real-size instances of the problem. An impressive evolutionary grey wolf optimizer for multi-objective F-JSP with job priority restrictions was developed by Zhu and Zhou (2020a). In this problem, the objective function was to minimize the greatest termination time and maximize the workload of machines simultaneously. An ameliorated memetic algorithm for the F-JSP with transportation times to minimize the greatest termination time was proposed by Zhang et al. (2020). A multi-objective JSP was studied by Li et al. (2020) in a robotic cell. In this problem, each job has a specific due date as a time window and the objective function is to minimize the greatest termination time and the total earliness and tardiness simultaneously. A TLBO algorithm was developed by them for large instances of the problem. A two-stage Genetic algorithm for a F-JSP with sequence-dependent setup times was developed by Defersha and Rooyani (2020). In this problem, the machines are available for processing operations at different times and each machine needs time to cool down after processing each operation. Ozolins (2020) studied a NW-JSP to minimize greatest termination time. In this study, a new exact algorithm was developed to solve benchmark instances within a sensible time limit. A NW-JSP with due date and subcontracting cost constraints was studied by Gao et al. (2021). Two mathematical models were proposed by them. Then, an artificial bee colony algorithm was developed based on a rolling timeline. A F-JSP with machine capacity, time lags, holding times, and sequence-dependent setup times was studied by Boyer et al. (2021). A mixed-integer linear programming and a constraint programming (CP) models were proposed by them to represent the problem, and a meta-heuristic based on a Greedy Randomized Adaptive Search Procedure was developed to solve real-size instances of the problem. A NW-JSP was studied by Valenzuela-Alcaraz et al. (2022) to minimize the greatest termination time and a cooperative coevolutionary algorithm was proposed to solve large instances. Weng et al. (2022) studied a F-JSP requiring operations to be performed by either a worker or a machine and to perform a machine operation, two workers are needed. The scheduling problem was modeled by them, and four methods that form a real-time scheduling and control system for JIT production were proposed. Fan and Su (2022) investigated a JSP with conveyor-based ongoing flow transferor to minimize greatest termination time. In this study, the jobs are processed on the machines which are connected in series via the conveyor. A mathematical model of the problem to find exact solutions in small instances was presented by them, and a Simulated Annealing algorithm with NGS scheme was developed to solve larger instances. Zhu et al. (2022) studied a NW-FSP with due windows to minimize the total weighted earliness and tardiness. In this problem, a concept called factory has been proposed, which exists in a specific number and includes machines for processing operations of jobs. In this scheduling problem, each job was assigned to a factory to process its operations , and a new approach to solve large instances was proposed by them. Nohair et al. (2022) studied a non-delay JSP with the objective of minimizing greatest termination time. A matrix heuristic was developed by them to generate non-delay schedules that are computationally swift to implement. Winklehner and Hauder (2022) investigated a F-JSP with periodic machines maintenance activities and processing constraints as a real-world problem. A constraint programming approach to minimize the total completion times was developed by them. Valenzuela-Alcaraz et al. (2022) proposed a cooperative algorithm approach for NW-JSP to minimize the greatest termination time. In table 1, all of the studies that are reviewed above are classified.

**Table 1. summarized Literature review**

| No | Author | Year | Type | Objective function | Constraints | | | | | | | Solving approach |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | No-wait | Machine availability | Machine capability | Job transportation time | Setup times | Type of parallel in each stage | Other | |
| 1 | El Khoukhi et al. | 2017 | F-JSP | Cmax | | × | | | | Pm | Priority restrictions | Based on ACO |
| 2 | Zandieh et al. | 2017 | F-JSP | Cmax | | × | | | | Qm | | ICA |
| 3 | Yazdani et al. | 2017 | JSP | $E_{max} + T_{max}$ | | | × | | | - | | Based on ICA approach |
| 4 | Lu et al. | 2017 | F-JSP | Cmax + Minimizing consuming resources | | | | | | Pm | Control the processing time of jobs by allocating resources | MODVOA |
| 5 | Benttaleb et al. | 2018 | JSP | Cmax | | × | | | | | | Algorithm based on Jackson's rule |

**Continue Table 1. summarized Literature review**

| No | Author | Year | Type | Objective function | Constraints | | | | | | | Solving approach |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | No-wait | Machine availability | Machine capability | Job transportation time | Setup times | Type of parallel in each stage | Other | |
| 6 | Shen et al. | 2018 | F-JSP | Cmax | | | | | × | Pm | | Tabu search |
| 7 | Tamssaouet et al. | 2018 | JSP | Cmax | | × | | | | - | | SA - TS |
| 8 | Bürgy and Bülbül | 2018 | JSP | Minimizing convex costs | | | | | | - | Considering the costs related to the start of operations | Tabu search |
| 9 | Fattahi et al. | 2018 | JSP | Cmax | | | | | | - | | new cyclic algorithm based on TS |
| 10 | Miyata et al. | 2019 | FSP | Cmax | × | × | | | × | - | Flexible preventive maintenance | Constructive heuristics |
| 11 | García-León et al. | 2019 | F-JSP | A set of regular objective functions | | | × | | | Pm | | Local search approach to obtain Pareto solutions |
| 12 | Caldeira and Gnanavelbabu | 2019 | F-JSP | Cmax | | | | × | × | Pm | | Jaya algorithm |
| 13 | Wu et al. | 2019 | F-JSP | Minimizing energy consumption + Cmax | | | × | | | Rm | The jobs processing times are variable | MOHPIOSA |
| 14 | Dai et al. | 2019 | F-JSP | Minimizing energy consumption + Cmax | | | × | × | | Pm | | NSGA II |
| 15 | Chen et al. | 2020 | FSP | Cmax | × | × | | | | - | Scheduling with two machines | Extremely NP-hard in the case of two machines |
| 16 | Zhang et al. | 2020 | FSP | Cmax | × | | | | | - | | MBO |
| 17 | Li et al. | 2020 | JSP | Minimizing the sum of completion time | × | | | | | - | | A heuristic algorithm called PBIG |
| 18 | Zhang et al. | 2020 | F-JSP | Cmax | | | | × | × | Pm | | GA |

**Continue Table 1. summarized Literature review**

| No | Author | Year | Type | Objective function | Constraints | | | | | | | Solving approach |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | No-wait | Machine availability | Machine capability | Job transportation time | Setup times | Type of parallel in each stage | Other | |
| 19 | Li et al. | 2020 | F-JSP | Minimizing energy consumption + Cmax | | | | × | × | Pm | | Jaya algorithm |
| 20 | Ahmadian et al. | 2020 | JSP | JIT | | | | | | - | Independent due date for each job | VNS |
| 21 | Zhu and Zhou | 2020a | F-JSP | Maximizing machines workload + Cmax | | | | | | Pm | Priority restrictions | GWO |
| 22 | Defersha and Rooyani | 2020 | F-JSP | Cmax | | × | × | | × | Pm | machines are available at different times | GA |
| 23 | Zhang et al. | 2020 | F-JSP | Cmax | | | | × | | Qm | | Ameliorated memetic algorithm |
| 24 | Ying and Lin | 2020 | JSP | Cmax | × | | | | | - | | MSA-BST |
| 25 | Zhu and Zhou | 2020b | F-JSP | Minimizing the length of Cmax interval | | | | | | Pm | Priority restrictions | New approach called SLHO |
| 26 | Zhang et al. | 2020 | F-JSP | Cmax | | | | × | | Qm | | Ameliorated GA |
| 27 | Li et al. | 2020 | JSP | Cmax + JIT | | | | Using robots | | - | Job due date as a time window | New effective approach called IMOTLBO |
| 28 | Ozolins | 2020 | JSP | Cmax | × | | | | | - | | New exact algorithm based on DP |
| 29 | Gao et al. | 2021 | JSP | Cmax and subcontracting cost | × | | | | | - | Subcontracting strategy to satisfy the deadlines | ABC based on a rolling timeline |
| 30 | Boyer et al. | 2021 | F-JSP | Cmax | | | | | × | Qm | With machine capacity and time lags | Meta-heuristic relying on a greedy procedure |
| 31 | Valenzuela-Alcaraz et al. | 2022 | JSP | Cmax | × | | | | | - | | Cooperative coevolutionary algorithm |

**Continue Table 1. summarized Literature review**

| **No** | **Author** | **Year** | **Type** | **Objective function** | **Constraints** | | | | | | | **Solving approach** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | **No-wait** | **Machine availability** | **Machine capability** | **Job transportation time** | **Setup times** | **Type of parallel in each stage** | **Other** | |
| 32 | Weng et al. | 2022 | F-JSP | JIT | | | | | × | Rm | Each operation needs worker and worker like machine has key role | Proposed four method that form a realtime scheduling. |
| 33 | Fan and Su | 2022 | JSP | Cmax | | | | | | - | The jobs are processed on the contiguous machines by the conveyor | Simulated Annealing algorithm with NGS scheme |
| 34 | Zhu et al. | 2022 | FSP | Weighted JIT | × | | | | | - | Considering time window for jobs due dates | A discrete algorithm |
| 35 | Nohair et al. | 2022 | JSP | Cmax | | | | | | - | Machines are never kept vacant while job is waiting | New matrix heuristic |
| 36 | Winklehner and Hauder | 2022 | F-JSP | Minimizing the total completion times | | × | × | | × | Rm | Deadline for each job<br><br>Release date for each job | Constraint programming approach |
| *** | **Current study** | | F-JSP | Minimizing sum of weighted tardiness | × | × | × | | | Rm | Independent periodic maintenance activities for each machine | ICA based on greedy decoding methodology |

In summary, the reviewed scientific works show that no single paper exists covering F-JSP with unrelated parallel machines in each stage, machines periodic maintenance activities, no-wait constraint and machines capability to minimize sum of weighted tardiness.

## III. PROBLEM DESCRIPTION AND FORMULATIONS

### *A. Problem explanation*

In a F-JSP, there are a set of machines and a set of jobs that have to be processed on the machines. In this problem, m machines and n jobs are considered. Each job consists of a sequence of operations where they are permitted to be processed on any among a set of accessible machines. The other assumptions of the problem are as follows:

- All jobs and machines are accessible from the beginning, each machine can only execute one operation at a given time.
- Each job has a specific processing path.
- To process each job, it may not require machines during all stages.
- There is no way to process a job on multiple machines simultaneously.
- There is no way to interrupt an operation once started (Preemption is not allowed).
- Waiting time between two sequential operations of the same job is not allowed.
- Each job has an independent due date so if it is fulfilled later than the due date, a penalty will be imposed.

- Machines can not necessarily process all operations.
- The type of parallel machines in each stage is unrelated.
- The setup time of machines is considered as a part of the processing time of the operations.
- Machines are periodically unavailable for maintenance activities.
- The length of each unavailability interval is specified.

The objective is to identify a feasible schedule that minimizes the total weighted tardiness.

## *B. Problem formulations*

The notation describing the indices, parameters, and decision variables used in the models are as follows:

Indices:

i, h: jobs' index (1, …, n)

j: operations' index (1, … ,$J_i$)

k: machines' index (1, …, m)

r: unavailability interval's index

Parameters:

n: jobs' total number

m: machines' total number

$J_i$: total number of operations of job i

$pr_{kij}$: processing time of $o_{ij}$ if performed on machine k

$d_i$ : due date of job i

$w_i$ : weight of job i

$SM_{kr}$ : starting time of rth unavailability interval on machine k

$FM_{kr}$ : finishing time of rth unavailability interval on machine k ($FM_{kr} - SM_{kr} = T$)

M: a large number

Decision variables:

$T_i$: tardiness of job i

$V_{ijk}$: $V_{ijk}$ is 1 if $o_{ij}$ performed on machine k; otherwise $V_{ijk}$ is 0.

$Z_{ijhgk}$: $Z_{ijhgk}$ is 1 if $o_{ij}$ precedes operation $o_{hg}$ on machine k; otherwise $Z_{ijhgk}$ is 0.
$cm_{ij}$: completion time of operation $o_{ij}$

$B_{ijkr}$: binary variable in inaccessibility restrictions

In this section, an approach called the precedence variable-based model is used to present a mathematical model for the problem. This approach relies on the precedence variable $Z_{ijhgk}$, introduced by Manne (1960). It denotes the sequence of operations assigned to the same machine. $Z_{ijhgk}$ is equal to one if operation $o_{ij}$ precedes operation $o_{hg}$ on machine k; otherwise $Z_{ijhgk}$ is equal to zero. Note that operation $o_{ij}$ is not necessarily positioned immediately before operation $o_{hg}$ when $Z_{ijhgk}$ is equal to one. For this type of variable, it has to be defined only i < h because $Z_{ijhgk} = 1 - Z_{hgijk}$. According to this approach, a precedence variable-based model for this problem is developed.

This kind of model was first proposed by Gao et al. (2006) to formulate F-JSP, and we have adopted it for our F-JSP. The objective function is to minimizing total weighted tardiness:

$$\text{Min} \sum_i (w_i * T_i) \tag{1}$$

The following constraints compel each job to pursue a specified operation sequence and guarantee the no-wait constraint:

$$cm_{ij} - cm_{ij-1} \geq pr_{kij}.V_{ijk} \qquad , \forall\, i, k, \forall j = 2, \dots, J_i \tag{2}$$

$$cm_{ij} - cm_{ij-1} \leq pr_{kij}.V_{ijk} \qquad , \forall\, i, k, \forall j = 2, \dots, J_i \tag{3}$$

Constraint 4 ensures the completion time of the first operation of job i equal to be at least the processing time of $o_{ij}$:

$$cm_{ij} \geq pr_{kij}.V_{ijk} \qquad , \forall\, i, j = 1 \quad \forall k \in M_{ij} \tag{4}$$

The following constraints are disjunctive constraints:

$$(cm_{hg} - cm_{ij} - pr_{khg}).V_{hgk}.V_{ijk}.\ Z_{ijhgk} \geq 0 \tag{5}$$

$$, \forall\, i, h, j, g, \quad \forall k \in M_{ij} \cap M_{hg}$$

$$(cm_{ij} - cm_{hg} - pr_{kij}).V_{ijk}.V_{hgk}.\ Z_{hgijk} \geq 0 \tag{6}$$

These constraints represent that the operation $o_{hg}$ should not be started before the completion of the operation $o_{ij}$ or that the operation $o_{hg}$ must be completed before the start of the operation $o_{ij}$ if they are assigned to the same machine k. Constraints 5 and 6 are nonlinear and should be linearized. For this purpose, the nonlinear expression $V_{hgk}.V_{ijk}.\ Z_{ijhgk}$ is first linearized by variable $O_{ijhgk}$:

$$\begin{aligned} &O_{ijhgk} \leq V_{hgk} \\ &O_{ijhgk} \leq V_{ijk} \\ &O_{ijhgk} \leq Z_{ijhgk} \\ &O_{ijhgk} \geq V_{hgk} + V_{ijk} + Z_{ijhgk} - 2 \end{aligned} \tag{7}$$

Therefore, constraints 5 and 6 become as follows:

$$cm_{hg}.O_{ijhgk} - cm_{ij}.O_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \tag{8}$$

$$, \forall\, i, h, j, g, \quad \forall k \in M_{ij} \cap M_{hg}$$

$$cm_{ij}.O_{ijhgk} - cm_{hg}.O_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \tag{9}$$

Then the nonlinear expressions $cm_{ij}.O_{ijhgk}$ and $cm_{hg}.O_{ijhgk}$ should be linearized .Expression $cm_{ij}.O_{ijhgk}$ is linearized as follows:

$$\begin{aligned} &D_{ijhgk} \leq cm_{ij} \\ &D_{ijhgk} \leq M, O_{ijhgk} \\ &D_{ijhgk} \geq cm_{ij} - M(1 - O_{ijhgk}) \end{aligned} \qquad (10)$$

And expression $cm_{hg}. O_{ijhgk}$ is linearized as follows:

$$\begin{aligned} &BS_{ijhgk} \leq cm_{hg} \\ &BS_{ijhgk} \leq M, O_{ijhgk} \\ &BS_{ijhgk} \geq cm_{hg} - M(1 - O_{ijhgk}) \end{aligned} \qquad (11)$$

Finally, the linear equivalent of constraints 5 and 6 are as follows:

$$\begin{aligned} &BS_{ijhgk} - D_{ijhgk} - pr_{khg}, O_{ijhgk} \geq 0 \\ &D_{ijhgk} - BS_{ijhgk} - pr_{khg}, O_{ijhgk} \geq 0 \qquad , \forall\, i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \\ &\text{And constraints 7. 10 and 11} \end{aligned} \qquad (12)$$

The following constraint clarifies that one machine ought to be selected from a set of available machines for each operation:

$$\textstyle\sum_{k \in M_{ij}} V_{ijk} = 1 \qquad , \forall\, i. j \qquad (13)$$

Constraint 14 enforces to be selected one of two preference relationships.

$$Z_{ijhgk} + Z_{hgijk} = V_{ijk}, V_{hgk} \qquad , \forall\, i. h. j. g. \ \forall k \in M_{ij} \cap M_{hg} \qquad (14)$$

Constraints 14 is nonlinear and should be linearized. For this purpose, the nonlinear expression $V_{ijk}. V_{hgk}$ is linearized by variable $F_{ijhgk}$:

$$\begin{aligned} &F_{ijhgk} \leq V_{hgk} \\ &F_{ijhgk} \leq V_{ijk} \\ &F_{ijhgk} \geq V_{hgk} + V_{ijk} - 1 \end{aligned} \qquad (15)$$

Finally, the linear equivalent of constraint 14 are as follows:

$$\begin{aligned} &Z_{ijhgk} + Z_{hgijk} = F_{ijhgk} \\ &\text{And constraints 15} \end{aligned} \qquad , \forall\, i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (16)$$

Constraints 17 to 22 describe unavailability intervals for machines and ensure that each operation $o_{ij}$ can be processed between intervals when the machine is active.

$$(cm_{ij} - pr_{kij}), V_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \qquad (17)$$

$$cm_{ij}, V_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \qquad (18)$$

$$(cm_{ij} - pr_{kij}), V_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \qquad (19)$$

$$cm_{ij}, V_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \qquad , \forall\, i. j. k. r \qquad (20)$$

$$(cm_{ij} - pr_{kij}), V_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \qquad (21)$$

$$cm_{ij}, V_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \qquad (22)$$

In these constraints, the expression $cm_{ij}. V_{ijk}$ is nonlinear, which becomes linear as follows:

$$\begin{aligned} & VC_{ijk} \le cm_{ij} \\ & VC_{ijk} \le M, V_{ijk} \\ & VC_{ijk} \ge cm_{ij} - M(1 - V_{ijk}) \end{aligned} \qquad (23)$$

Finally, the linear equivalent of constraints 17 to 22 are as follows:

$$VC_{ijk} - pr_{kij}, V_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \qquad (24)$$

$$VC_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \qquad (25)$$

$$VC_{ijk} - pr_{kij}, V_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \qquad (26)$$

$$VC_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \qquad , \forall\, i\,.\, j.\, k.\, r \qquad (27)$$

$$VC_{ijk} - pr_{kij}, V_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \qquad (28)$$

$$VC_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \qquad (29)$$

And constraints 23

Finally with constraint 30 can determine the tardiness of each job:

$$T_i \ge cm_{ij} - d_i \qquad , \forall\, i\,.\, j = J_i \qquad (30)$$

According to mentioned above, the linearized mathematical model is as follows:

$$\text{Min} \sum_i (w_i * T_i) \qquad (31)$$

S, T,

$$cm_{ij} - cm_{ij-1} \ge pr_{kij}, V_{ijk} \qquad (32)$$

$$cm_{ij} - cm_{ij-1} \le pr_{kij}, V_{ijk} \qquad , \forall\, i\,.\, k\,.\, \forall j = 2\,.\, \ldots\,.\, J_i \qquad (33)$$

$$cm_{ij} \ge pr_{kij}, V_{ijk} \qquad , \forall\, i\,.\, j = 1 \quad \forall k \in M_{ij} \qquad (34)$$

$$BS_{ijhgk} - D_{ijhgk} - pr_{khg}, O_{ijhgk} \ge 0 \qquad (35)$$

$$D_{ijhgk} - BS_{ijhgk} - pr_{khg}, O_{ijhgk} \ge 0 \qquad (36)$$

$$O_{ijhgk} \le V_{hgk} \qquad (37)$$

$$O_{ijhgk} \le V_{ijk} \qquad (38)$$

$$O_{ijhgk} \le Z_{ijhgk} \qquad (39)$$

$$O_{ijhgk} \ge V_{hgk} + V_{ijk} + Z_{ijhgk} - 2 \qquad , \forall\, i\,.\, h.\, j.\, g. \quad \forall k \in M_{ij} \cap M_{hg} \qquad (40)$$

$$D_{ijhgk} \le cm_{ij} \qquad (41)$$

$$D_{ijhgk} \le M, O_{ijhgk} \qquad (42)$$

$$D_{ijhgk} \ge cm_{ij} - M(1 - O_{ijhgk}) \qquad (43)$$

$$BS_{ijhgk} \le cm_{hg} \qquad (44)$$

$$BS_{ijhgk} \le M, O_{ijhgk} \qquad (45)$$

$$BS_{ijhgk} \geq cm_{hg} - M(1 - O_{ijhgk}) \tag{46}$$

$$\sum_{k \in M_{ij}} V_{ijk} = 1 \quad , \forall i.j \tag{47}$$

$$Z_{ijhgk} + Z_{hgijk} = F_{ijhgk} \tag{48}$$

$$F_{ijhgk} \leq V_{hgk} \quad , \forall i.h.j.g. \ \forall k \in M_{ij} \cap M_{hg} \tag{49}$$

$$F_{ijhgk} \leq V_{ijk} \tag{50}$$

$$F_{ijhgk} \geq V_{hgk} + V_{ijk} - 1 \tag{51}$$

$$VC_{ijk} - pr_{kij}, V_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \tag{52}$$

$$VC_{ijk} < (SM_{k.r} * V_{ijk}) + M * B_{ijkr} \tag{53}$$

$$VC_{ijk} - pr_{kij}, V_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \tag{54}$$

$$VC_{ijk} < (SM_{k.r+1} * V_{ijk}) + M(1 - B_{ijkr}) \tag{55}$$

$$VC_{ijk} - pr_{kij}, V_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \quad , \forall i.j.k.r \tag{56}$$

$$VC_{ijk} > (FM_{k.r} * V_{ijk}) - M(1 - B_{ijkr}) \tag{57}$$

$$VC_{ijk} \leq cm_{ij} \tag{58}$$

$$VC_{ijk} \leq M, V_{ijk} \tag{59}$$

$$VC_{ijk} \geq cm_{ij} - M(1 - V_{ijk}) \tag{60}$$

$$T_i \geq cm_{ij} - d_i \quad , \forall i.j = J_i \tag{61}$$

$$V_{ijk} \in \{0.1\} \quad , \forall i.j.k \tag{62}$$

$$B_{ijkr} \in \{0.1\} \quad , \forall i.j.k.r \tag{63}$$

$$T_i \geq 0 \quad , \forall i \tag{64}$$

$$cm_{ij} \geq 0 \quad , \forall i.j \tag{65}$$

$$O_{ijhgk} \in \{0.1\} \tag{66}$$

$$F_{ijhgk} \in \{0.1\} \tag{67}$$

$$D_{ijhgk} \geq 0 \quad , \forall i.h.j.g. \ \forall k \in M_{ij} \cap M_{hg} \tag{68}$$

$$BS_{ijhgk} \geq 0 \tag{69}$$

$$VC_{ijk} \geq 0 \tag{70}$$

## IV. IMPERIALIST COMPETITIVE ALGORITHM

In the previous section, the mathematical model was presented to obtain the optimal solution in GAMS software for small instances of the problem. However, as the size of the instances increase, the exact methods lose their effectiveness because of the high complication of the problem and in such cases, estimated methods such as heuristic or meta-heuristic algorithms have to be utilized. In this study, the ICA approach has been presented for the considered problem and its effectiveness in solving different instances has been evaluated. The ICA approach is a meta-heuristic algorithm which is based on population of solutions, and proposed by Atashpaz-Gargari and Lucas (2007). This algorithm is proposed to solve optimization problems and has been gradually developed by various researchers to solve scheduling problems. For example, Zandieh et al. (2017) developed an ameliorated ICA approach for F-JSP. Ahmadizar et al. (2019) developed an ICA approach for unrelated parallel machine scheduling problem.

The ICA approach starts with the initial population of solutions, each called a country. Some of the best countries are chosen as imperialists and the other countries are allotted to these imperialists as colonies. The whole power of an empire depends on the imperialist and its colonies. Each imperialist will gradually try to attract its colonies to itself, which will lead the search to the good areas of the solution space. Also, the occurrence of a revolution in a colony can cause changes in it, which can lead to the search for new areas of solution space. Over time, if a colony achieves a better position (according to the objective that the problem pursues it) than its imperialist, it will replace it. After the formation of the early empires, the imperialist competitive algorithm for the possession of each other's colonies begins amongst them; in each iteration of the algorithm, a competition is formed between the empires to seize the weakest colony of the weakest empire. Any empire that lose outs to enlarge its power will gradually lose its colonies during the competition and eventually will be eliminated. This process continues until all the empires drop and only one empire stands with control over the rest of the countries. Since reaching such a state can be very time-consuming, a top limit for the number of iterations of the proposed approach is also considered as a stop condition; if the number of iterations gets to a certain amount, the algorithm terminates. In the following section, the various components of the proposed algorithm are examined.

## *A. Solution representation*

Solution representation is the first and most important step in the development of meta-heuristic algorithms. Thus, for the proposed algorithm, an effective solution representation with an efficient and greedy decoding methodology is adopted to lessen the search space. The job-based encoding is used for this problem to represent a solution, i.e., a sequence of the execution order of the job on the machines. For an instance with n jobs, this presentation gives a sequence of n elements in which each job emerges exactly one time. Due to the no-wait constraint in the problem and the method of performing and sequencing operations, the first operation of a job starts when all subsequent operations belonging to that job continue without any interruption. For this reason, all operations of a job can be joined together and considered as an operation that is processed at specific intervals and on predetermined machines without any interruption. After processing one job, the next job is processed. Therefore, the length of the solution vector is tantamount to the sum of the total jobs (n), and the location of the jobs inside the solution vector is the order of their processing on the machines. For example, Fig.1 shows the solution for a problem with four jobs, where each job consists of some operations.

| | | | |
|---|---|---|---|
| 1 | 4 | 2 | 3 |

**Fig. 1. An encoded solution for a problem with four jobs**

To calculate the amount of the objective function of a solution, it must first be decoded. In the literature related to JSPs, various approaches to decoding have been proposed. In this research, the decryption algorithm developed by Brizuela et al. (2001) for problems with no-wait constraint has been used to decode the problem under study. The steps of the decryption algorithm are as follows:

Step 1: An idle times list is provided for each machine, and at the beginning of the schedule when no job is started, each machine is completely idle except periods that are assigned for maintenance activities.
Step 2: The operations of the first unprocessed job in the solution representation should be processed respectively.
Step 3: The list of machine idle times is updated.
Step 4: If the processing of all jobs is finished, the algorithm stops; otherwise, it returns to step 2.

Due to the objective that the problem pursues it, it is clear that according to a greedy approach, the jobs should be processed as soon as possible to minimize total tardiness. To better illustrate the decoding approach, consider a solution which is shown in Fig. 1 with 4 jobs and 3 stages. It is remarkable that processing each job may not require machines of all stages. The processing route for each job is shown in Fig.2, and jobs processing time on machines is presented in Table 2.

Machines also become unavailable once every 5-time units due to preventive maintenance and repair activities. The length of the unavailability period for each machine is two-time units. In the beginning, the list of machines' idle times is presented in Table 3. According to the encoded solution in Fig.1, at first, job 1 must be processed, therefore, job 1 is scheduled according to the timetable in Fig.3. After scheduling for job 1, the machines' idle times are updated, as shown in Table 4.

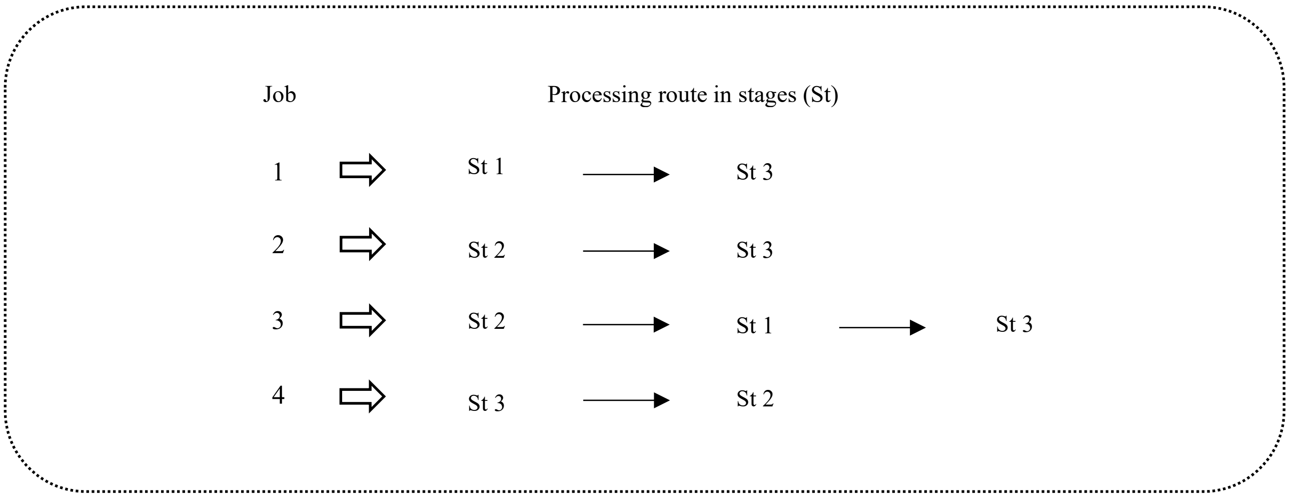

**Fig. 2. Jobs processing routes**

**Table 2. jobs processing times**

| Jobs | Stage 1 | Stage 2 | | Stage 3 |
|---|---|---|---|---|
| | $M_1$ | $M_2$ | $M_2'$ | $M_3$ |
| 1 | 3 | - | - | 2 |
| 2 | - | 1 | 1 | 2 |
| 3 | 3 | 1 | 2 | 1 |
| 4 | - | 3 | 2 | 3 |

**Table 3. Machines idle times list in the beginning**

| **Machine** | **Idle time** |
|---|---|
| $M_1$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_2$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M'_2$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_3$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |

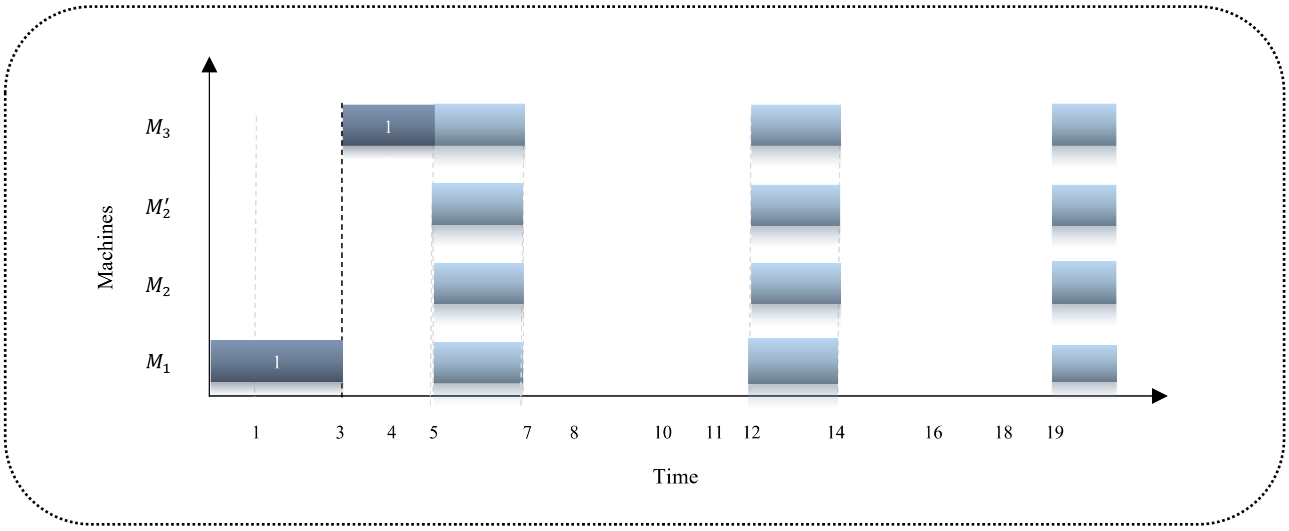

**Fig. 3. Timetable for job 1 according to presented encoded solution**

**Table 4. Machines idle times list after scheduling job 1**

| **Machine** | **Idle time** |
|---|---|
| $M_1$ | $[3,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_2$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M'_2$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_3$ | $[0,3] \cup [7,12] \cup [14,19] + \dots$ |

After updating the idle times' list, it should be examined when the processing of job 4 should start so that all its operations are able to be processed in a row on various required machines without interruption. Job 4 is scheduled according to the timetable in Fig.4. After scheduling for jobs 1 and 4, the machines' idle times are updated, as shown in Table 5.

After updating the idle times' list, it should be examined when the processing of job 2 should start so that all its operations are able to be processed in a row on various required machines without interruption. Job 2 is scheduled according to the timetable in Fig.5. After scheduling for jobs 1, 4 and 2, the machines' idle times are updated, as shown in Table 6.

After updating the idle times' list, it should be examined when the processing of job 3 should start so that all its operations are able to be processed in a row on various required machines without interruption. Job 3 is scheduled according to the timetable in Fig. 6.

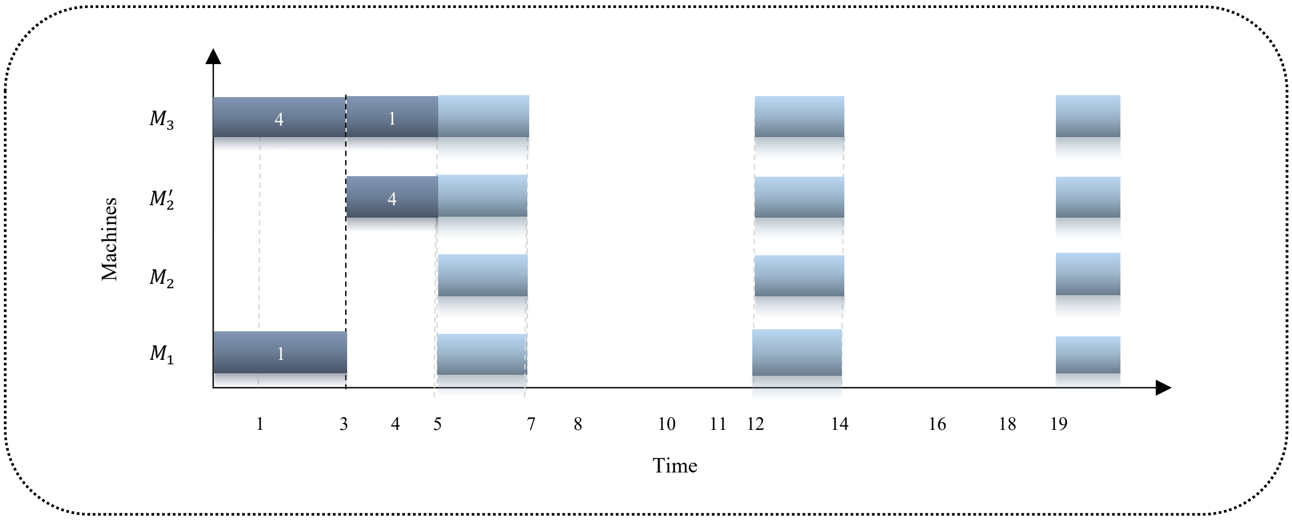

**Fig. 4. Timetable for job 1 and 4 according to presented encoded solution**

**Table 5. Machines idle times list after scheduling jobs 1 and 4**

| Machine | Idle time |
|---|---|
| $M_1$ | $[3,5] \cup [7,12] \cup [14,19] + \ldots$ |
| $M_2$ | $[0,5] \cup [7,12] \cup [14,19] + \ldots$ |
| $M_2'$ | $[0,3] \cup [7,12] \cup [14,19] + \ldots$ |
| $M_3$ | $[7,12] \cup [14,19] + \ldots$ |

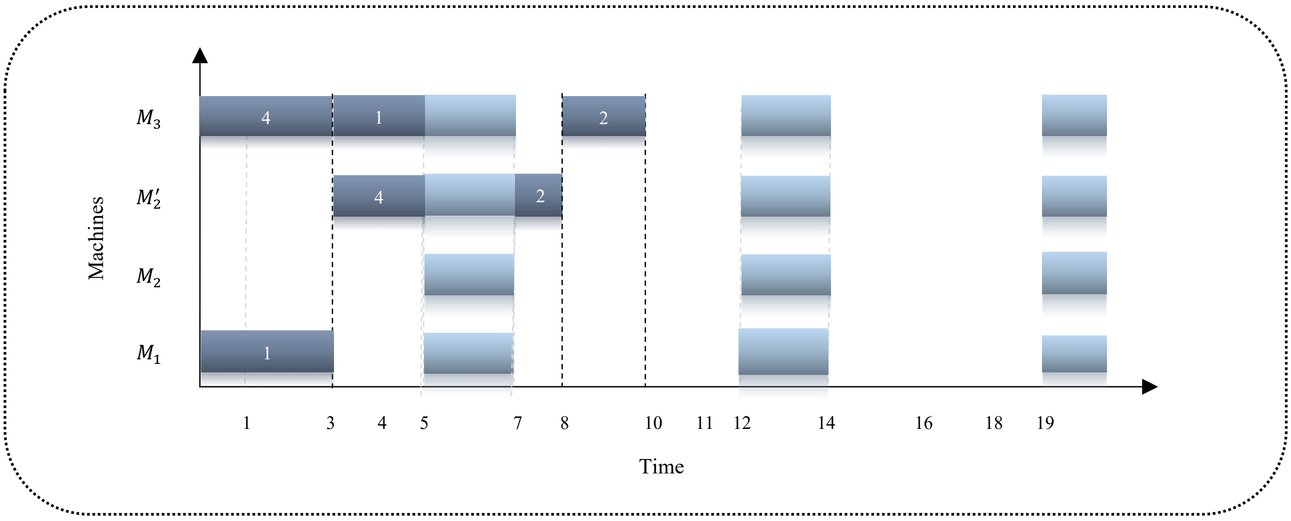

**Fig. 5. Timetable for jobs 1, 4 and 2 according to presented encoded solution**

**Tabe 6. Machines idle times list after scheduling jobs 1, 4 and 2**

| Machine | Idle time |
|---|---|
| $M_1$ | $[3,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_2$ | $[0,5] \cup [7,12] \cup [14,19] + \dots$ |
| $M_2'$ | $[0,3] \cup [8,12] \cup [14,19] + \dots$ |
| $M_3$ | $[7,8] \cup [10,12]\ [14,19] + \dots$ |

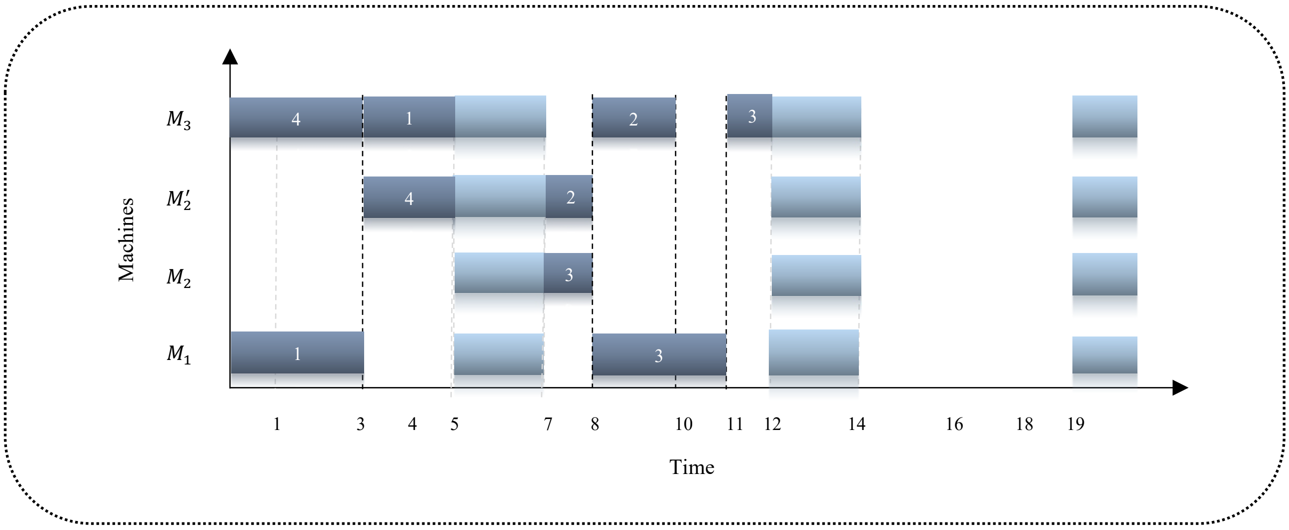

**Fig. 6. Timetable for all jobs according to presented encoded solution**

As shown in Fig.6, due to the no-wait constraint and the machines availability constraint, job 3 cannot start earlier than time unit 7. As the processing of all jobs is completed, the decoding algorithm is stopped and the timetable in Fig.6 is selected as the final timetable of the encrypted solution in Fig.1.

## *B. Formation of initial Empires*

At first, the initial population has been produced then, the $N_{imp}$ number of the best members is selected as the imperialist. To assign the rest of the members to the imperialists, the normalized amount of each imperialist's objective function is first calculated by equation 71:

$$\acute{f}(\mathrm{imp}) = f_{\max} - f(\mathrm{imp}) \tag{71}$$

Where f (imp) is the amount of the imperialist objective function for imp and $f_{\max}$ is the maximum amount of the objective function among the imperialists. Note that normalization is done because minimizing the problem's objective function is intended; Now, minimizing f (imp) is equivalent to maximizing $\acute{f}$ (imp). Then, the comparative power of each imperialist is calculated and the colonized countries are distributed among the imperialists based on equation 72.

$$\mathrm{pw}(\mathrm{imp}) = \acute{f}(\mathrm{imp}) \Big/ \sum_{\tau=1}^{N_{imp}} \acute{f}(\tau) \tag{72}$$

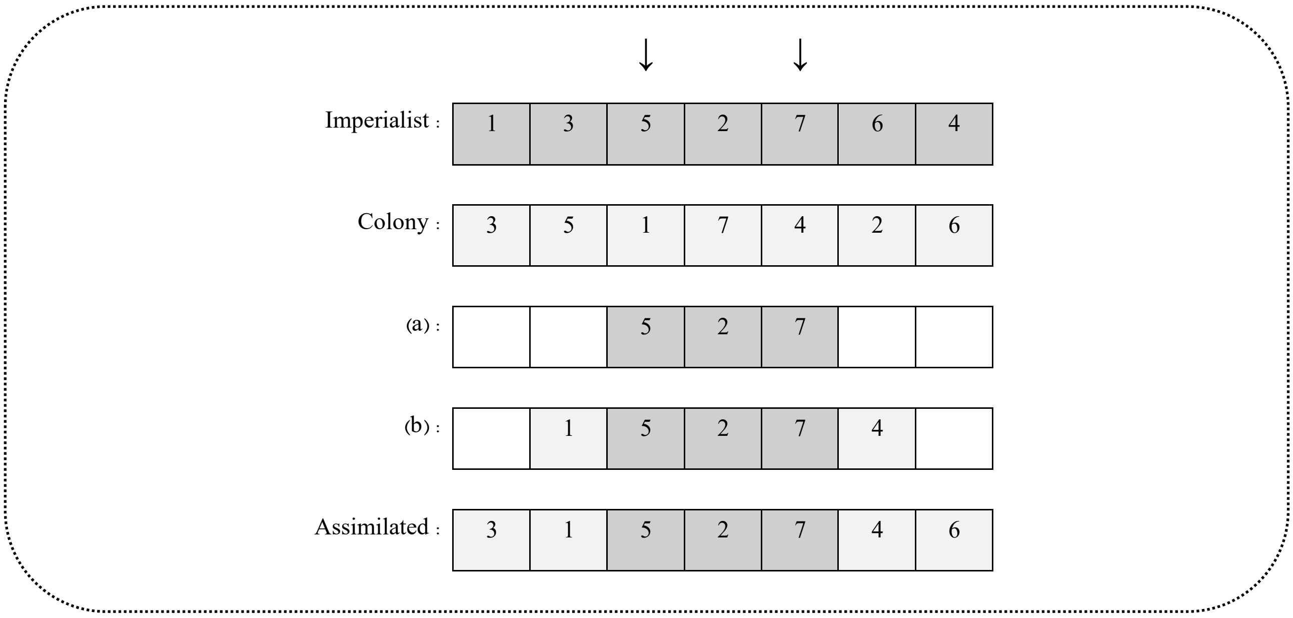

**Fig. 7. Assimilation process**

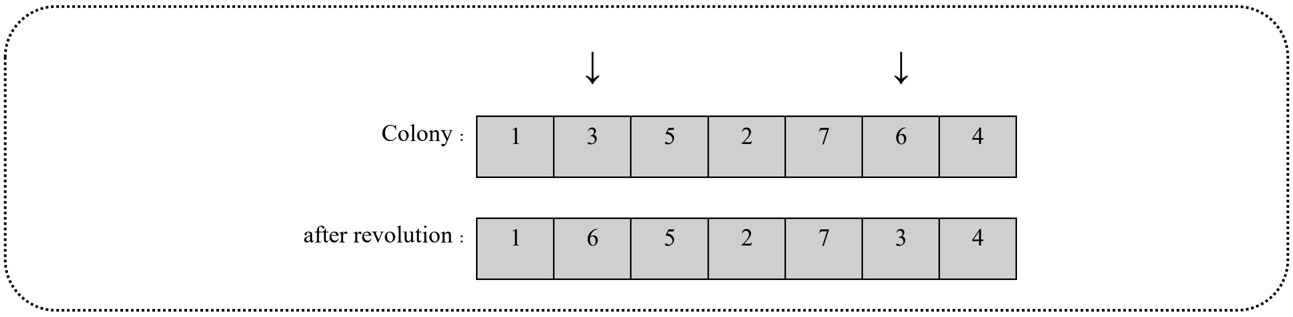

**Fig. 8. Revolution process**

## *C. Assimilation operator*

According to what is called the assimilation policy, each imperialist tends its colonies towards it from different social and political dimensions to gradually cause their evolution. In the developed ICA, an operator is used to execute the assimilation policy. To illustrate this operator, consider an example with 7 jobs. First, two cells are randomly selected and the numbers between them are copied from the imperialist to the assimilated colony (Fig.7 (a)). Then, the numbers that existed in the initial colony between the two selected cells and were not copied, are copied into the assimilated colony after matching the numbers that occupied their place. In other words, the number 1 is copied in place of number 5 and the number 4 is copied in place of number 2. Finally, the remaining numbers of the assimilated colony are allocated like the order of the initial colony (Fig.7 (b)).

## *D. Revolution operator*

In some cases, a social-political revolution can suddenly change the characteristics of a country. In the proposed ICA approach, the revolution is modeled by moving a colony to a new random circumstances, and due to the diversity of the search path, prevents it from dropping into the trap of local optimal. In the proposed meta-heuristic, after the assimilation process, a revolution occurs in each colony with a probability of $\mathrm{P_{rev}}$ (which is called the revolution rate). The operator of the revolution is that two cells are randomly chosen and their jobs are moved together (Fig.8).

### *E. Imperialist competitive*

Empires compete for possession of each other's colonies and aim to increase their power. Imperialist competition gradually enhances the strength of stronger empires and diminishes the power of weaker empires. If an empire fails to boost its power, it will eventually be eliminated from the competition. To compete in each iteration of the algorithm, the total power of each empire in relation to the power of the imperialist and its colonies is computed as follows:

$$\mathrm{Tf(imp)} = \mathrm{f(imp)} + \alpha \mathrm{f}_{\mathrm{avg}}^{\mathrm{col}}\mathrm{(imp)} \tag{73}$$

Where $\mathrm{f}_{\mathrm{avg}}^{\mathrm{col}}\mathrm{(imp)}$ is the average objective function value of the colonies in the imp empire and α is also a number between zero and one. Since in most implementations, α equal to 0.05 has led to favorable results, the same value has been used in this study. The normalized value of the whole power of each empire is then computed by equation 74:

$$\mathrm{Tf'(imp)} = \mathrm{Tf}_{\mathrm{max}} - \mathrm{Tf(imp)} \tag{74}$$

Where $\mathrm{Tf}_{\mathrm{max}}$ is the whole power of the weakest empire. Finally, the probability of taking over the weakest colony of the weakest empire by each imp empire is computed by equation 75:

$$\mathrm{P}_{\mathrm{pos}}\mathrm{(imp)} = \mathrm{Tf'(imp)} \Big/ \sum_{\tau=1}^{\mathrm{N}_{\mathrm{imp}}} \mathrm{Tf'}(\tau) \tag{75}$$

Thus, the weakest colony of the weakest empire will not necessarily be seized by the strongest empire, but the stronger empires will compete with more chances. Fig.9 shows the flowchart of the ICA approach.

## V. COMPUTAIONAL RESULT

In this study, the mathematical model is solved with GAMS software and the CPLEX solver is used. In this software, the maximum solution time for each instance is 7200 seconds and if the optimal solution is obtained during this period, it is reported. On the other hand, the ICA approach and its components are coded in the Python programming language and performed all the computational experiments on a Laptop with an Intel RCoreTMi7-4600U CPU clocked at 2.10 GHz with 8GB of memory operating under the Windows 10 operating system. To adjust the parameters of the proposed ICA, different values were considered for each parameter, and then, by solving some numerical problems in the initial experiments, the appropriate values were selected as described in Table 7 for use in subsequent experiments.

**Table 7. Parameters of the ICA approach**

| Adjusted value | Parameter |
|---|---|
| 10 × n | Population size |
| 100 | Max Number of iterations |
| 0. 3 × n | Number of initial empires |
| 0.4 | Revolution rate |
| 0.05 | $\alpha$ |

In Table 7, n indicates the number of jobs that is used to determine the population size of the algorithm. In this research, the primary population is generated randomly and due to the random nature of the proposed algorithm, each of the generated problems has been solved 5 times by the ICA approach and the minimum, average and maximum values of the objective function for obtained solutions have been reported. Also, to facilitate the comparison of the results, the following comparative percentage deviation has been used to measure the value of the objective function of each solution (sol) for a given problem:

$$\mathrm{RPD} = \left(\frac{\mathrm{f(sol)} - \mathrm{f(sol_{best})}}{\mathrm{f(sol_{best})}}\right) \times 100 \tag{76}$$

$\mathrm{sol_{best}}$ is the best solution for that problem among the obtained solutions. The RPD indicator compares each solution with the best solution obtained for that problem and declares their discrepancy as a percentage; the lower the value of this indicator, the better the quality of the solution.

As shown in Table 8, the results obtained from solving different sizes of the problem are reported, in which the RPD values for the problems that the mathematical model was able to find the optimal solution are bolded. In all instances presented in Table 8, the processing of each job requires exactly one machine from each stage. In small instances (problems with a maximum of 7 jobs) Gams software can produce an optimal solution in a sensible time and ICA approach also shows high efficiency and provides the optimal solution every 5 times. In large instances, since the Gams software loses its efficiency, the ICA approach does very well and provides a suitable solution in a sensible time. In other words, in large instances of the problem, the Gams software cannot find the optimal solution, and present solution that has low quality in comparison with the solution provided by the ICA approach. This discrepancy in quality of the solutions becomes more visible in large instances which is completely obvious in the instance 22 , 23 and 24. The weakness of Gams software becomes more obvious when it cannot find a solution in cases that have more than 25 jobs. The solution obtained from the ICA approach is reliable because it provides the optimal solution in small instances every 5 times such as exact method. The remarkable point of the proposed ICA is the low deviation of its presented solutions for each instance and this confirms the convergence of the algorithm. In general, according to the results, it can be asserted that the ICA approach performs well and can gain good solutions in an acceptable time, especially for large instances of the problem.

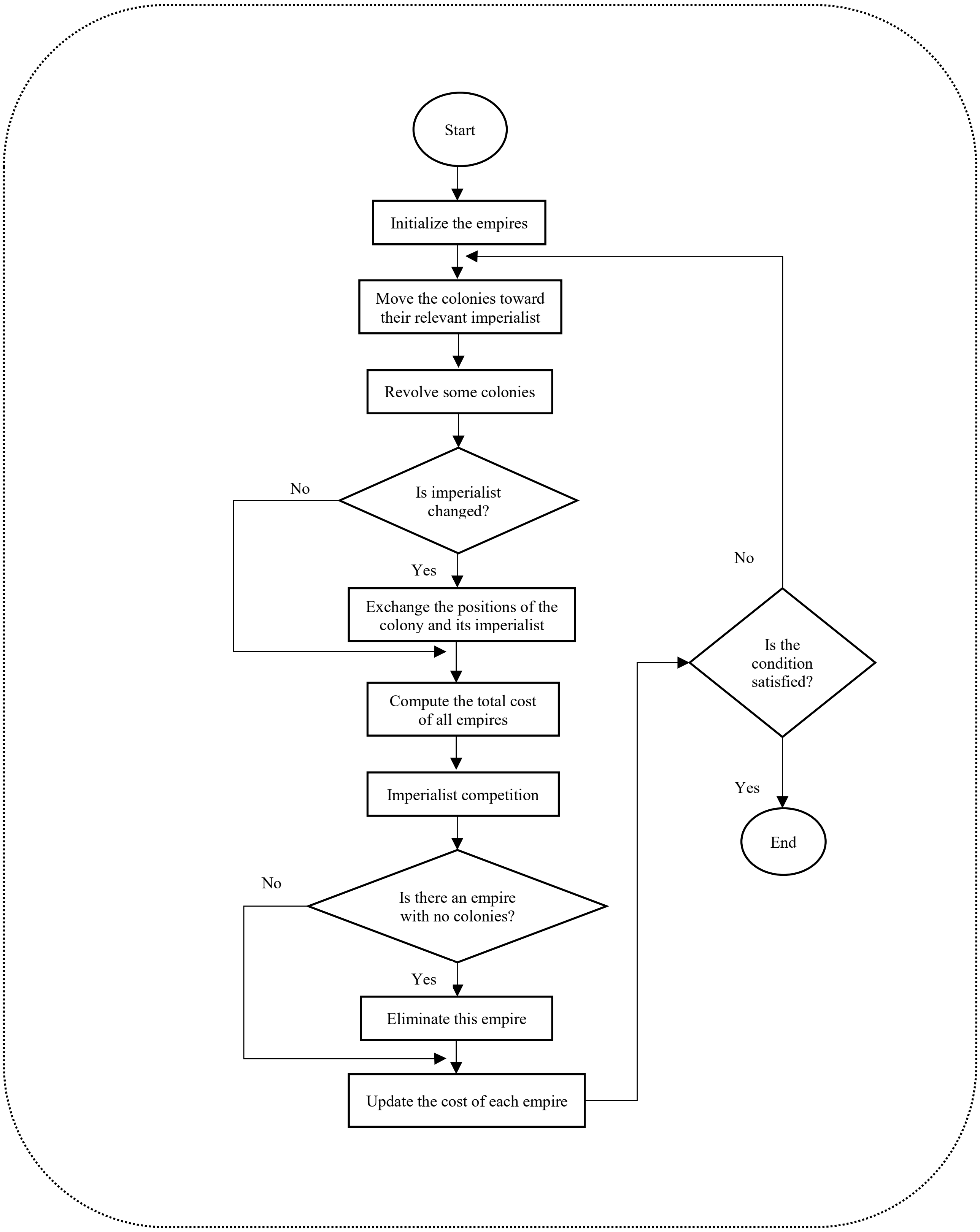
Start
Initialize the empires
Move the colonies toward their relevant imperialist
Revolve some colonies
Is imperialist changed?
No
Yes
Exchange the positions of the colony and its imperialist
Compute the total cost of all empires
Imperialist competition
Is there an empire with no colonies?
No
Yes
Eliminate this empire
Update the cost of each empire
Is the condition satisfied?
No
Yes
End

**Fig. 9. ICA approach flowchart**

**Table 8. Comparison outcomes of the mathematical model and ICA approach**

| Instance | No. job | No. stages | GAMS | | ICA | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Output (RPD) | Run time(s) | Output (RPD) | | | Run time (s) |
| | | | | | Min | Avg | Max | |
| 1 | 5 | 5 | **0** | 2231 | **0** | **0** | **0** | 61 |
| 2 | 5 | 7 | **0** | 2768 | **0** | **0** | **0** | 70 |
| 3 | 5 | 9 | **0** | 3952 | **0** | **0** | **0** | 68 |
| 4 | 6 | 5 | **0** | 4721 | **0** | **0** | **0** | 86 |
| 5 | 6 | 7 | **0** | 5147 | **0** | **0** | **0** | 79 |
| 6 | 6 | 9 | **0** | 5772 | **0** | **0** | **0** | 91 |
| 7 | 7 | 5 | **0** | 6692 | **0** | **0** | **0** | 104 |
| 8 | 7 | 7 | **0** | 7083 | **0** | **0** | **0** | 113 |
| 9 | 7 | 9 | 0 | 7200 | 0 | 0 | 0 | 129 |
| 10 | 8 | 5 | 0 | 7200 | 0 | 0 | 0 | 122 |
| 11 | 8 | 7 | 0.04 | 7200 | 0 | 0 | 0 | 149 |
| 12 | 8 | 9 | 0.07 | 7200 | 0 | 0.01 | 0.04 | 167 |
| 13 | 9 | 5 | 0.15 | 7200 | 0 | 0.02 | 0.04 | 160 |
| 14 | 9 | 7 | 0.23 | 7200 | 0 | 0.02 | 0.05 | 173 |
| 15 | 9 | 9 | 0.26 | 7200 | 0 | 0.05 | 0.08 | 209 |
| 16 | 10 | 5 | 0.37 | 7200 | 0 | 0.06 | 0.15 | 196 |
| 17 | 10 | 7 | 0.41 | 7200 | 0 | 0.05 | 0.10 | 245 |
| 18 | 10 | 9 | 0.38 | 7200 | 0 | 0.08 | 0.12 | 231 |
| 19 | 15 | 5 | 1.17 | 7200 | 0 | 0.36 | 0.52 | 416 |
| 20 | 15 | 7 | 1.74 | 7200 | 0 | 0.35 | 0.63 | 452 |
| 21 | 15 | 9 | 1.98 | 7200 | 0 | 0.51 | 0.84 | 478 |
| 22 | 20 | 5 | 3.25 | 7200 | 0 | 0.78 | 1.38 | 610 |
| 23 | 20 | 7 | 4.18 | 7200 | 0 | 0.94 | 1.65 | 592 |
| 24 | 20 | 9 | 4.71 | 7200 | 0 | 1.27 | 1.86 | 640 |
| 25 | 25 | 5 | - | 7200 | 0 | 1.54 | 1.93 | 795 |
| 26 | 25 | 7 | - | 7200 | 0 | 1.64 | 2.39 | 846 |
| 27 | 25 | 9 | - | 7200 | 0 | 1.47 | 2.28 | 817 |
| 28 | 30 | 5 | - | 7200 | 0 | 1.84 | 2.86 | 923 |
| 29 | 30 | 7 | - | 7200 | 0 | 1.54 | 2.53 | 992 |
| 30 | 30 | 9 | - | 7200 | 0 | 1.73 | 2.70 | 1031 |
| Average | | | 0.78 | 6558 | 0 | 0.47 | 0.74 | 368.1 |

## VI. CONCLUSION AND FUTURE RESEARCH

In this work, the NW-F-JSP with machines availability constraint for periodic maintenance activities and machines processing capability to minimize the sum of weighted tardiness has been formulated. Duo to its combinatorial complexity, the ICA approach is developed. The proposed ICA as well as GAMS software presented good performance in solving small instances. In addition, the ICA approach was able to solve large instances (which involve hundreds of operations) and show good performance according to quality and run time factors. The application of the considered problem is in the production of perishable products where delay during production can be very destructive and often production process is undertaken without any delay. In such production environments, it is common to consider periodic machines maintenance activities to prevent machine failure, and for this reason, these conditions have been considered in this research.

For future research, the possibility of waiting during production process due to machine failure can be considered by allowing interruption during operations processing time. In addition, due to the discrete solution space of the problem, heuristic algorithms based on neighborhood search can be used.

## REFERENCES

Ahmadian, M. M., Salehipour, A., & Cheng, T. C. E. (2020). A meta-heuristic to solve the just-in-time job-shop scheduling problem. *European Journal of Operational Research*.

Ahmadizar, F., Mahdavi, K., & Arkat, J. (2019). Unrelated parallel machine scheduling with processing constraints and sequence dependent setup times. *Advances in Industrial Engineering*, *53*(1), 495–507.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *2007 IEEE Congress on Evolutionary Computation*, 4661–4667. IEEE.

Benttaleb, M., Hnaien, F., & Yalaoui, F. (2018). Two-machine job shop problem under availability constraints on one machine: Makespan minimization. *Computers & Industrial Engineering*, *117*, 138–151.

Boyer, V., Vallikavungal, J., Rodríguez, X. C., & Salazar-Aguilar, M. A. (2021). The generalized flexible job shop scheduling problem. *Computers & Industrial Engineering*, *160*, 107542.

Brizuela, C. A., Zhao, Y., & Sannomiya, N. (2001). No-wait and blocking job-shops: Challenging problems for GA's. 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), 4, 2349–2354. IEEE.

Bürgy, R., & Bülbül, K. (2018). The job shop scheduling problem with convex costs. *European Journal of Operational Research*, *268*(1), 82–100.

Caldeira, R. H., & Gnanavelbabu, A. (2019). Solving the flexible job shop scheduling problem using an improved Jaya algorithm. *Computers & Industrial Engineering*, *137*, 106064.

Dai, M., Tang, D., Giret, A., & Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, *59*, 143–157.

Defersha, F. M., & Rooyani, D. (2020). An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time. *Computers & Industrial Engineering*, *147*, 106605.

El Khoukhi, F., Boukachour, J., & Alaoui, A. E. H. (2017). The "Dual-Ants Colony": A novel hybrid approach for the flexible job shop scheduling problem with preventive maintenance. *Computers & Industrial Engineering*, *106*, 236–255.

Fan, H., & Su, R. (2022). Mathematical Modelling and Heuristic Approaches to Job-shop Scheduling Problem with Conveyor-based Continuous Flow Transporters. *Computers & Operations Research*, *148*, 105998.

Fattahi, P., Messi Bidgoli, M., & Samouei, P. (2018). An improved Tabu search algorithm for job shop scheduling problem trough hybrid solution representations. *Journal of Quality Engineering and Production Optimization*, *3*(1), 13–26.

Gao, Jie, Gen, M., & Sun, L. (2006). Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, *17*(4), 493–507.

Gao, Jinsheng, Zhu, X., Bai, K., & Zhang, R. (2021). New controllable processing time scheduling with subcontracting strategy for no-wait job shop problem. *International Journal of Production Research*, 1–21.

García-León, A. A., Dauzère-Pérès, S., & Mati, Y. (2019). An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. *Computers & Operations Research*, *108*, 187–200.

Li, J., Deng, J., Li, C., Han, Y., Tian, J., Zhang, B., & Wang, C. (2020). An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowledge-Based Systems*, 106032.

Li, X., Yang, X., Zhao, Y., Teng, Y., & Dong, Y. (2020). Metaheuristic for Solving Multi-Objective Job Shop Scheduling Problem in a Robotic Cell. *IEEE Access*, *8*, 147015–147028.

Lu, C., Li, X., Gao, L., Liao, W., & Yi, J. (2017). An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. *Computers & Industrial Engineering*, *104*, 156–174.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*(2), 219–223.

Miyata, H. H., Nagano, M. S., & Gupta, J. N. D. (2019). Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. *Computers & Industrial Engineering*, *135*, 79–104.

Nohair, L., El Adraoui, A., & Namir, A. (2022). Solving Non-Delay Job-Shop Scheduling Problems by a new matrix heuristic. *Procedia Computer Science*, *198*, 410–416.

Ozolins, A. (2020). A new exact algorithm for no-wait job shop problem to minimize makespan. *Operational Research*, *20*(4), 2333–2363.

Samarghandi, H. (2019). Solving the no-wait job shop scheduling problem with due date constraints: A problem transformation approach. *Computers & Industrial Engineering*, *136*, 635–662.

Samarghandi, H., & Firouzi Jahantigh, F. (2019). Comparing Mixed-Integer and Constraint Programming for the No-Wait Flow Shop Problem with Due Date Constraints. *Journal of Quality Engineering and Production Optimization*, *4*(1), 17–24.

Shen, L., Dauzère-Pérès, S., & Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, *265*(2), 503–516.

Tamssaouet, K., Dauzère-Pérès, S., & Yugma, C. (2018). Metaheuristics for the job-shop scheduling problem with machine availability constraints. *Computers & Industrial Engineering*, *125*, 1–8.

Valenzuela-Alcaraz, V. M., Cosío-León, M. A., Romero-Ocaño, A. D., & Brizuela, C. A. (2022). A cooperative coevolutionary algorithm approach to the no-wait job shop scheduling problem. *Expert Systems with Applications*, *194*, 116498.

Weng, W., Chen, J., Zheng, M., & Fujimura, S. (2022). Realtime scheduling heuristics for just-in-time production in large-scale flexible job shops. *Journal of Manufacturing Systems*, *63*, 64–77.

Winklehner, P., & Hauder, V. A. (2022). Flexible job-shop scheduling with release dates, deadlines and sequence dependent setup times: a real-world case. *Procedia Computer Science*, *200*, 1654–1663.

Wu, X., Shen, X., & Li, C. (2019). The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously. *Computers & Industrial Engineering*, *135*, 1004–1024.

Yazdani, M., Aleti, A., Khalili, S. M., & Jolai, F. (2017). Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. *Computers & Industrial Engineering*, *107*, 12–24.

Ying, K.-C., & Lin, S.-W. (2020). Solving no-wait job-shop scheduling problems using a multi-start simulated annealing with bi-directional shift timetabling algorithm. *Computers & Industrial Engineering*, *146*, 106615.

Zandieh, M., Khatami, A. R., & Rahmati, S. H. A. (2017). Flexible job shop scheduling under condition-based maintenance: Improved version of Imperialist competitive algorithm. *Applied Soft Computing*, *58*, 449–464.

Zhang, G., Hu, Y., Sun, J., & Zhang, W. (2020). An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation*, *54*, 100664.

Zhang, G., Sun, J., Lu, X., & Zhang, H. (2020). An improved memetic algorithm for the flexible job shop scheduling problem with transportation times. *Measurement and Control*, 0020294020948094.

Zhang, S. J., Gu, X. S., & Zhou, F. N. (2020). An improved discrete migrating birds optimization algorithm for the no-wait flow shop scheduling problem. *IEEE Access*.

Zhu, N., Zhao, F., Wang, L., Ding, R., & Xu, T. (2022). A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows. *Expert Systems with Applications*, *198*, 116921.

Zhu, Z., & Zhou, X. (2020a). An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints. *Computers & Industrial Engineering*, *140*, 106280.

Zhu, Z., & Zhou, X. (2020b). Flexible job-shop scheduling problem with job precedence constraints and interval grey processing time. *Computers & Industrial Engineering*, 106781.