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

Kasra Mahdavi[1], Mohammad Mohammadi[2*], Fardin Ahmadizar[3]

[1]*Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
[2]*Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
[3]*Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran*

[*]Corresponding author: Mohammad Mohammadi (Email: Mohammadi@khu.ac.ir)

***Abstract*— Flexible job-shop scheduling problem (FJSP) is an extension of job shop scheduling problem which allows an operation to be performed by any machine amongst a set of available machines in each stage. This paper addresses a no-wait flexible job shop scheduling problem with machines availability constraints for maintenance activities and machines processing capability to minimize total weighted tardiness. The study is organized in two steps. In the first step, a new nonlinear mathematical model is developed for the considered problem, and then it is converted into a linear mathematical model using the techniques in the literature. Since the structure of the problem is NP-hard, thus in the second step, an Imperialist competitive algorithm is proposed to solve real-size 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 evaluate the performance of the developed algorithm. It is concluded that in small instances, solving the mathematical model by GAMS leads to the optimal solution, but with the increased size of instances, this method loses its efficiency and ICA performs better under these conditions.**

***Keywords*—** flexible job shop, no-wait, maintenance activities, Imperialist competitive algorithm.

I. INTRODUCTION

Job shop scheduling is a type of scheduling problem that is used in various production environments. The job shop scheduling problem was first raised by Manne (1960). In this study, it is proved that job shop scheduling, known as the NP-hard optimization problem in production scheduling literature, is highly complex. To the best of our knowledge, no methodologies in the literature are reported to be able to solve large instances in real-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

Manuscript Received day-Month-year and revised day-Month-year *ISSN: ****-****
Accepted on day- month-year

the perishable products industry. Delay 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. 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 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 with processing constraints is investigated. This scheduling problem is of special importance both from a theoretical and practical point of view. From theoretical point, considering the machines capability and machines maintenance activities in no-wait FJSSP show, a number of feasible solutions, albeit limited, are available. And in situations where there are such constraints in a flexible job shop scheduling problem, such solutions can be applied. From 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 job shop scheduling. 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, 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.

As far as we are aware, this is the first study on the flexible job shop scheduling problem (FJSSP) 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 precedence variable is established for the no-wait flexible job shop scheduling problem with processing constraints.
- The proposed model is linearized by techniques in the literature to be solved by linear solvers.
- An Imperialist competitive algorithm (ICA) is customized to solve the no-wait flexible job shop scheduling problem.

The rest of the paper is organized as follows: In Section II, the latest and related works are presented. Section III describes the problem and presents the mathematical model and its linearization process. In Section IV, an Imperialist competitive algorithm is proposed to solve real-size 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 job shop scheduling problem with various constraints. In this section, the newest related works are reviewed. El Khoukhi et al. (2017) investigated a flexible job shop scheduling problem with machines availability constraints to minimize the makespan. They proposed a mathematical model for this problem and due to its complexity, they developed a new optimization algorithm based on the ant nest algorithm. Zandieh et al. (2017) studied the flexible job shop scheduling problem with machines availability constraints to minimize makespan and proposed an improved Imperialist competitive algorithm for real-size instances. Yazdani et al. (2017) studied a job shop scheduling problem to minimize the sum of maximum tardiness and maximum earliness. They developed a mathematical model and a new optimization approach based on the Imperialist competitive algorithm. Lu et al. (2017) studied a multi objective flexible job shop scheduling problem with controllable processing times. In this study, makespan and minimizing the sum of consuming resources are considered to be objective functions. They developed a new multi-objective meta-heuristic algorithm called MODVOA. Benttaleb et al. (2018) studied job shop scheduling problem with two machines where one of the machines is out of reach in a certain period. In this study, the objective function is minimizing the longest completion time. They investigated the optimality of Jackson's algorithm and designed a heuristic algorithm using Jackson's law. Then, they proposed a branch and bound algorithm for the problem. Fattahi et al (2018) proposed a new cyclic algorithm based on Tabu search to improve the exploration and exploitation powers of some solution encoding that are suggested in the literature. In this research by solving several instances, the effectiveness of the proposed solution representation is shown. Bürgy and Bülbül (2018) studied a job shop scheduling problem 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. Shen et al. (2018) proposed a mathematical model for a flexible job shop scheduling problem with sequence-dependent setup times to minimize the makespan. They also developed a Tabu search algorithm based on new neighborhood search functions and various structures. Tamssaouet et al. (2018) studied a job shop scheduling problem with the objective function of minimizing the makespan in which machines are not always available and become unavailable at intervals. They developed simulated annealing and Tabu search algorithms with neighborhood functions for real-size instances of the problem. García-León et al. (2019) proposed a local search approach for the multi-objective flexible job shop scheduling problem to obtain Pareto solutions for any combination of regular functions. Caldeira et al. (2019) developed an improved Jaya algorithm for a flexible job shop scheduling problem with the objective function of minimizing the makespan. In this problem, they considered machines' setup time and transfer time between machines. Dai et al. (2019) studied a multi-objective flexible job shop scheduling problem with energy consumption and transportation constraints. In this problem, the objective function is minimizing the makespan and energy consumption. They developed an improved multi-objective Genetic algorithm

for this problem. Shen et al. (2019) investigated a flexible job shop scheduling problem, in which the processing time of jobs are variable and depend on the start time of their processing. In this study, the objective function is minimizing the longest completion time and the amount of energy consumed by machines. For this problem, they presented a hybrid multi-objective algorithm called MOHPIOSA. Samarghandi (2019) studied a no-wait job shop scheduling problem with delivery deadline constraints and the objective function of minimizing the longest completion time. He turned the problem into another problem and presented a mathematical model for both of them. Then, he developed the genetic algorithm to solve large instances. Miyata et al. (2019) studied a no-wait flow shop scheduling problem with dependent sequenced setup times and machines preventive maintenance to minimize makespan. In this problem, they postulated new policy for preventive maintenance which its parameters are based on Weibull distribution. In this study, they developed constructive heuristics for proposed problem. Samarghandi and Jahantigh (2019) studied a no-wait flow shop scheduling problem with due dates constraints to minimize makespan. They proposed two mathematical models and applied a Constraint Programming Model.

Zhang et al. (2020) investigated a no-wait flow shop scheduling problem with the objective function of minimizing the longest completion time. They developed the Discrete Migratory Bird Optimization (MBO) algorithm to achieve high quality solutions. Ahmadian et al. (2020) studied a job shop scheduling problem in which delivery date is considered for each job. In this problem, any difference between the completion time of a job and its delivery date is considered as a penalty, and the objective function is to minimize the sum of earliness and tardiness. They developed a Variable Neighborhood Search (VNS) algorithm for this problem. Zhang et al. (2020) developed an improved Genetic algorithm for a multi-objective flexible job shop scheduling problem. In this study, they considered machines setup time and jobs transfer time between machines. Li et al. (2020) developed an improved Jaya algorithm for a flexible job shop scheduling problem, in which machine setup times and jobs transfer time between machines are considered. Ying and Lin (2020) studied a no-wait job shop scheduling problem to minimize the makespan. They developed MSA-BST algorithm, which is based on the simulated annealing algorithm. Zhu et al. (2020b) studied a flexible job shop scheduling problem with job precedence constraints in which the processing time of jobs is expressed as an interval and the objective function is to minimize the interval length that is obtained for makespan. They developed a new optimization algorithm called SLHO for real-size instances of the problem. Zhu et al. (2020a) developed an efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with job precedence constraints. In this problem, the objective function is to minimize the makespan and maximizing the workload of machines simultaneously. Zhang et al. (2020) proposed an improved memetic algorithm for the flexible job shop scheduling problem with transportation times to minimize the makespan. Li et al. (2020) studied a multi-objective job shop scheduling problem 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 makespan and the total earliness and tardiness simultaneously. They developed a TLBO algorithm for large instances of the problem. Defersha and Rooyani (2020) developed a two-stage Genetic algorithm for a flexible job shop scheduling problem with sequence-dependent setup times. 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 no-wait job shop scheduling problem to minimize makespan. In this study, a new exact algorithm is developed to solve benchmark instances within a reasonable time limit. GAO et al. (2021) studied a no-wait job shop scheduling problem with due date and subcontracting cost constraints. They proposed two mathematical models. Then, they developed an artificial bee colony algorithm based on a rolling timeline. Boyer et al. (2021) studied a flexible job shop scheduling problem with machine capacity, time lags, holding times, and sequence-dependent setup times. They proposed a mixed integer linear programming and a constraint programming (CP) models to represent the problem and developed a metaheuristic based on a Greedy Randomized Adaptive Search Procedure to solve real size instances of the problem. Valenzuela et al. (2022) studied a no-wait job shop scheduling problem to minimize makespan and proposed a cooperative coevolutionary algorithm to solve large instances. Weng et al. (2022) studied a flexible job shop scheduling requiring operations to be performed by either a worker or a machine and to perform a machine operation, two workers are needed. They modeled the scheduling problem and propose four methods that form a realtime scheduling and control system for JIT production. Fan and Su (2022) investigated a job shop scheduling problem with conveyor-based Continuous Flow Transporters to minimize makespan. In this study, the jobs are processed on the machines which are connected in series via the conveyor. They presented mathematical programming model of the problem to find exact solutions in small instances and developed simulated annealing algorithm with NGS scheme to solve larger instances. Zhu et al. (2022) studied a no-wait flow shop scheduling problem 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 is assigned to a factory to process its operations. To solve large instances they proposed a new approach known as discrete knowledge-guided learning fruit fly optimization algorithm. Nohair et al. (2022) studied a non-delay job shop scheduling problem with the objective of minimizing makespan. They developed a new matrix heuristic to generate non-delay schedules that is computationally fast, simple and easy to implement. Winklehner et al. (2022) investigated a flexible job shop scheduling problem with periodic machines maintenance activities and processing constraints as a real world problem. They developed a constraint programming approach to minimize the total completion times. Valenzuela et al. (2022) proposed cooperative coevolutionary algorithm approach for no-wait job shop scheduling problem to minimize the makespan. In table 1, all of the studies that is 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	Flexible job shop	Cmax		×				Pm	Precedence constraints	Based on ACO
2	Zandieh et al.	2017	Flexible job shop	Cmax		×				Qm		ICA
3	Yazdani et al.	2017	Job shop	$E_{max} + T_{max}$			×			-		Based on ICA
4	Lu et al.	2017	Flexible job shop	Cmax + Minimizing consuming resources						Pm	Control the processing time of jobs by allocating resources	MODVOA
5	Bentaleb et al.	2018	Job shop	Cmax		×						Algorithm based on Jackson's rule
6	Shen et al.	2018	Flexible job shop	Cmax					×	Pm		Tabu search
7	Tamssaou et et al.	2018	Job shop	Cmax		×				-		SA - TS
8	Burgy et al.	2018	Job shop	Minimizing convex costs						-	Considering the costs related to the start of operations	Tabu search
9	Fattahi et al.	2018	Job shop	Cmax						-		new cyclic algorithm based on TS
10	Miyata et al.	2019	Flow shop	Cmax	×	×			×	-	Flexible preventive maintenance	Constructive heuristics
11	Garcia Leon et al.	2019	Flexible job shop	A set of regular objective functions			×			Pm		Local search approach to obtain Pareto solutions
12	Caldeira et al.	2019	Flexible job shop	Cmax				×	×	Pm		Jaya algorithm
13	Shen et al.	2019	Flexible job shop	Minimizing energy consumption + Cmax			×			Rm	The jobs processing times are variable	MOHPIOSA
14	Dai et al.	2019	Flexible job shop	Minimizing energy consumption + Cmax			×	×		Pm		NSGA II

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	
15	Chen et al.	2020	Flow shop	Cmax	×	×				-	Scheduling with two machines	Problem is extremely NP-hard in the case of two machines
16	Zhang et al.	2020	Flow shop	Cmax	×					-		MBO
17	Deng et al.	2020	Job shop	Minimizing the sum of completion time	×					-		A heuristic algorithm called PBIG
18	Zhang et al.	2020	Flexible job shop	Cmax				×	×	Pm		GA
19	Li et al.	2020	Flexible job shop	Minimizing energy consumption + Cmax				×	×	Pm		Jaya algorithm
20	Ahmadian et al.	2020	Job shop	JIT						-	Independent due date for each job	VNS
21	Zhu et al.	2020	Flexible job shop	Maximizing machines workload + Cmax						Pm	Precedence constraints	Gray wolf algorithm
22	Defersha et al.	2020	Flexible job shop	Cmax		×	×		×	Pm	machines are available at different times	GA
23	Zhang et al.	2020	Flexible job shop	Cmax				×		Qm		Improved memetic algorithm
24	Ying et al.	2020	Job shop	Cmax	×					-		MSA-BST
25	Zhu et al.	2020	Flexible job shop	Minimizing the length of Cmax interval						Pm	Precedence constraints	New optimization algorithm called SLHO
26	Zhang and Sun	2020	Flexible job shop	Cmax				×		Qm		Improved GA
27	Li et al.	2020	Job shop	Cmax + JIT				Using robots		-	Job due date as a time window	New optimization algorithm called IMOTLBO
28	Ozolins	2020	Job shop	Cmax	×					-		New exact algorithm based on DP
29	Gao et al.	2021	Job shop	Cmax and subcontracting cost	×					-	Subcontracting strategy to satisfy the deadlines	ABC based on a rolling timeline

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	
30	Boyer et al.	2021	Flexible job shop	Cmax					×	Qm	With machine capacity and time lags	Metaheuristic based on a Greedy Randomized Adaptive Search Procedure
31	Valenzuela et al.	2022	Job shop	Cmax	×					-		Cooperative coevolutionary algorithm
32	Weng et al.	2022	Flexible job shop	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	Job shop	Cmax						-	The jobs are processed on the machines which are connected in series via the conveyor	Simulated Annealing algorithm with NGS scheme
34	Zhu et al.	2022	Flow shop	Weighted JIT	×					-	Considering time window for jobs due dates	A discrete algorithm
35	Nohair et al.	2022	Job shop	Cmax						-	Machines are never kept idle while job is waiting	New matrix heuristic
36	Winklehner et al.	2022	Flexible job shop	Minimizing the total completion times		×	×		×	Rm	Deadline for each job Release date for each job	Constraint programming approach
***	**Current study**		Flexible job shop	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, which covers flexible job shop scheduling problem 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 statement

In a flexible job shop scheduling, 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 allowed to be processed on any among a set of available machines. The other assumptions of the problem are as follows:

- All jobs and machines are available at time 0, each machine can only execute one operation at a given time.
- Each job has its operation sequence, which indicates its processing route.
- To process each job it may not require machines during all stages.
- No job can be processed on more than one machine simultaneously.
- No operation can be interrupted once started (preemption is not allowed).
- Waiting time between two consecutive operations of the same job is not allowed.
- Each job has an independent due date so if it is processed 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: index of jobs (1, …, n)

j: index of operations (1, … ,J_i)

k: index of machines (1, …, m)

r: index of unavailability interval

Parameters:

n: total number of jobs

m: total number of machines

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 unavailability constraints

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 (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 proposed first by Gao et al. (Jie Gao, Gen, & Sun, 2006) to formulate FJSSP and we have adopted it for our FJSSP. The objective function is as minimizing total weighted tardiness:

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

The following constraints enforce each job to follow a specified operation sequence and guarantee no-wait constraint:

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

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

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

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

The following constraints are disjunctive constraints:

$$(cm_{hg} - cm_{ij} - pr_{khg}).V_{hgk}.V_{ijk}.\ Z_{ijhgk} \geq 0 \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (5)$$

$$(cm_{ij} - cm_{hg} - pr_{kij}).V_{ijk}.V_{hgk}.\ Z_{hgijk} \geq 0 \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (6)$$

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}:

$$\begin{aligned} &O_{ijhgk} \leq V_{hgk} \\ &O_{ijhgk} \leq V_{ijk} \\ &O_{ijhgk} \leq\ Z_{ijhgk} \\ &O_{ijhgk} \geq V_{hgk} + V_{ijk} +\ Z_{ijhgk} - 2 \end{aligned} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (7)$$

Therefore, constraints 5 and 6 become as follows:

$$cm_{hg}.O_{ijhgk} - cm_{ij}.O_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{8}$$

$$cm_{ij}.O_{ijhgk} - cm_{hg}.O_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{9}$$

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:

$$\begin{aligned} &D_{ijhgk} \leq cm_{ij} \\ &D_{ijhgk} \leq M.O_{ijhgk} \\ &D_{ijhgk} \geq cm_{ij} - M(1 - O_{ijhgk}) \end{aligned} \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{10}$$

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

$$\begin{aligned} &BS_{ijhgk} \leq cm_{hg} \\ &BS_{ijhgk} \leq M.O_{ijhgk} \\ &BS_{ijhgk} \geq cm_{hg} - M(1 - O_{ijhgk}) \end{aligned} \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{11}$$

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

$$\begin{aligned} &BS_{ijhgk} - D_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \\ &D_{ijhgk} - BS_{ijhgk} - pr_{khg}.O_{ijhgk} \geq 0 \end{aligned} \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{12}$$

And constraints 7, 10 and 11

The following constraint states that one machine must be selected from a set of available machines for each operation:

$$\sum_{k \in M_{ij}} V_{ijk} = 1 \quad , \forall i, j \tag{13}$$

Constraint 14 enforces to be chosen one of two precedence relationships.

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

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

$$\begin{aligned} &F_{ijhgk} \leq V_{hgk} \\ &F_{ijhgk} \leq V_{ijk} \\ &F_{ijhgk} \geq V_{hgk} + V_{ijk} - 1 \end{aligned} \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{15}$$

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

$$Z_{ijhgk} + Z_{hgijk} = F_{ijhgk} \quad , \forall i, h, j, g, \ \forall k \in M_{ij} \cap M_{hg} \tag{16}$$

And constraints 15

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

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

$$cm_{ij}.V_{ijk} < (SM_{k,r} * V_{ijk}) + M * B_{ijkr} \quad , \forall i, j, k, r \tag{18}$$

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

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

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

$$cm_{ij}.V_{ijk} > \left(FM_{k,r} * V_{ijk}\right) - M\left(1 - B_{ijkr}\right) \quad , \forall i, j, k, r \quad (22)$$

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

$$\begin{aligned} &VC_{ijk} \le cm_{ij} \\ &VC_{ijk} \le M.V_{ijk} \quad , \forall i, j, k, r \quad (23) \\ &VC_{ijk} \ge cm_{ij} - M(1 - V_{ijk}) \end{aligned}$$

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

$$VC_{ijk} - pr_{kij}.V_{ijk} < \left(SM_{k,r} * V_{ijk}\right) + M * B_{ijkr} \quad , \forall i, j, k, r \quad (24)$$

$$VC_{ijk} < \left(SM_{k,r} * V_{ijk}\right) + M * B_{ijkr} \quad , \forall i, j, k, r \quad (25)$$

$$VC_{ijk} - pr_{kij}.V_{ijk} < \left(SM_{k,r+1} * V_{ijk}\right) + M\left(1 - B_{ijkr}\right) \quad , \forall i, j, k, r \quad (26)$$

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

$$VC_{ijk} - pr_{kij}.V_{ijk} > \left(FM_{k,r} * V_{ijk}\right) - M\left(1 - B_{ijkr}\right) \quad , \forall i, j, k, r \quad (28)$$

$$VC_{ijk} > \left(FM_{k,r} * V_{ijk}\right) - M\left(1 - B_{ijkr}\right) \quad , \forall i, j, k, r \quad (29)$$

And constraints 23

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

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

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

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

S. T.

$$cm_{ij} - cm_{ij-1} \ge pr_{kij}.V_{ijk} \quad , \forall i, k, \forall j = 2, \dots, J_i \quad (32)$$

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

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

$$BS_{ijhgk} - D_{ijhgk} - pr_{khg}.O_{ijhgk} \ge 0 \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (35)$$

$$D_{ijhgk} - BS_{ijhgk} - pr_{khg}.O_{ijhgk} \ge 0 \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (36)$$

$$O_{ijhgk} \le V_{hgk} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (37)$$

$$O_{ijhgk} \le V_{ijk} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (38)$$

$$O_{ijhgk} \le Z_{ijhgk} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (39)$$

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

$$D_{ijhgk} \le cm_{ij} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (41)$$

$$D_{ijhgk} \le M.O_{ijhgk} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (42)$$

$$D_{ijhgk} \ge cm_{ij} - M(1 - O_{ijhgk}) \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (43)$$

$$BS_{ijhgk} \le cm_{hg} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (44)$$

$$BS_{ijhgk} \le M.O_{ijhgk} \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (45)$$

$$BS_{ijhgk} \ge cm_{hg} - M(1 - O_{ijhgk}) \quad , \forall i, h, j, g, \ \ \forall k \in M_{ij} \cap M_{hg} \quad (46)$$

$$\sum_{k\in M_{ij}} V_{ijk} = 1 \quad , \forall i,j \qquad (47)$$

$$Z_{ijhgk} + Z_{hgijk} = F_{ijhgk} \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (48)$$

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

$$F_{ijhgk} \leq V_{ijk} \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (50)$$

$$F_{ijhgk} \geq V_{hgk} + V_{ijk} - 1 \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (51)$$

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

$$VC_{ijk} < (SM_{k,r} * V_{ijk}) + M * B_{ijkr} \quad , \forall i,j,k,r \qquad (53)$$

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

$$VC_{ijk} < (SM_{k,r+1} * V_{ijk}) + M(1 - B_{ijkr}) \quad , \forall i,j,k,r \qquad (55)$$

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

$$VC_{ijk} > (FM_{k,r} * V_{ijk}) - M(1 - B_{ijkr}) \quad , \forall i,j,k,r \qquad (57)$$

$$VC_{ijk} \leq cm_{ij} \quad , \forall i,j,k,r \qquad (58)$$

$$VC_{ijk} \leq M.V_{ijk} \quad , \forall i,j,k,r \qquad (59)$$

$$VC_{ijk} \geq cm_{ij} - M(1 - V_{ijk}) \quad , \forall i,j,k,r \qquad (60)$$

$$T_i \geq cm_{ij} - d_i \quad , \forall i, j = J_i \qquad (61)$$

$$V_{ijk} \in \{0,1\} \quad , \forall i,j,k \qquad (62)$$

$$B_{ijkr} \in \{0,1\} \quad , \forall i,j,k,r \qquad (63)$$

$$O_{ijhgk} \in \{0,1\} \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (64)$$

$$F_{ijhgk} \in \{0,1\} \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (65)$$

$$T_i \geq 0 \quad , \forall i \qquad (66)$$

$$cm_{ij} \geq 0 \quad , \forall i,j \qquad (67)$$

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

$$BS_{ijhgk} \geq 0 \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (69)$$

$$VC_{ijk} \geq 0 \quad , \forall i,h,j,g, \ \forall k \in M_{ij} \cap M_{hg} \qquad (70)$$

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 due to the high complexity of the problem and in such cases, we have to use approximate methods such as heuristic or metaheuristic algorithms. In this study, the Imperialist competitive algorithm has been presented for the considered problem and its effectiveness in solving different instances has been evaluated. The Imperialist competitive algorithm is one of the population-based metaheuristic algorithms proposed by Atashpaz-Gargari et al. (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 improved Imperialist competitive algorithm for flexible job shop scheduling problem. Ahmadizar et al. (2019) developed an Imperialist competitive algorithm for unrelated parallel machine scheduling problem.

The Imperialist competitive algorithm starts with the initial population of solutions, each called a country. Some of the best countries are selected as imperialists and the rest of the population is allocated to these imperialists as colonies. The total 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 function of the problem) than its imperialist, it will replace it. After the formation of the early empires, 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 fails to increase its power will gradually lose its colonies during the competition and eventually will be eliminated. This process continues until all the empires fall and only one empire remains with control over the rest of the countries. Since reaching such a state can be very time-consuming, an upper bound for the number of iterations of the proposed algorithm is also considered as a stop condition; if the number of iterations reaches a certain value, 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 metaheuristic algorithms. Thus, for the proposed algorithm, an effective solution representation with efficient and greedy decoding methodology is adopted in order to reduce the search space. We use the job-based encoding for this problem to represent a solution, i.e., a sequence of the execution order of the job on the machines. For a problem with n job, this representation gives a sequence of n elements in which each job appears 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 equal 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 value of the objective function of a solution, it must first be decoded. In the literature related to job shop scheduling problems, various approaches to decoding have been proposed. In this research, the decryption algorithm which is 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 function of the problem, 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 are 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 in Table 4.

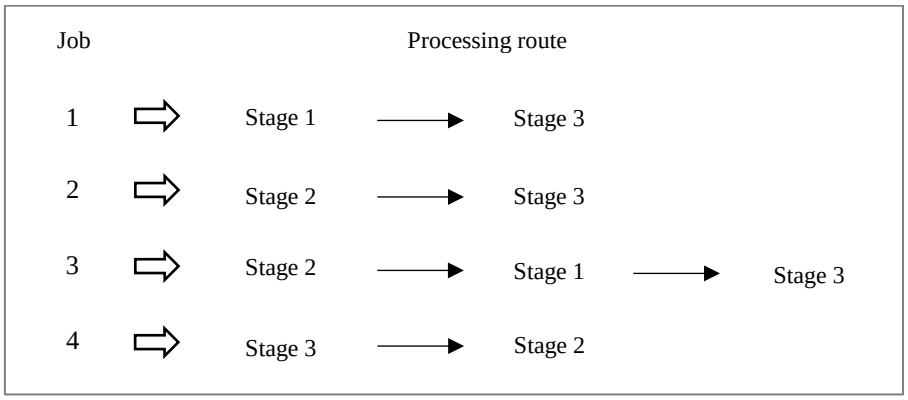

Fig. 2. Jobs processing routes

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	$[0,5] \cup [7,12] \cup [14,19] + \ldots$
M_2	$[0,5] \cup [7,12] \cup [14,19] + \ldots$
M_2'	$[0,5] \cup [7,12] \cup [14,19] + \ldots$
M_3	$[0,5] \cup [7,12] \cup [14,19] + \ldots$

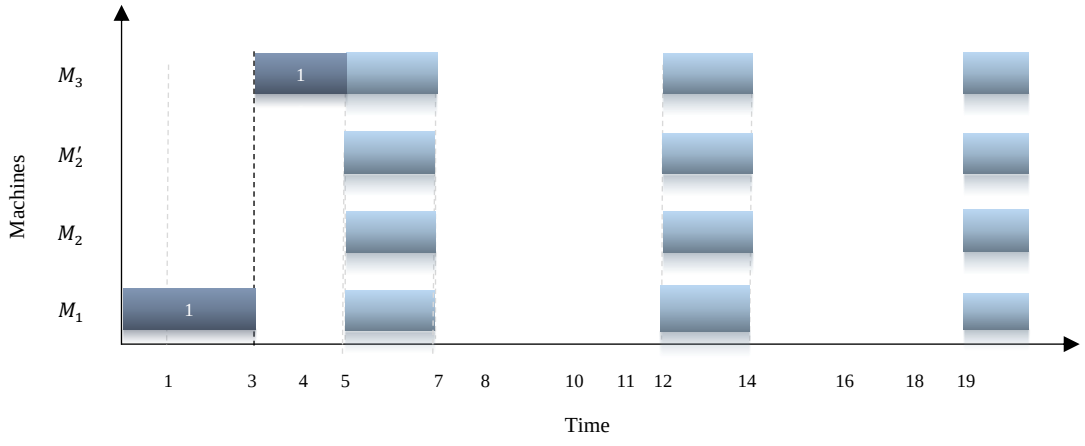

Fig. 3. Timetable for job 1 according to presented encoded solution

TABLE 4. MACHINES IDLE TIMES LIST AFTER SCHEDULING JOB 1

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

After updating the idle times' list, it should be examined when the processing of job 4 should start so that all its operations can be processed in a row on different 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 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 can be processed in a row on different 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 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 can be processed in a row on different required machines without interruption. Job 3 is scheduled according to the timetable in Fig.6.

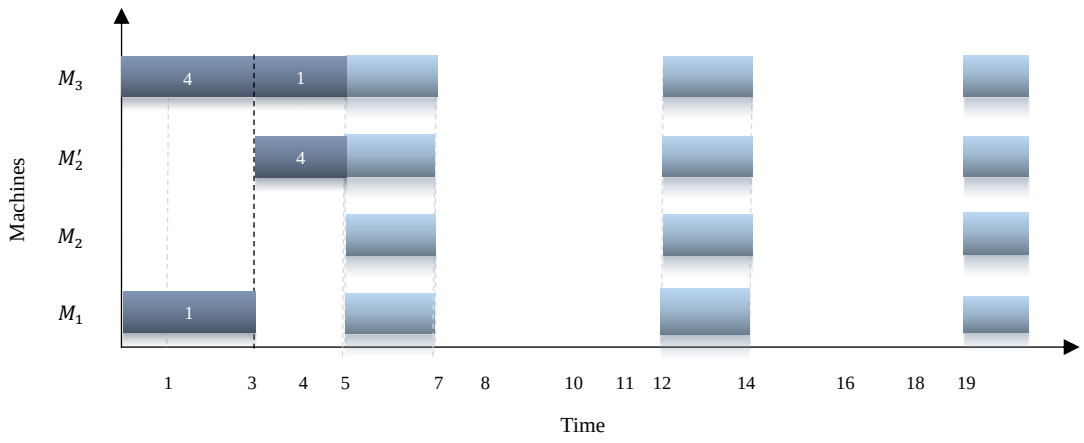

Fig. 4. Timetable for job 1 and 4 according to presented encoded solution

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	$[7,12] \cup [14,19] + \ldots$

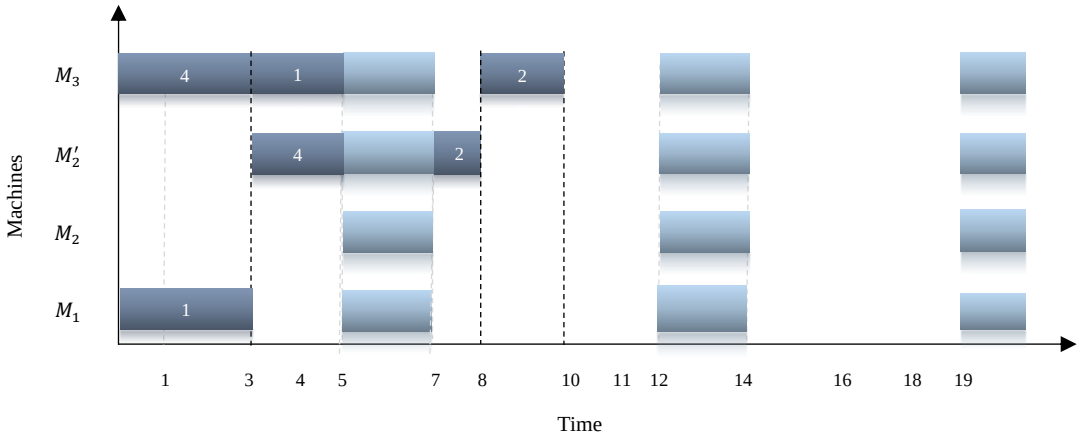

Fig. 5. Timetable for jobs 1, 4 and 2 according to presented encoded solution

TABLE 6. MACHINES IDLE TIMES LIST AFTER SCHEDULING JOBS 1, 4 AND 2

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

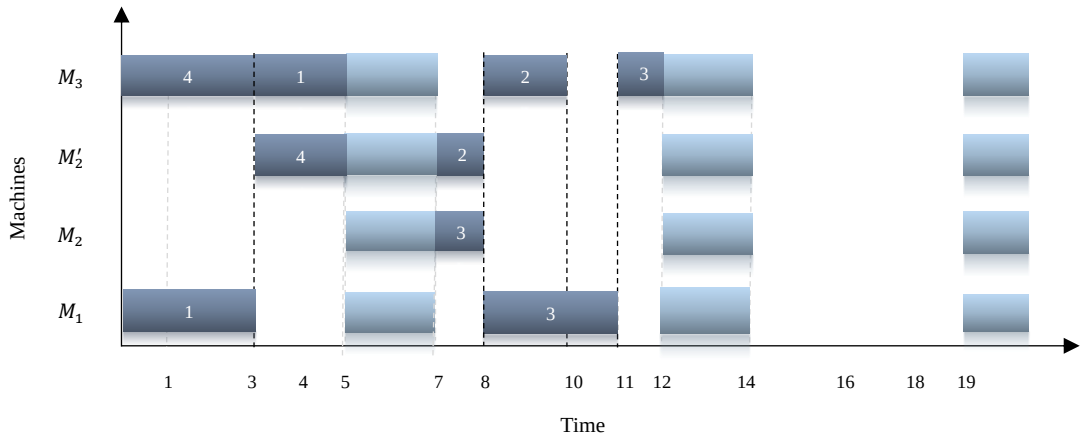

Fig. 6. Timetable for all jobs according to presented encoded solution

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 value of each imperialist's objective function is first calculated as follows:

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

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

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

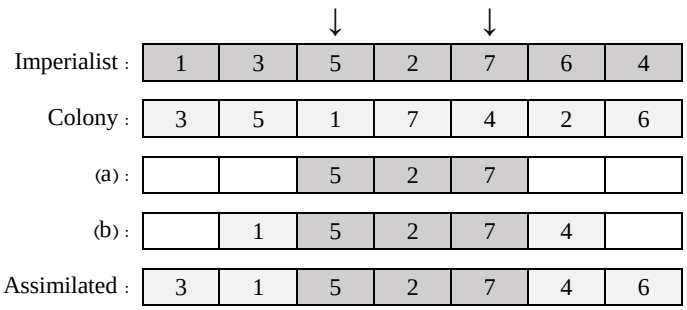

Fig. 7. Assimilation process

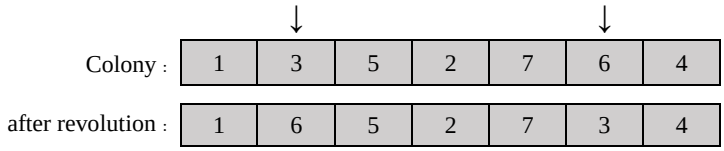

Fig. 8. Revolution process

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 proposed algorithm, 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 Imperialist competitive algorithm, the revolution is modeled by moving a colony to a new random position and due to the diversity of the search path, prevents it from falling into the trap of local optimal. In the proposed algorithm, after the assimilation process, a revolution occurs in each colony with a probability of P_{rev} (which is called the revolution rate). The operator of the revolution is that two cells are randomly selected and their jobs are moved together (Fig.8).

E. Imperialist competitive

Empires compete for possession of each other's colonies and increase their power. Imperialist competition gradually increases the power of stronger empires and decreases the power of weaker empires. If an empire fails to increase 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 proportion to the power of the imperialist and its colonies is calculated as follows:

$$Tf(imp) = f(imp) + \alpha f_{avg}^{col}(imp) \quad (73)$$

Where $f_{avg}^{col}(imp)$ 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 total power of each empire is then calculated as follows:

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

Where $\mathrm{Tf}_{\max}$ is the total power of the weakest empire. Finally, the probability of taking over the weakest colony of the weakest empire by each imp empire is calculated as follows:

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

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 Imperialist competitive algorithm.

V. COMPUTATIONAL 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 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 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

ICA	
Adjusted value	Parameter
$10 \times n$	Population size
100	Max Number of iterations
$0.3 \times n$	Number of initial empires
0.4	Revolution rate
0.05	α

In the Table 7, n indicates the number of jobs that is used to determine the population size of the algorithm. In this research, the initial 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 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 relative percentage deviation has been used to measure the value of the objective function of each solution (sol) for a given problem:

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

$\mathrm{sol_{best}}$ 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 difference 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 provide an optimal solution in a reasonable time and ICA also shows high efficiency and provides the optimal solution every 5 times. In large instances, since the Gams software loses its efficiency, the ICA does very well and provides a suitable solution in a reasonable 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. This difference 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 is reliable because it provided the optimal solution in small instances every 5 times such as exact method. The remarkable point of 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 claimed that the proposed ICA performs well and can obtain good solutions in an acceptable time, especially for large instances of the problem.

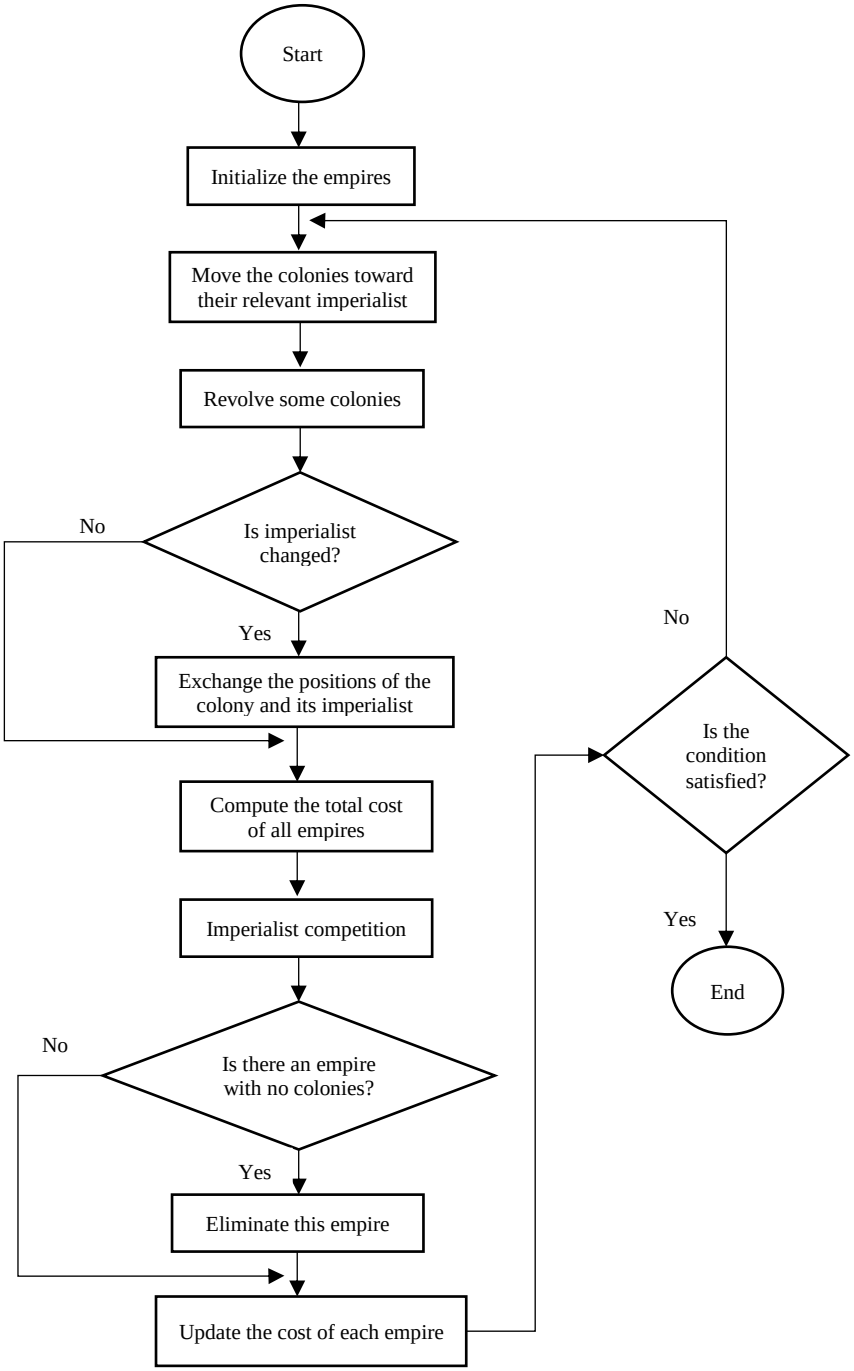

Fig. 9. Imperialist competitive algorithm flowchart

TABLE 8. COMPARISON RESULTS OF THE MATHEMATICAL MODEL AND ICA

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 no-wait flexible job shop scheduling problem 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 Imperialist competitive algorithm (ICA) is developed. The ICA as well as GAMS software presented good performance in solving small instances. In addition, the proposed ICA 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.