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.2024.16744.1244

Optimizing Open Shop Scheduling: Minimizing Makespan through Whale Optimization Algorithm and Transportation Time Consideration

Morteza Enayati [1], Mahdi Yousefi Nejad Attari [2*], Fahime Lotfian Delouyi [3]

[1] *Department of Industrial Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran*

[2] *Department of Industrial Engineering, Bonab Branch, Islamic Azad University, Bonab, Iran*

[3] *Department of Mechanical Engineering, Faculty of Engineering, University of Zabol, Zabol*

**** Corresponding Author:** Mahdi Yousefi Nejad Attari (Email: mahdi108108@gmail.com)*

Abstract – This paper addresses the open shop scheduling problem, considering parallel machines within each stage and integrating job transportation times between stages, independent of job specifics. In this scheduling problem, all jobs traverse each stage, and once a job commences on a machine, it must complete without machine breakdowns. To meet this challenge, a mixed-integer linear programming (MILP) model is introduced to minimize the makespan, which represents the maximum job completion time. Given the NP-hard nature of the open-shop scheduling problem, this study employs the whale metaheuristic algorithm to solve instances across various dimensions, spanning small, medium, and large scales. The algorithm parameters are systematically optimized using the Taguchi Method. Results from comparing the whale algorithm with the linear model implemented in GAMS highlight its exceptional efficiency in handling randomly generated small and medium-sized instances. Moreover, in a comparative analysis with other algorithms such as PSO and DE, the whale algorithm not only competes effectively but, in some instances, outperforms its counterparts. This observation underscores the algorithm's prowess in maintaining efficiency and high performance, particularly when addressing large-scale open-shop scheduling challenges. It excels in achieving a delicate balance between exploration and exploitation, thereby avoiding local optimal solutions.

***Keywords**– Open shop-scheduling, Parallel machines, Transportation time, Mixed-integer linear programming, Whale optimization algorithm.*

I. INTRODUCTION

In today's realm of operations research, scheduling has emerged as a paramount concern. Scheduling, the process of systematically arranging incoming requests (tasks) to ensure the efficient utilization of available resources (Gawali & Shinde, 2018), holds pivotal significance.

A scheduling program embodies a timetable that coordinates jobs and machines in a manner that specifies the assignment of each job to its respective machine and delineates the commencement time for each task. Scheduling not only increases efficiency, optimizes production capacity, and bolsters profitability but also significantly reduces job completion durations, thereby conserving time–the most precious resource of all.

Manuscript Received: 1-September-2022 & Revised: 19-November-2022 & Accepted: 9-January-2023
ISSN: 2423-3781

One prominent scheduling model is shop scheduling, which involves the allocation of a set of tasks to a specific set of machines. Suppose there is a set of machines S $(i = 1,2, \dots, s)$, that must process a set of jobs N $(j = 1,2, \dots, n)$. A schedule can be defined as the allocation of time periods for processing these jobs across the machines. Notably, the open shop system, an important stream of shop scheduling, has its origins in large automotive garages specializing in vehicle repairs (Gonzalez & Sahni, 1976). In this problem, a system including a set of work stages S is considered that dynamically receives a set of jobs N over a given planning horizon. Since employing a single machine at each stage can lead to bottlenecks in the work stages, it is uncommon in practical scenarios to encounter a workshop where there is only one machine available at each stage (Rastgar et al., 2021; Rezvan et al., 2021), the work stage i could encompass a set of parallel machines m_i. In this regard, the extended open shop replaces a single machine at each stage with a collection of identical parallel machines (Kubiak, 2022). An open shop system with parallel machines is depicted in Fig. (1):

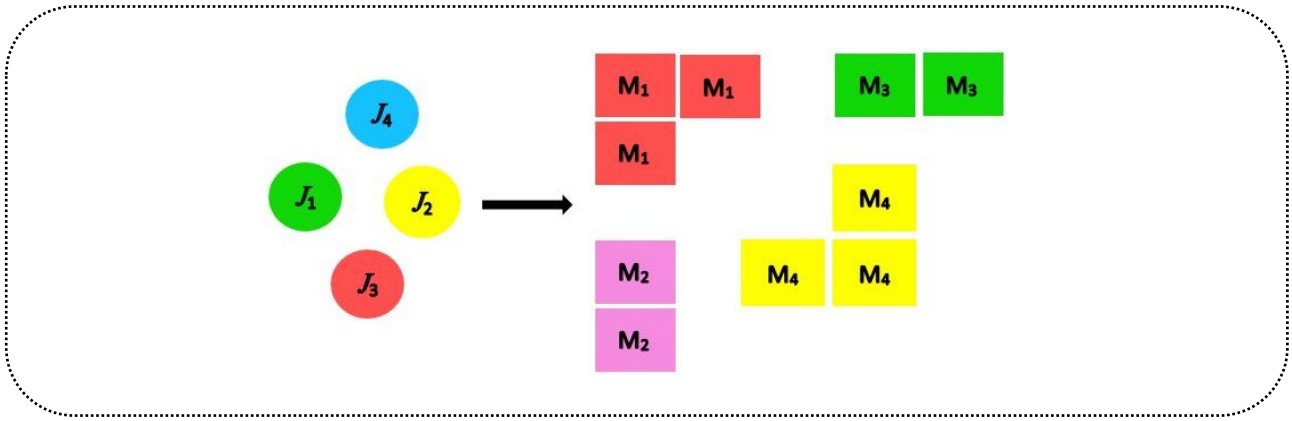

Fig. 1. Open shop scheduling with parallel machines

It is worth noting that in this context, the term 'machine' refers to any non-depletable resource capable of performing operations. The parallel machines within a particular work stage i may conduct similar types of operations (Abdelmaguid, 2020b). The open shop scheduling problem with parallel machines finds applications across diverse industrial sectors, including healthcare, automotive maintenance, electronics manufacturing, and quality control operations. Conversely, the proportionate case is predominantly prevalent in healthcare, especially medical testing, where each test requires a uniform amount of time, regardless of the number of patients involved (Adak, 2021). Importantly, there is no predetermined order for job processing. In other words, each job j must be visited exactly once in each work stage i and be processed by a single machine r within that stage. A job is considered completed once all its required operations have been executed (Kubiak, 2022). In the open shop scheduling problem, no fixed processing route is predefined for each job, and there is no predetermined sequence of jobs assigned to each machine. Consequently, for each job j, it is essential to determine the order of visits to the work stages and the corresponding machines to be employed. Similarly, for each machine in the system, it is imperative to determine the order in which jobs are processed. It is important to note that a job cannot be processed by more than one machine simultaneously, and each machine is limited to executing one job at a time. Furthermore, the processing times for jobs are considered deterministic positive integers and are contingent on both the specific job and the machine involved. Any interruptions during processing are strictly prohibited.

The primary objective in addressing the open shop scheduling problem is to minimize the makespan, defined as the maximum completion time of the jobs.

To the best of our knowledge, prior research on the open shop scheduling problem has traditionally overlooked a crucial factor: transportation times between machines. In these earlier studies, it was assumed that the initiation of a job's processing on a machine could commence immediately following the completion of the previous machine's operation. However, it is well-established that transportation times are a fundamental consideration within the field of

scheduling (Ahmadizar & Shahmaleki, 2014). In practical production scenarios, the completion of a job on one machine often necessitates its transfer to subsequent machines. Notably, in some domains, transportation times can exert a direct influence on product quality (Zhang et al., 2019). Consequently, in this paper, considerable emphasis is placed on incorporating transportation time considerations into our mathematical model.

Gonzalez & Sahni (1976) established the NP-hardness of the open shop scheduling problem when involving more than two processors, intending to minimize the makespan. Hence, the demand for efficient metaheuristic approaches is paramount to yield solutions within computationally acceptable time frames. Various research endeavors have utilized metaheuristic approaches to address scheduling issues (Esmaeili et al., 2021; Sharifzadegan et al., 2021). This study applies the whale optimization algorithm (WOA) to address the open shop scheduling problem to minimize the makespan. The novelty of our proposed model lies in its consideration of transportation times between machines within an open shop scheduling problem featuring parallel machines.

The subsequent sections of this paper are meticulously organized to facilitate an in-depth exploration of the topic. Section 2 presents a comprehensive review of pertinent prior studies. In Section 3, a mixed-integer linear programming model is introduced to address the open shop scheduling problem. Section 4 is dedicated to a detailed exposition of the metaheuristic algorithm, the whale optimization algorithm. Section 5 elaborates on our metaheuristic approach. In Section 6, the results and analysis of our computational experiments are unveiled. Concluding our work, Section 7 offers valuable insights and recommendations for future research endeavors.

II. LITERATURE REVIEW

In recent years, open shop scheduling problems have gained increasing attention among researchers. In this context, Barzegar et al. (2012) employed a hybrid approach that combines a genetic algorithm (GA) and tabu search (TS) to address the open shop scheduling problem. The primary objectives of this approach were to minimize both the maximum completion time and the total completion time. Naderia and Roshanaei (2014) considered an open shop scheduling problem that involves no-idle time scheduling to minimize the makespan. Initially, they introduced a mixed-integer linear programming model to tackle the problem. Furthermore, to address large-scale instances, the researchers developed and presented two metaheuristic approaches, namely the genetic algorithm (GA) and the simulated annealing (SA) algorithm. Rahmani Hosseinabadi et al. (2019) investigated the impact of operator selection, focusing specifically on crossover and mutation, within the framework of GA for optimizing the open shop scheduling problem. Their proposed algorithm was compared with other pre-existing algorithms to evaluate its effectiveness. Gu et al. (2019) proposed a hybrid whale optimization algorithm (WOA) aimed at minimizing the makespan in open shop scheduling problems. Abdelmaguid (2020b) conducted an in-depth investigation into a static and deterministic variation of the multiprocessor open shop scheduling problem, where processing times are influenced by both the job and the machine. The primary objective of this study was to minimize the makespan. Within the scope of this paper, two distinct neighborhood exploration functions and two solution combination functions were developed and effectively applied as part of a scatter search with path relinking metaheuristic. In a related research study, Abdelmaguid (2020a) explored the dynamic multiprocessor open shop scheduling problem (DMOSSP) with a focus on two concurrent objective functions: the reduction of the makespan and the mean weighted flow time. Subsequently, an exact algorithm founded on the ε-constraint method was introduced to produce optimal Pareto front solutions. It is worth noting that this algorithm is particularly well-suited for addressing smaller problem instances. Mejía and Yuraszeck (2020) explored open shop scheduling problems encompassing travel times between machines and sequence-dependent setup times using a variable neighborhood search (VNS) algorithm. Notably, this paper also incorporates a self-tuning routine to optimize the key algorithm parameters. Behnamian et al. (2021) addressed a flexible open shop scheduling problem characterized by two objective functions, specifically minimizing the makespan and total tardiness, along with independent setup times. Their approach involved the development of a mixed-integer nonlinear programming model. To tackle the inherent multi-objective nature of the problem, they employed the weighted LP-metric method. Additionally, they introduced a scatter search algorithm designed to yield near-optimal solutions within a reasonable computational timeframe.

In another research endeavor, Abdelmaguid (2021) conducted an investigation into the dynamic multiprocessor open shop scheduling problem focusing on minimizing both the mean weighted flow time and the makespan. This study led to the development of two metaheuristic approaches based on NSGA-II and the multi-objective grey wolf optimizer (MOGWO). Significantly, both metaheuristics were further enhanced through hybridization with a simulated annealing local search. Shareh et al. (2021) investigated the open shop scheduling problem to minimize makespan. They employed the bat algorithm (BA) based on a two-fold approach. The first component involves a heuristic function known as ColReuse, designed to quantify the difference between a randomly generated solution and the best solution. The second component employs substitution meta-heuristic functions to create a new permutation, which replaces the previous permutation if deemed suitable. Adak (2021) studied a novel solution representation design for a proportionate multiprocessor open shop aimed at minimizing makespan, where proportionality implies that processing times depend only on workstations and are independent of jobs. Additionally, Adak et al. (2022) applied an ant colony optimization (ACO) algorithm to reduce makespan in an open shop environment featuring parallel machines. This algorithm employs a randomized exploration of the solution space, enabling the efficient identification of favorable solution features in a time-efficient manner. Kurdi (2022) proposed an ACO algorithm for optimizing the open shop scheduling problem (OSSP). They introduced a new exploratory heuristic information approach (ACONEH) to minimize makespan. Their approach incorporates three key exploratory features. Firstly, it leverages randomness by generating heuristic information in a stochastic manner. Secondly, it prioritizes diversity, ensuring that each ant acquires distinct information. Finally, it emphasizes adaptability, with each ant periodically enhancing its acquired information.

Table I. Provides a summary of the research conducted on the open shop scheduling problem.

TABLE I. PREVIOUS STUDIES ON OPEN SHOP SCHEDULING PROBLEM

Reference	Modelling approach	Single/parallel machine	Transportation time	Solving approach
Gu et al. (2019)	MILP	Single	-	Whale optimization
Shareh et al. (2021)	MIP	parallel	-	An improved bat optimization algorithm
Mejía and Yuraszeck (2020)	MIP	Parallel	-	Self-tuning variable neighborhood search algorithm
Mafarja and Mirjalili (2018)	MILP	single	-	Whale optimization
Kurdi (2022)	MILP	Parallel	-	Ant colony optimization
Bezoui et al. (2023)	MILP	Parallel		Multi-objective genetic algorithm
Ying and Lin (2023)	MILP	Parallel	-	Iterated epsilon-greedy algorithm
Torres-Tapia et al. (2022)	MILP	Parallel	-	Ant colony system
Schworm et al. (2023)	MILP	Parallel	-	Quantum annealing
Current research	MILP	Parallel	√	Whale optimization

This paper endeavors to address a notable gap in the current literature pertaining to the open shop scheduling problem. Specifically, this study delves into the often-neglected aspect of transportation time for jobs between machines, which holds pivotal importance. The research leverages the whale optimization algorithm (WOA) to optimize the makespan in an open shop environment featuring parallel machines. Given the relatively scarce application of this optimization algorithm within the domain of open shop scheduling problems, a comparative analysis will be conducted with well-established algorithms to evaluate its effectiveness and performance.

III. MIXED-INTEGER LINEAR PROGRAMMING MODEL

This section introduces a mixed-integer linear programming (MILP) model under the following assumptions:

- There are a minimum of two parallel machines at each stage.

- The processing times are considered as positive integers.
- All jobs are initially available independently, with equal importance.
- Each job can be processed at a maximum of one workstage at any given time.
- Each machine is capable of processing only one job concurrently.
- Each job must be processed by one of the parallel machines.
- All jobs traverse through all stages, with the possibility of zero processing time on some stages (Pinedo, 2008).
- Each job can be processed in any sequence.
- Processing interruptions are not permitted.
- The transportation time of jobs between workstations, independent of specific jobs, is also taken into account.

In this context, two primary decisions need to be made in the process of minimizing the makespan: determining the processing route for each job across the machines and establishing the sequence of jobs on each machine.

The notations used in this model is outlined as follows:

Sets

S	Set of work stages $= \{1,2,\ldots,s\}$
N	Set of jobs $= \{1,2,\ldots,n\}$

Indexes

i, l	Stage indexes form set *S*
j, k	Job indexes form set *N*
r	Parallel machines index at each stage: $r \in \{1,2,\ldots,m_i\}$

Parameters

P_{ij}	Processing time of the *i*th job in the *j*th stage (a positive integer)
Tr_{il}	Transportation time of jobs between the machines in the *i*th and *l*th work stages, $\forall i,l \in S$, $Tr_{ii} = Tr_{ll} = 0$, $i \neq l$; $Tr_{il} = Tr_{li} \neq 0$
m_i	The number of parallel machines in the *i*th stage, $(m_i \geq 2)$
s	The number of work stages
M	A very large positive number

Decision variables

O_{ij}	A binary variable that equals 1 if the operation of the *j* th job is decided to be processed in the *i* th stage, and equals 0 otherwise
X_{ijk}	A binary variable that equals 1 if the operation O_{ik} is processed after the operation O_{ij}, and equals 0 otherwise

Y_{ijk}	A binary variable that equals 1 if the starting time of the operation O_{ij} is before the operation O_{ik} while having interference with each other, and equals 0 otherwise
Z_{ilj}	A binary variable that equals 1 if the operation O_{lj} is exactly processed after the operation O_{ij}, and equals 0 otherwise
V_{irj}	A binary variable that equals 1 if the *j*th job is decided to be processed on the *r*th machine in the *i*th stage, and equals 0 otherwise
St_{ij}	The start time when the *j*th job begins processing in the *i*th stage (or entry time of the *j*th job into the *i*th stage)
C_{ij}	The time that the processing of the *j*th job in the *i*th stage is completed
C_j	The time that the *j*th job is fully processed and completed

Accordingly, the open shop scheduling problem is modeled as a mixed-integer linear programming problem with the aim of minimizing the makespan as follows:

$$\textbf{Min } C_{max} = max\, C_j \tag{1}$$

s.t.

$$C_j \geq C_{ij} \quad ; \quad \forall i \in S, \quad \forall j \in N \tag{2}$$

$$C_{ij} = St_{ij} + P_{ij} \quad ; \quad \forall i \in S, \quad \forall j \in N \tag{3}$$

$$St_{lj} \geq C_{ij} + Tr_{il} - M(1 - Z_{ilj}) \quad ; \quad \forall i, l \in S, \quad \forall j \in N, \quad i \neq l \tag{4}$$

$$St_{ij} \geq C_{lj} + Tr_{li} - M\left(1 - Z_{lij}\right) \quad ; \quad \forall i, l \in S, \quad \forall j \in N, \quad i \neq l \tag{5}$$

$$St_{ik} \geq C_{ij} - M\left(1 - X_{ijk}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{6}$$

$$St_{ij} \geq C_{ik} - M\left(1 - X_{ikj}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{7}$$

$$St_{ik} \geq St_{ij} - M\left(1 - Y_{ijk}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{8}$$

$$C_{ij} \geq St_{ik} + 1 - M\left(1 - Y_{ijk}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{9}$$

$$St_{ij} \geq St_{ik} - M\left(1 - Y_{ikj}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{10}$$

$$C_{ik} \geq St_{ij} + 1 - M\left(1 - Y_{ikj}\right) \quad ; \quad \forall i \in S, \quad \forall j, k \in N, \quad j \neq k \tag{11}$$

$$\sum_{l\in S-\{i\}} Z_{ilj} \le 1 \quad ; \ \forall i \in S, \ \forall j \in N \tag{12}$$

$$\sum_{i\in S-\{l\}} Z_{ilj} \le 1 \quad ; \ \forall l \in S, \ \forall j \in N \tag{13}$$

$$\sum_{i\in S-\{l\}} \sum_{l\in S-\{i\}} Z_{ilj} = s-1 \quad ; \ \forall j \in N \tag{14}$$

$$X_{ijk} + X_{ikj} + Y_{ijk} + Y_{ikj} = 1 \quad ; \ \forall i \in S, \quad \forall j,k \in N, \quad j \ne k \tag{15}$$

$$Y_{ijk} + Y_{ikj} + V_{irj} + V_{irk} \le 2 \quad ; \ \forall i \in S, \quad \forall r \in M_i, \quad \forall j,k \in N, \quad j \ne k \tag{16}$$

$$\sum_{r=1}^{m_i} V_{irj} = 1 \quad ; \ \forall i \in S, \ \forall j \in N \tag{17}$$

$$St_{ij}, C_{ij}, C_j \in Z^+ \cup \{0\} \quad ; \ \forall i \in S, \quad \forall j \in N \tag{18}$$

$$X_{ijk} \in \{0,1\}, \ Y_{ilj} \in \{0,1\}, \ Z_{ijk} \in \{0,1\}, \ V_{irj} \in \{0,1\} \tag{19}$$

The objective of minimizing the makespan is formulated by Eq. (1). Constraint (2) denotes the final completion time of the *j*th job. The equality in constraint (3) guarantees uninterrupted and continuous job processing. Constraints (4) and (5) articulate the sequential progression of stages for a given job, specifying stages that immediately follow one another. Constraints (6) and (7) express the sequencing of jobs within a specific stage. Constraints (8) to (11) indicate that the next job starts after begins the current job and before its completion. The presence of '1' in Constraints (9) and (11) indicates that one job does not start immediately after the completion of another job because, otherwise, this scenario would be addressed by Constraints (6) and (7) since the minimum difference between two integers is one.

Constraint (12) ensures that the *j*th job after the *i*th stage is only transferred to one of the remaining stages, indicating that it doesn't simultaneously enter two or more stages. Alternatively, the job will not proceed after the current, indicating that the current stage is the last. Furthermore, Constraint (13) guarantees that the *j*th job will enter the *l*th stage from only one of the existing stages, or it will not enter the current stage from any stage, implying that the current stage is the initial stage. Equation (14) ensures that both Constraints (12) and (13) cannot simultaneously be zero for each job. When Constraint (12) equals zero, it signifies the final stages, and when Constraint (13) equals zero, it implies the initial stage.

Constraint (15) ensures that only one of the variables within this constraint can take a value of one.. Constraint (16) ensures that if multiple jobs commence simultaneously in a certain stage or if their processes overlap within the same stage, the machine assigned to one job will not be assigned to another. Finally, constraint (17) guarantees that each machine at each stage is assigned to only one job at any given moment.

IV. METAHEURISTIC APPROACH

In this study, a recently developed population-based metaheuristic algorithm known as the "Whale Algorithm" has been employed to address the problem. Inspired by the social behavior of humpback whales, this algorithm proves particularly effective in solving complex optimization challenges. The Whale Algorithm begins with an initial set of random solutions. In each iteration, the search agents adjust their positions based on either a random search agent or the

best solution obtained thus far. Humpback whales employ two distinct hunting strategies: direct attack within a narrow siege or pursuit of prey along a spiral path. In the following sections, we elaborate on the modeling of these two strategies.

A. Prey siege modeling

Humpback whales engage in a randomized search process based on the positions of their fellow whales. They continuously update their locations and directions while considering the movements of other whales. This behavior can be described by the following equations proposed by Mirjalili and Lewis (2016):

$$D = \left|C.\vec{X}_{rand} - \vec{X}(t)\right| \tag{20}$$

$$\vec{X}(t+1) = \vec{X}_{rand} - A.D \tag{21}$$

The $\vec{X}_{rand}$ vector represents a randomly selected position vector, which corresponds to a random whale chosen from the current population. In this context, "t" denotes the current iteration number, and $\vec{X}$ represents the position vector of a particular solution. The values of "A" and "C" can be calculated as described by Mirjalili and Lewis (2016):

$$A = 2.a.r - a \tag{22}$$

$$C = 2.r \tag{23}$$

During the course of the algorithm's iterations, the parameter "a" decreases linearly from 2 to 0. Simultaneously, "r" represents a random value within the range $[0,\ 1]$. To promote the dispersion of search agents away from a reference whale, the parameter "A" is randomized to fall within the range $[-1,\ 1]$. By adopting this mechanism and ensuring that $|A| \geq 1$, the algorithm facilitates exploration, enabling a comprehensive search.

Following the search process and the identification of solutions in each iteration, the whales operate under the assumption that the best solution attained by the current candidate is either the optimal solution or close to it. Once the best search factor is identified, other search agents endeavor to adjust their positions in the direction of this best factor. This dynamic is mathematically represented by the following equations, as described by Mirjalili and Lewis (2016):

$$D = \left|C.\vec{X}^{*}(t) - \vec{X}(t)\right| \tag{24}$$

$$\vec{X}(t+1) = \vec{X}^{*}(t) - A.D \tag{25}$$

The vector $\vec{X}^{*}$ represents the position of the best solution obtained up to the current iteration. It is crucial to update $\vec{X}^{*}$ when a superior solution is acquired during each iteration. Consequently, in order to update the positions of search agents, the "random search agent" is selected when $|A| \geq 1$, while "best solution" is selected when $|A| < 1$.

B. Logarithmic helix modeling

This method initiates by computing the distance between the whale positioned at X and the prey situated at X^{*},

subsequently generating a spiral equation that connects the whale's location to the prey. This process mimics the attack strategy of humpback whales, as outlined below (Mirjalili & Lewis, 2016):

$$D' = \left|\vec{X}^*(t) - \vec{X}(t)\right| \tag{26}$$

$$\vec{X}(t+1) = D'.e^{b.l}.cos(2\pi l) + \vec{X}^*(t) \tag{27}$$

The vector D' represents the distance between the ith whale and the prey (the best solution obtained thus far). Here, "b" serves as a constant defining the logarithmic spiral shape, while "l" signifies a random number within the range $[-1,\ 1]$.

The whale algorithm seamlessly transitions between two strategies based on the value of the random number "p", which varies within the range $[0,\ 1]$. If $p \geq 0.5$, then the algorithm opts for the spiral motion strategy, whereas a shrinking siege strategy is employed when $p < 0.5$. Ultimately, the whale algorithm concludes its execution upon meeting the predefined termination conditions. The pseudocode for WOA is depicted in Fig. (2).

Input data, Number of maxiter and population etc.
Initialize the whales' population X_i *(*i *= 1, 2, ..., n)*
Initialize *a, A, C, l* and *p*
Calculate the fitness of each search agent
*X** = the best search agent
while ($t < t_{max}$)
  for each search agent
    if ($p < 0.5$)
      if ($|A| < 1$)
        Update the position of the current search agent by the equation:
        $X(t+1) = X^*(t) - A.D$
      else if ($|A| \geq 1$)
        Select a random search agent (X_{rand});
        Update the position of the current search agent by the equation:
        $X(t+1) = X_{rand}(t) - A.D$
      end
    else if ($p \geq 0.5$)
      Update the position of the current search by the by the equation:
      $X(t+1) = D'.e^{b.l}\,cos(2\pi l) + X^*(t)$
    end if
  end for
Calculate the fitness of each search agent
Update *X** if there is a better solution
t=t+1
Update *a, A, C, l* and *p*
end while
return *X**

Fig. 2. Pseudo-code of WOA (Mafarja & Mirjalili, 2018)

V. SUGGESTED METHOD

Evidently, the prudent choice of algorithm specifications, encompassing solution representation, initial population generation, solution evaluation during each iteration, and termination criteria, significantly impacts its overall efficiency. The ensuing sections expound upon the method employed in the whale algorithm to address these crucial aspects.

A. Representation of solutions

In the context of the open shop scheduling problem under investigation in this research, a set of parallel machines is designated at each stage. This entails the allocation of the first "type" of machines in the initial stage, followed by the allocation of the second "type" of machines in the subsequent stage, and so forth. Each "type" of machine possesses distinct performance characteristics, and at each stage, each job is processed by only one of the parallel machines. With these considerations in mind, the solution vector for the open shop scheduling problem comprises two parts, as outlined in Table II: the sequence of operations and the allocation of parallel machines at each stage.

Table II. Vector of solutions

Sequence of Operations	Allocation of Parallel Machines

To represent the sequence of operations, one common approach is to employ a "permutation list", which is generated randomly. For a system consisting of "n" jobs and "m" work stages, the permutation list encompasses a total of $n*m$ operations. These permutations are denoted as O_{ji}, where the index "j" represents the job, and the index "i" represents the work stage.

To encode each of these permutations, a natural number is assigned based on their order and sequence from left to right. An example of this assignment is shown in Table III for permutations involving 2 jobs and 3 stages:

Table III. Correlation of permutations and natural numbers for 2 jobs and 3 stages

Permutation O_{ji}	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}
Natural Number	1	2	3	4	5	6

The second step involves the assignment of parallel machines, with the number of permutations in this step being proportional to those generated in the first step. For each permutation in the first step, a corresponding permutation is created in the second step for the assignment of parallel machines. As in the the previous step, this step is created randomly.

Now, the general representation method of a possible solution is illustrated through an example in Table IV:

Table IV. An example of representing a possible solution for the problem

3	6	1	4	5	2		1	2	2	3	1	1

B. Producing the initial population

To randomly generate the initial population, a permutation list equal to the initial population is generated. Essentially, a series of whales is randomly created, each with distinct and random characteristics. This process is repeated to match the population size.

C. Modeling the algorithm

Let time t_1 represent the completion time of the last operation for the current job up to the current moment in stage *A*. Additionally, suppose that after this operation in stage *A*, the job will proceed to another new operation in a different stage, such as stage *B*. Consequently, the transportation time tr between stages *A* and *B* will be added to the completion time.

$$a= t_1 + tr \tag{28}$$

Let's assume that in stage *B*, the machine responsible for processing this job becomes available at time $b = t_2$.

Consequently, the start time for processing this job in this stage is determined as follows:

$$St = Max\ \{a, b\} \tag{29}$$

The completion time of a job at each stage is then calculated as follows:

$$Ft = St + Pt \tag{30}$$

Here, Pt represents the processing time of the job at each stage.

To illustrate, consider the random permutation for the sequence of operations shown in Table IV:

Permutation $\{3 \times 2\}= (3, 6, 1, 4, 5, 2)$

Where, the first operation of this permutation, represented by the number 3, corresponds to operation O_{13}. This signifies that the processing of the first job in the third stage is assigned to machine 1, as indicated in Table IV. This operation has a time duration Pt:

j=1, *s*=3, *m*=1

Assume operation O_{13} is the initial operation of the respective job at time $t_1 = 0$. Furthermore, since this operation is the first one in stage 3 and no prior operations have taken place, then $t_2 = 0$, Consequently:

$$St = Max\ \{\text{a, b}\}= Max\ \{0, 0\}=0 \tag{31}$$

$$Ft = St\ +\ Pt = Pt \tag{32}$$

D. Cost function

The objective is to minimize the final completion time of jobs. Therefore, the cost function is defined as the final completion time of jobs. In the proposed algorithm, this function indicates the degree of compliance or competence of each whale or solution and is defined as follows:

$$CostFunction = C_{max} \tag{33}$$

E. Algorithm Termination condition

The termination criterion in this research is based on the number of algorithm iterations.

VI. COMPUTATIONAL EXPERIMENTS AND RESULTS

To evaluate the performance of the proposed algorithm in addressing the open shop scheduling problem under investigation, first, the parameters of this algorithm are introduced. Following this, the optimization of these parameters is delved into using the Taguchi method and the computational evaluation of the problem is subsequently conducted using the whale algorithm.

A. Tuning metaheuristics parameters

The whale algorithm operates with two key parameters: population size and the number of algorithm iterations. Selecting optimized values for these parameters can positively influence the algorithm's performance.

In this study, the parameters of the whale algorithm undergo optimization via the Taguchi method, facilitated by Minitab software. To accomplish this, a large-scale instance with dimensions of 10×10 is considered. The processing times of jobs in this model adhere to a uniform distribution of integers within the range of 1 to 10. Designing experiments for this instance entails considering of various parameter values. Therefore, three different levels have been designated for each parameter, as outlined in Table V:

Table V. Level of parameters

	Level 1	**Level 2**	**Level 3**
Npop	20	30	40
Maxiter	300	400	500

Subsequently, the Minitab software generates the proposed scenarios based on the number of parameters and levels. This is achieved through the design of experiments using the Taguchi method, and the resulting scenarios are presented in Table VI.

Table VI. Scenarios suggested for adjustment of parameters

	Sen 1	**Sen 2**	**Sen 3**	**Sen 4**	**Sen 5**	**Sen 6**	**Sen 7**	**Sen 8**	**Sen 9**
Npop	Level 1	Level 1	Level 1	Level 2	Level 2	Level 2	Level 3	Level 3	Level 3
Maxiter	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3

Each of these scenarios is then subjected to resolution by the whale algorithm through five distinct implementations. The average ($\bar{c}_{max}$) is calculated and displayed from these five implementations for each scenario, as outlined in Table VII:

Table VII. Average results of the suggested scenarios

	Sen 1	**Sen 2**	**Sen 3**	**Sen 4**	**Sen 5**	**Sen 6**	**Sen 7**	**Sen 8**	**Sen 9**
$\bar{c}_{max}$	112.8	110.6	104.6	108.4	107.8	109.8	114.6	114.8	109.8

Through the analysis of the results using Minitab, the signal-to-noise ratio (SNR) for both parameters is calculated based on the average outputs ($\bar{c}_{\max}$). The diagram presented in Fig. (3) illustrates the software's output.

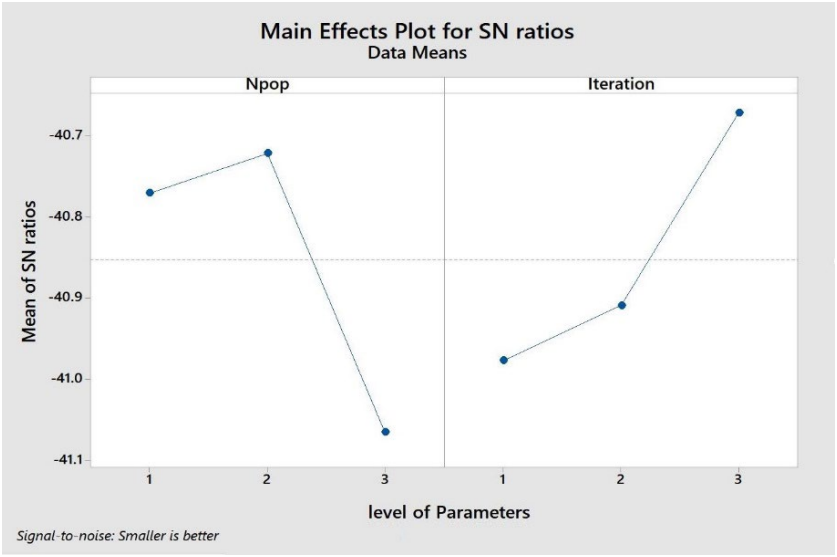

Fig. 3. Diagram of Signal-to-Noise Ratio of Parameters

In Fig. (3), the horizontal axis corresponds to the number of levels of both algorithm parameters, while the vertical axis represents the average signal-to-noise ratio (S/N). The term "signal" generally refers to the desirable value (average response variable), and "noise" denotes the undesirable value (standard deviation). The objective is to maximize the S/N ratio. As depicted in Fig. (3), the optimal value for the population size parameter is at level 2, which is equivalent to 30, and for the number of iterations parameter, it is at level 3 with a value of 500, as indicated in Table VIII:

Table VIII. Algorithm parameters & their corresponding values

Parameter	Value
Population Size	30
Algorithm Iterations Number	500

B. Computational results

To assess the efficiency of the presented algorithm, two categories of problems are considered: the first category comprises small and medium instances, while the second one deals with large instances. In the initial phase, random instances with small and medium sizes are used to solve the presented linear model and measure the efficiency of the whale algorithm, benchmarked against the optimal solution generated by the model using GAMS 24.8.3 software. The algorithm is implemented via MATLAB software on a personal computer with Intel Core 2 Quad 2.50 GHz specifications and 4 GB of RAM.

For small and medium-sized instances, the processing times follow a homogenous distribution of integers within the range of [1,99], generated using the GAMS uniformint() function. At each stage, the number of parallel machines ranges from 2 to 3, and the number of jobs and stages varies from 2, 3, 4, and 5. To evaluate the algorithm's efficiency, the results are presented for ten random small and medium-sized instances. The average outcomes of five algorithm implementations for these instances are compared with the optimal solutions obtained through GAMS as shown in Table IX.

The analysis highlights the exceptional efficiency of the algorithm in handling randomly generated small and

medium-sized instances. This efficiency is underscored by the consistent alignment of the whale algorithm's minimum values and the average results from five algorithm runs with those achieved by GAMS. Essentially, the whale algorithm consistently attains the same optimal solutions as GAMS, thus affirming its reliability in obtaining general optimal solutions.

Table IX. Computational results of linear model & algorithm for small & medium-sized problems

Instance No.	Number of Jobs	Number of Stages	Number of Parallel Machines at each Stage	GAMS		WOA		
				Objective Function	Solution Time (s)	The Minimum Value of Objective Function	Solution Time of Minimum Value (s)	The Average of 5 Runs
1	2	2	(2,2)	130	0.09	130	79.59	130
2	2	3	(2,3,2)	152	0.16	152	82.65	152
3	3	2	(2,2)	162	0.23	162	83.58	162
4	3	3	(2,2,2)	186	0.3	186	90.34	186
5	3	4	(3,2,2,3)	193	1.22	193	95.82	193
6	4	3	(2,2,2)	212	0.89	212	95.79	212
7	4	4	(2,3,2,3)	290	2.96	290	101.93	290
8	5	3	(3,2,3)	202	1.08	202	100.49	202
9	5	4	(2,2,3,3)	281	2.34	281	108	281
10	5	5	(2,3,3,2,2)	354	7.48	354	161.4	354

Table X. A Comparison of the averages of five runs for three algorithms on each instance

Instance No.	Number of Jobs	Number of Stages	Number of Parallel Machines at Each Stage	WOA	PSO	DE
1	6	5	(2,3,2,3,2)	415	415	415
2	6	6	(2,3,2,3,2,2)	496.2	496.3	496
3	7	6	(2,3,2,3,2,2)	424.2	424	424
4	7	7	(2,3,2,3,2,2,3)	533	534.6	533.5
5	8	7	(2,3,2,3,2,2,3)	537.8	539.4	537.2
6	8	8	(2,3,2,3,2,2,3,2)	547.6	551.2	548.9
7	9	8	(2,3,2,3,2,2,3,2)	596.5	598.7	598.25
8	9	9	(2,3,2,3,2,2,3,2,3)	600.8	600.4	600.8
9	10	9	(2,3,2,3,2,2,3,3,3)	605.9	617.6	614.8
10	10	10	(2,3,2,3,2,2,3,3,3,2)	680.8	682.4	676.7

However, to comprehensively assess its performance, the Whale Algorithm (WOA) has been benchmarked against other significant optimization algorithms, namely Differential Evolution (DE) and Particle Swarm Optimization (PSO), particularly on larger instances. All of these algorithms have been meticulously simulated in MATLAB software.

For these large-scale instances, processing times that follow a uniform distribution of integers within the range of [1, 99] have been chosen, generated using the MATLAB randi() function. Each stage is equipped with 2 to 3 parallel machines, and the total number of jobs and stages ranges from 5 to 10. To conduct this comparative analysis, ten randomly generated large-sized instances have been employed, each of which has been subjected to optimization by WOA, DE, and PSO algorithms. The average of the results from five implementations for each algorithm applied to each instance, have been computed. The comparative findings are presented in Table X.

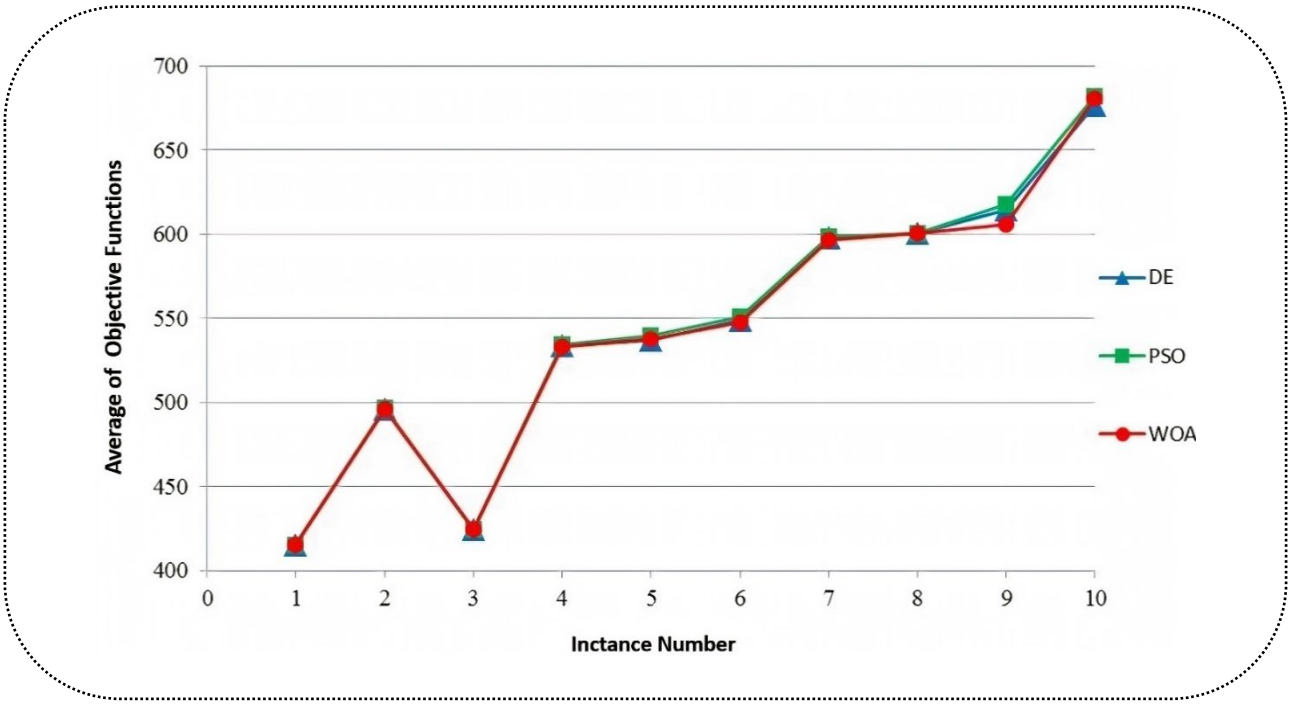

Fig. 4. Comparative diagram of the average of five runs of all three algorithms in Table X

The numerical results obtained from Table X are depicted in Fig. (4). In this comparative diagram, the horizontal axis represents the instance number, while the vertical axis displays the average value of the objective functions derived from five runs of each algorithm.

When comparing the average results obtained from five runs of the whale algorithm with those of two alternative algorithms, PSO and DE, it was observed that the whale algorithm excels in solving large instances. Particularly, in instances 4, 6, 7, and 9, the whale algorithm outperforms both PSO and DE. This highlights the algorithm's remarkable efficiency and high-performance capabilities in addressing large-scale open shop scheduling problems. It emphasizes its proficiency in navigating the trade-off between exploration and exploitation, thereby effectively circumventing local optima.

VII. CONCLUSIONS AND FUTURE WORK

The open shop scheduling problem (OSSP) stands as a widely acknowledged scheduling challenge with significant implications for diverse industries. Achieving an optimal and practical scheduling solution within this domain holds the potential to enhance industrial operations. This paper specifically tackled the open shop scheduling problem within the

context of parallel machines, integrating the consideration of transportation times between workstations — an aspect that has not been extensively explored in existing literature. To address this gap, a mixed-integer linear programming (MILP) model was formulated, primarily aimed at minimizing the makespan in systems of this nature. Given the NP-hard nature of the OSSP, the study employed the emerging metaheuristic technique known as the whale algorithm for effective problem-solving.

Based on the analysis that was conducted, which involved comparing the results obtained from implementing the linear model in GAMS with those from the whale algorithm, it is evident that the algorithm operates very efficiently when handling randomly generated small and medium-sized instances. This is supported by the observation that the minimum values obtained by the whale algorithm, as well as the average results from five runs of this algorithm, consistently align with the outcomes of GAMS. Essentially, the whale algorithm consistently achieves the same optimal solution as GAMS, showcasing its reliability in obtaining general optimal solutions.

Furthermore, in the comparative analysis of the average results from five runs of the whale algorithm with those from two other algorithms, PSO and DE, it was found that the whale algorithm's performance in solving large instances was not only competitive, but in specific instances (4, 6, 7, and 9), surpassed the results achieved by the other two algorithms. This suggests that the whale algorithm retains its efficiency and high performance when tackling large-scale open shop scheduling problems. This highlights its ability to strike a balance between exploration and exploitation, thereby avoiding local optimal solutions.

As a basis for future research in this domain, it is recommended to explore scenarios where distinct machines are allocated to each workstation instead of identical ones. Additionally, the inclusion of multiple vehicles for job transportation, each with its own unique transportation times, can offer new insights. Another avenue for research is to investigate situations where machines are mobile and jobs are stationary. It would also be beneficial to explore the application of alternative metaheuristic algorithms and optimize various objective functions, including scenarios with multiple objectives.

REFERENCES

Abdelmaguid, T. F. (2020). Bi-objective dynamic multiprocessor open shop scheduling: an exact algorithm. *Algorithms*, *13*(3), 74.

Abdelmaguid, T. F. (2020). Scatter search with path relinking for multiprocessor open shop scheduling. *Computers & Industrial Engineering*, *141*, 106292.

Abdelmaguid, T. F. (2021). Bi-objective dynamic multiprocessor open shop scheduling for maintenance and healthcare diagnostics. *Expert Systems With Applications*, *186*, 115777.

Adak, Z. (2021). Solution Representation in Proportionate Multiprocessor Open Shop. *Journal of Intelligent Systems: Theory and Applications*, *4*(2), 86–93.

Adak, Z., Arıoğlu, M. Ö., & Bulkan, S. (2022). An ant colony optimization approach for the proportionate multiprocessor open shop. *Journal of Combinatorial Optimization*, *43*, 785–817.

Ahmadizar, F., & Shahmaleki, P. (2014). Group shop scheduling with sequence-dependent setup and transportation times. *Applied Mathematical Modelling*, *38*(21–22), 5080–5091.

Barzegar, B., Motameni, H., KHOSROZADEH, G. A., & Divsalar, A. (2012). A Hybrid Genetic Algorithm for the Open Shop Scheduling with Makespan and Total Completion Time.

Behnamian, J., Memar Dezfooli, S., & Asgari, H. (2021). A scatter search algorithm with a novel solution representation for flexible open shop scheduling: a multi-objective optimization. *The Journal of Supercomputing*, *77*, 13115-13138.

Bezoui, M., Olteanu, A. L., & Sevaux, M. (2023). Integrating preferences within multiobjective flexible job shop scheduling. *European Journal of Operational Research*, *305*(3), 1079–1086.

Esmaeili, M., Ahmadizar, F., & Sadeghi, H. (2021). Minimizing the sum of earliness and tardiness in single-machine scheduling. *Journal of Quality Engineering and Production Optimization*, *6*(2), 59-78.

Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. *Journal of Cloud Computing*, *7*, 1-16.

Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM (JACM)*, *23*(4), 665–679.

Gu, H. M., Hu, R., Qian, B., Jin, H. P., & Wang, L. (2019). Whale optimization algorithm with local search for open shop scheduling problem to minimize makespan. In Intelligent Computing Theories and Application: 15th International Conference, ICIC 2019, Nanchang, China, August 3–6, 2019, Proceedings, Part II 15 (pp. 678-687). Springer International Publishing.

Kubiak, W. (2022). *Book of Open Shop Scheduling*. Springer International Publishing.

Kurdi, M. (2022). Ant colony optimization with a new exploratory heuristic information approach for open shop scheduling problem. *Knowledge-Based Systems*, *242*, 108323.

Mafarja, M. M., & Mirjalili, S. (2018). Whale Optimization Approaches for Wrapper Feature Selection. *Applied Soft Computing Journal*, *62*, 441–453.

Mejía, G., & Yuraszeck, F. (2020). A self-tuning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel / setup times. *European Journal of Operational Research*, *285*(2), 484–496.

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, *95*, 51–67.

Naderia, B., & Roshanaei, V. (2014). No-idle time Scheduling of Open shops: Modeling and Meta-heuristic Solution Methods. *International Journal of Supply and Operations Management*, *1*(1), 54–68.

Pinedo Michael, L. (2008). Scheduling Theory, Algorithms, and Systems. New York University. ISBN: 978-0-387-78934-7, e-ISBN: 978-0-387-78935.

Rahmani Hosseinabadi, A. A., Vahidi, J., Saemi, B., Sangaiah, A. K., & Elhoseny, M. (2019). Extended genetic algorithm for solving open-shop scheduling problem. *Soft computing*, *23*, 5099-5116.

Rastgar, I., Rezaeian, J., Mahdavi, I., & Fattahi, P. (2021). Opportunistic maintenance management for a hybrid flow shop scheduling problem. *Journal of Quality Engineering and Production Optimization*, *6*(2), 17-30.

Rezvan, M. T., Gholami, H., & Zakerian, R. (2021). A new algorithm for solving the parallel machine scheduling problem to maximize benefit and the number of jobs processed. *Journal of Quality Engineering and Production Optimization*, *6*(2), 115-142.

Schworm, P., Wu, X., Glatt, M., & Aurich, J. C. (2023). Solving flexible job shop scheduling problems in manufacturing with Quantum Annealing. *Production Engineering*, *17*(1), 105–115.

Shareh, M. B., Bargh, S. H., Hosseinabadi, A. A. R., & Slowik, A. (2021). An improved bat optimization algorithm to solve the tasks scheduling problem in open shop. *Neural Computing and Applications*, *33*, 1559–1573.

Sharifzadegan, M., Tahmoores Sohrabi, & Chaghoshi, A. J. (2021). Hybrid optimization of production scheduling and maintenance using mathematical programming and NSGA-II meta-heuristic method. *Journal of Quality Engineering and Production Optimization*, *6*(2), 79-96.

Torres-Tapia, W., Montoya-Torres, J. R., Ruiz-Meza, J., & Belmokhtar-Berraf, S. (2022). A Matheuristic based on Ant Colony System for the Combined Flexible Jobshop Scheduling and Vehicle Routing Problem. *IFAC-PapersOnLine*, *55*(10) 1613-1618 .

Ying, K. C., & Lin, S. W. (2023). Minimizing makespan in two-stage assembly additive manufacturing: A reinforcement learning iterated greedy algorithm. *Applied Soft Computing*, *138, 110190*.

Zhang, G., Sun, J., Liu, X., Wang, G., & Yang, Y. (2019). Solving flexible job shop scheduling problems with transportation time based on improved genetic algorithm. *Mathematical Biosciences and Engineering*, *16*(3), 1334–1347.