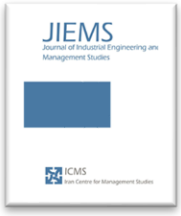
JIEMS
Journal of Industrial Engineering and Management Studies

journal homepage: www.jiems.icms.ac.ir



# A two-objective Mathematical Model for Job Scheduling on Parallel Machines and Solving by Particle Swarm Optimization

Shahram Saeidi[1*]

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

Received: Jan 2025-09/ Revised: Feb 2025-12/ Accepted: Feb 2025-14

**Abstract**

Time is one of the most valuable assets in industry, and cost is another highly regarded factor. Optimal utilization of these resources can increase efficiency and profit. The parallel machine scheduling problem is a fundamental issue in industry and services. This research proposes a two-objective mathematical model for parallel machine scheduling. The first objective function is defined as the makespan, which is the completion time of the last job. The second objective function is defined as the maximum cost incurred by any single machine, which is a function of the sum of the processing costs of each operation and the fixed cost of purchasing and maintaining the machines. Each job consists of multiple operations, and all operations must be completed to finish the job. Additionally, it is assumed that jobs have priorities, and precedence constraints between operations must be satisfied. Due to the model's non-linearity and the problem's complexity, a metaheuristic algorithm based on the particle swarm optimization (PSO) approach is developed to solve the proposed model by aggregating the objective functions. The proposed method is simulated in MATLAB on three sample instances in small, medium, and large scales. The computational results demonstrate the robustness and efficiency of the proposed method.

**Keywords**: Job Scheduling, Linear Programming, Parallel Machines, Particle Swarm Optimization.

**Paper Type**: Original Research

## 1. Introduction

In today's industrial world, the dwindling availability of production resources such as machinery and equipment, coupled with escalating energy costs and increasing machine downtime, has significantly elevated the value of optimizing resource and time utilization within production systems (Hao et al., 2014). One particular area of interest among industrial researchers and practitioners is the concept of scheduling. Scheduling, which follows the determination of task sequences, defines the start and end times for each task in a workshop. These times are contingent upon the task's arrival time, machine availability, and the completion of preceding tasks. The scheduling process must be designed to fulfill the objectives defined by production management. Implementing an effective and efficient scheduling plan to determine the production sequence is fundamentally linked to enhancing the productivity of production systems (Aggoune, 2004). In sequencing, scheduling, and operations, scheduling refers to allocating and determining start and end times for operations on available machines within a production system. The static scheduling problem of tasks on heterogeneous parallel machines holds significant importance due to the need for optimal machine utilization and reduced processing time. This problem is categorized as NP-hard (Aggoune, 2004). In this problem, a set of parallel machines with varying processing speeds (and costs) are available, and some tasks are to be processed by these machines in a workshop. Each task is divided into multiple processes. Processes may have priorities for execution. Some or all processes may be interdependent, and all processes can be executed in parallel on different machines. Tasks are arranged in a waiting queue. Scheduling is performed for a specific planning horizon, and no new tasks enter the system until the end of the period. In this system, there are a limited number of heterogeneous machines. Each machine has a different processing speed and operational cost compared to the others. Due to the diversity of machines, parallel processing, many tasks, and the inherent complexity of the problem, obtaining an optimal solution requires significant computational time (Naseri et al., 2013). Therefore, metaheuristic methods are used to solve this problem. This research investigates the parallel machine scheduling problem with the dual objectives of minimizing the total completion time (makespan) and the total processing cost under the following assumptions. A mathematical programming model is presented for scheduling

*Corresponding Author: sh_saeidi@iaut.ac.ir

tasks on heterogeneous parallel machines, and a particle swarm optimization algorithm is proposed to solve the model. The model's assumptions are listed below.

- The production machines are heterogeneous: They differ in their capabilities or characteristics.
- The system is static: The system's conditions or parameters do not change over time.
- There is no priority among jobs: All jobs are equally important, and there is no preference for one job over another.
- The scheduling method is non-preemptive: Once a job starts, it cannot be interrupted until it is completed.
- The processing time and cost of each job on each machine are known and deterministic: The time taken and the cost incurred for each job on a specific machine are fixed and can be determined beforehand.
- Dependencies between processes are considered: The order in which jobs are executed is constrained by specific dependencies or prerequisites.
- Priorities among processes are considered: Some jobs are more important than others and should be scheduled accordingly.
- Setup time for each process on each machine is considered: The time required to prepare a machine for a specific job is factored into the scheduling process.

This research investigates the parallel machine scheduling problem with the dual objectives of minimizing the total completion time (makespan) and the total processing cost under the considered assumptions.

The rest of the paper is organized as follows: The second section deals with the literature review on the subject and previous work. The third section discusses the proposed mathematical model, the research methodology, and data analysis. The fourth section examines the findings and simulation results. The conclusions are discussed in the fifth section.

## 2. Literature Review and Previous Work

This section defines and introduces the subject literature's concepts, methods, and standard definitions and terminology. Subsequently, previous studies are reviewed.

### 2.1 Sequencing and Scheduling

Sequencing and scheduling is a decision-making process pivotal in enhancing productivity within manufacturing and service industries. In today's competitive environment, optimizing the sequence of operations and scheduling activities has become a fundamental requirement for organizational survival. In essence, there are no orders without due dates. Sequencing refers to determining the order in which operations are processed, while scheduling involves assigning start and end times for operations on resources (Aggoune, 2004). Scheduling is contingent upon sequencing; the order of operations must be established before assigning specific times. In other words, scheduling is a decision-making activity to optimize one or more objectives. Depending on the particular process, there may be prerequisites, earliest start times, and due dates to consider. The objectives of sequencing problems can vary widely (Pinedo,2008).

### 2.2 Parallel Machine Scheduling

In the simplest case, a single machine executes tasks sequentially, following a predefined objective function. However, when multiple machines are available, they can process tasks concurrently, thus reducing the overall completion time. Parallel machine systems can be classified into two categories. In homogeneous systems, all machines have identical processing capabilities and costs; examples include security screening, banking operations, and passport control. In heterogeneous systems, machine processing speeds vary based on their type or model, independent of the task being processed (Pinedo. 2008).

### 2.3 Particle Swarm Optimization Approach

The particle swarm optimization (PSO) algorithm was introduced by Kennedy and Eberhart in 1995 as a novel optimization technique. Their primary research objective was to simulate the social behavior of bird and fish flocks (Kennedy & Eberhart, 1995). PSO is a global optimization method that can be applied to problems where the solution is a point or surface in an n-dimensional space. In such a space, initial assumptions are made, and the particles' initial velocity is assigned. Additionally, communication channels between the particles are considered. These particles then move in the solution space, and the resulting outcomes are evaluated based on a fitness criterion after each time step. Over time, the particles accelerate towards those with higher fitness values within the same communication group. The main advantage of this method over other optimization strategies is that the large number of swarming particles makes the technique more resilient to the problem of local optima (Kennedy & Eberhart, 1995).

## 2.4 Previous Work

During the 1970s to the mid-1980s, the complexity of scheduling problems gained significant attention. Subsequent research revealed that many instances solved in the 1950s were classified as NP-hard when generalized. Tavakkoli Mogaddam et al. (2005) investigated static multiprocessor systems, considering maintenance costs early and tardy completion penalties for interdependent tasks while disregarding communication costs and inter-task delays. They employed a hybrid genetic algorithm and tabu search to obtain optimal solutions. Haouari et al. (2006) cataloged a collection of 320 NP-hard scheduling problems. Due to the limitations of exact methods, approximation algorithms became the preferred approach. Systematic research on the job shop scheduling problem was first conducted by Johnson (1954), widely considered a pioneer in scheduling theory, and provided an optimal algorithm for a two-machine problem. Joulaei et al. (2006) focused on minimizing tardiness costs associated with task completion times without considering early completion penalties. They introduced a condition based on the task's start time, considering maintenance costs and priorities. Parsa et al. (2007), working with static homogeneous systems, introduced the shortest execution path parameter to minimize communication costs in scheduling. They also employed a genetic algorithm to find optimal solutions. Abdyazdan et al. (2007) employed a genetic algorithm with a population-based search to minimize the overall execution time in a homogeneous system where tasks have priorities and dependencies, and each task is executed on a dedicated machine. Javani et al. (2010) studied response time minimization in static homogeneous systems with task priorities. They used a hybrid genetic and memetic algorithm to solve this problem. Pezzella et al. (2008) employed a hybrid genetic algorithm to address the job shop scheduling problem. This approach was tested on 43 standard benchmarks and compared with 12 other methods. Computational results demonstrated the superior performance of the proposed algorithm. Naderi et al. (2009) minimized the maximum production time using simulated annealing and genetic algorithms. Arianjad et al. (2010) explored task scheduling based on task precedence and priorities and calculated holding, earliness, and tardiness costs in static heterogeneous systems. They considered the critical path in task execution to find the optimal solution for minimizing costs. Gholipour Kanani et al. (2011) proposed a model incorporating a new parameter and solved it using a genetic algorithm. They considered system exclusivity and machine heterogeneity in the scheduling problem. Garshasbi (2012) employed a genetic algorithm to minimize the total execution time in heterogeneous multiprocessor systems, considering task priorities and system exclusivity. Rahmati and Zandiyeh (2012) proposed a multi-objective algorithm to minimize total energy cost to solve the job shop scheduling problem. Wang and Uzsoy (2012) employed a genetic algorithm to minimize the maximum tardiness in production scheduling models. They aimed to propose a method that could reduce the solution time by decreasing the length of production scheduling sequences. Naseri et al. (2013) investigated the scheduling problem in static heterogeneous systems without considering task dependencies, task priorities, communication costs, time-dependent tardiness costs, and machine unavailability. Namazi and Golmakani (2013) addressed the problem of power consumption optimization. They investigated the scheduling problem in dynamic heterogeneous systems, disregarding task priorities and dependencies. They analyzed the execution cost of each task with machine access at any given time, assuming 100% efficiency. Mattfeld and Bierwirth (2014) proposed a genetic algorithm for job shop scheduling to minimize total tardiness. Ak and Koc (2015) stated that classical methods cannot be used to solve parallel machine scheduling and flexible job shop scheduling problems, and they employed genetic algorithms to solve these combinatorial problems. Mousavi et al. (2018) introduced a bi-objective model that considers machine setup times, learning, and training times to optimize makespan and tardiness. Mousavipoor et al. (2019) developed an integer linear programming model to solve the job shop scheduling problem, incorporating the effects of learning and flexible maintenance, and solved it using the GAMS software. Ayough and Khorshidvand (2019) proposed a new model for solving the cell formation problem. They developed an algorithm based on Simulated Annealing(SA) and Particle Swarm Optimization(PSO) to solve the proposed model. Roohnavazfar et al. (2021) focused on the stochastic single-machine scheduling problem and compared the results with a deterministic approximation approach. Rezvan et al. (2021) proposed a heuristic algorithm for job scheduling on parallel machines considering the sequence of operations. Statsny et al. (2021) used a graph-based method to solve the job shop scheduling problem. In their comprehensive review paper, Geurtsen et al. (2023) studied 250 articles on maintenance and resource scheduling and classified the single and parallel scheduling problems. Liu et al. (2023) developed a deep-learning technique for parallel machine scheduling in a dynamic environment. In their research, Ayough and Khorshidvand (2023) considered the assignment problem of the heterogeneous workforce to a U-shaped assembly line with uncertain processing time. They proposed a nonlinear programming model for solving the problem. Ying et al. (2024) used the Main Path Analysis (MPA) method to review the scheduling of semiconductor production operations. Ayough et al. (2024) proposed a nonlinear mixed integer programming model for a critical chain project scheduling problem and developed a genetic algorithm for solving the proposed model.

## 3. The Proposed Method

The proposed mathematical model is described in detail in the following section.

### 3.1 The Model Assumptions

The problem considered in this research is a specific instance of the flow shop scheduling problem. It has been customized with unique conditions to enhance its efficiency and realism compared to the general case. The model's assumptions are as follows:

- Job Decomposition: Each Job is composed of multiple activities(tasks). All constituent activities must be processed for a job to be completed. The number of tasks in each job varies.
- Single-Task Processing: A single machine can only process one activity at any given time.
- Activity Precedence: There exist precedence relationships between the activities within a job. An activity cannot commence until its prerequisite activities have been fully completed. There are no precedence relationships between jobs.
- Fixed Machine Capacity: The number of machines available to execute tasks is fixed. Each machine can process only one activity at a time.
- Non-Preemptive and Non-Delegable Processing: Machines cannot interrupt and resume an executing activity later. Once a machine has undertaken the processing of an activity, it cannot delegate this task to another machine.
- Once a machine completes the execution time of an activity from a job and is ready to execute another activity from the same job, the machine must undergo a corresponding setup time before implementing the new activity. The setup time for each machine's first activity is zero. Additionally, no setup time is considered for activities of two different jobs.
- Jobs and machines have types. Machines can only be responsible for processing the activities of a job if they are of the same type. Two types of machines and jobs are considered. A separate type is not defined for tasks.
- The activities of a job have priorities. Activities with higher priorities must be processed sooner. Moreover, the priority between activities for two different jobs is not defined.
- Each machine must incur a cost (such as depreciation) for performing the activities of a job. Costs are fixed and predefined. No fee is considered for setup times.
- While a machine performs the last job process, other machines previously involved in different processes of the same job can start a new job. Machines cannot perform multiple tasks simultaneously.

### 3.2 Indices, Sets, Parameters, and Decision Variables

I: The set of all machines used, where i is the index of a machine and m is the total number of machines.

J: The set of all jobs considered, where j is the index of a job and n is the total number of jobs.

Pj: The set of processes(tasks) of job j, where l is the index of a process, and nj is the total number of processes for job j.

lj: Represents the lth process of job j. Furthermore, Klj denotes the type of process l, with values 1 and 4 assigned to the different process types. Each process can also include two values.

Dlji: Represents the execution time of the lth process of job j when processed by machine i.

Rlj'ji: Represents the time required to prepare machine i for the commencement of process l' of job j, following the process l's completion.

Ci: The total cost incurred on machine i, and Clji is the cost incurred on machine i by process l.

CostM: A fixed cost considered for the purchase and maintenance of each machine.

Ki: Indicates the type of machine. Values of one and four are considered for different machine types, and each machine can also have two values.

Olj: The priority of process l among other processes of job j. The considered values are positive integers.

STlj: The start time for processing the lth task of job j.

FTlj: The finish time for processing the lth task of job j.

When constructing a project schedule, it is crucial to identify and define the logical sequence of activities. The following parameter is additionally considered for prerequisite relationships:

$$\begin{cases} u_{l_j l'_j} = 1, & \text{If process } l \text{ is a predecessor for process l'} \\ u_{l_j l'_j} = 0, & \text{otherwise} \end{cases}$$

Considering the objective functions and constraints defined for the proposed model, the decision variables are as follows:

Xlji: Binary decision variables. It is one if the machine i performs the lth process of the jth job; otherwise, it is zero.

Xljl'ji: Binary decision variables. It is one if process l is processed by machine i before process l'; otherwise, it is zero.

STlj: A real positive number that represents the start time of the lth process within job j.

$$\text{Min } f_2 = \max\ C_i \qquad \forall\, i \in \text{I} \qquad (1)$$

$$C_i = \sum_{j=1}^{n} \sum_{\forall l_j \in P_j} x_{l_j i} * C_{l_j i} + CostM \qquad \forall\, l_j \in P_j\ , i \in I \qquad (2)$$

### 3.3 The objective functions

In the proposed model, two objective functions are defined. The first objective function is to minimize the maximum completion time of jobs, where the completion time of a job is defined as the maximum completion time of its constituent activities.

$$\text{Min } f_1 = \sum_{j=1}^{n} FT_j \qquad (3)$$

$$FT_j = \max\ FT_{l_j} \qquad \forall\, l_j \in P_j \qquad (4)$$

$$FT_{l_j} = x_{l_j i} * (ST_{l_j} + D_{l_j}) \qquad \forall\, l_j \in P_j\ , i \in I \qquad (5)$$

The second objective function is to minimize the maximum cost of executing all machine job activities. To accommodate the PSO algorithm, which operates on a single fitness function, the multiple objectives of the proposed model were aggregated into a unified objective. A direct summation is infeasible considering the disparate units and scales of the original objectives. The objective functions were linearly normalized and weighted to form a composite objective function to tackle this issue, as expressed in Equation 6.

$$f = w_1 f_1 + w_2 f_2 \qquad (6)$$

Where w1 and w2 are the weighting coefficients for each objective function, respectively, since both objective functions are to be minimized, the final objective function (f) will also be a minimization problem.

### 3.4 The Proposed Mathematical Model

The complete mathematical model is presented below.

Min f

s.t.

$$\sum_{i=1}^{m} x_{l_j} = 1 \qquad \forall\, \text{l} \qquad (7)$$

$$\sum_{i=1}^{m} \sum_{\substack{\forall l_j \in P_j \\ l_j \neq l'_j}} x_{l_j i} = 1 \qquad \forall \text{ i, j, t} \qquad (8)$$

$$x_{l_j l'_j i} * \left(ST_{l'_j} - FT_{l_j}\right) \geq x_{l_j i} * x_{l_j i} * R_{l_j l'_j i} \qquad \forall\, l_j, l'_j \in P_j, l_j \neq l'_j, i \in I \qquad (9)$$

$$(O_{l_j} - O_{l'_j}) * (ST_{l_j} - ST_{l'_j}) \geq 0 \qquad \forall\, l_j, l'_j \in P_j \qquad (10)$$

$$u_{l_j l'_j} * \left(ST_{l'_j} - FT_{l_j}\right) \geq 0 \qquad \forall\, l_j, l'_j \in P_j \qquad (11)$$

$$min\left(x_{l_j i} * \left(1 - \left(\left|k_i - k_{l_j}\right|\right)\right), 0\right) \geq 0 \qquad \forall\, i \in I, l_j \in P_j\ \exists k_i, k_{l_j} \qquad (12)$$

$$x_{l_j i}, x_{l_j l'_j i}, k_i, k_{l_j}, ST_{l_j}, ST_{l'_j} \geq 0 \qquad (13)$$

$$O_{l_j}, O_{l'_j}, R_{l_j l'_j i}, u_{l_j l'_j} \geq 0 \qquad (14)$$

Constraint (4) minimizes the makespan, defined as the time required to complete all jobs. The makespan is determined by the job with the longest completion time. Equation (5) seeks to minimize the maximum cost of any machine. The load on a machine is calculated as the sum of the processing costs of all jobs assigned to it plus a fixed setup cost. Constraint (7) shows that machines must perform each job process. By jointly considering constraints (8) and (9), it can be demonstrated that a machine cannot perform more than one process at any given time. Constraint (9) claims that the machine must go through the preparation time to start a process. Constraint (10) shows that if two processes are priorities, they can ignore this constraint. Still, if the priority relationship for the two processes is not the same, the process's start time with the higher priority must be earlier than the other process. Constraint (11) shows that if a process has a prerequisite, its post-requirement process cannot be started until the prerequisite process is completed. Constraint (12) also emphasizes that there must be a type match between the job and the machine. Constraints (13) and (14) define the model's decision variables. Due to the complexity and nonlinear nature of the proposed model, deterministic methods are impractical for large-scale instances. Consequently, a Particle Swarm Optimization metaheuristic has been devised to tackle this challenge.

### 3.5 The Proposed PSO Approach

To solve the proposed model using the Particle Swarm Optimization algorithm, it is necessary to map the operators and concepts of this method to the variables and constraints of the mathematical model. Each particle in this problem represents a solution to the problem, and its fitness is calculated using Equation (3). In the proposed model, each particle is formed by initially creating a micro-particle for each job and machine. These sub-particles are constructed using a permutation of length equal to the number of jobs plus the number of machines minus one. Genes with values greater than the number of jobs act as separators between jobs. These separators indicate the assignment of jobs to machines. Figure (1) demonstrates how the nine tasks of a single job are assigned to five machines.

| Machine 1 | | | | Machine 2 | | | | Machine 3 | | Machine 4 | | Machine 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 7 | 11 | 4 | 8 | 2 | 13 | 9 | 5 | 10 | 12 | 6 |

**Figure 1**. The structure of a proposed micro-article for a sample job

As illustrated in Figure 1, red fields function as machine delimiters, clearly delineating the processes(tasks) associated with each machine. Upon creating micro-particles, they are concatenated to construct the primary particle. When two micro-particles are linked, a gene is introduced as an intermediary. The value attributed to these genes is a positive integer exceeding G, as determined by equation (15). These numerical values serve as task delimiters and do not influence the problem's solution. Figures (2) and (3) demonstrate this issue.

$$G = m + \text{Max}(|Pj|, \forall j) \quad (15)$$

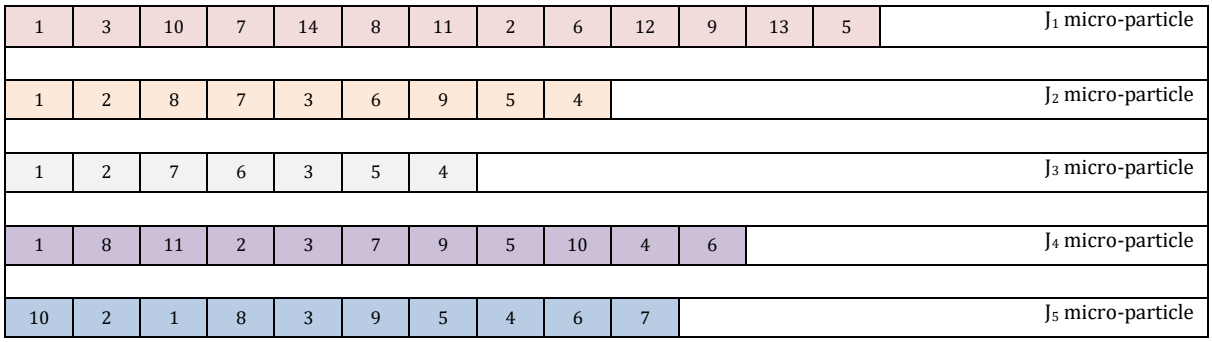



**Figure 2.** Constructed micro-particles for five jobs

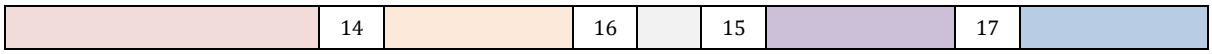



**Figure 3**. Concatenating the micro-particles to form the primary particle

### 3.6 Evaluating the Fitness Function

The objective function is the most crucial component of an optimization algorithm. When solving constrained optimization problems using metaheuristic algorithms, two primary approaches are considered:

1. Defining problem particles in a way that naturally satisfies the constraints. It is essential to note that this definition should not restrict the search space, as this would hinder the algorithm's performance.

2. Defining a penalty function for constraint violations. If a solution falls outside the feasible region, a penalty is imposed proportional to the constraint violation. In minimization problems, this penalty is added to the objective function. The penalty value must be carefully calibrated. If it is too low, the algorithm may find solutions that optimize the objective function but violate constraints, which is undesirable. Conversely, if the penalty is too high, the algorithm may become overly focused on a specific region, leading to local optima. In this research, the second method was employed to evaluate the objective function, wherein the penalty function value is added to the objective function value according to Equation (16). The parameter α represents the influence of the penalty on the objective function value and is often tuned through trial and error; in this study, its value is set to 5.

$$f_{new} = f_{old} + \alpha * violation \qquad \alpha > 0 \quad (16)$$

## 4. The Computational Results

The proposed model was tested with three scenarios involving varying numbers of machines and tasks. Their constituent processes and hierarchical dependencies characterized tasks. The model was evaluated under conditions of resource constraints and task complexity.

### 4.1 The Simulation Scenarios

In the first example, the maximum number of machines available to perform tasks is limited to five. This implies a machine might be idle and not assigned to any task. While this scenario is infrequent due to the potential for increased costs and time, it could be considered if the procurement and maintenance costs of the machine are prohibitively high and the algorithm determines that reducing the number of machines would enhance the objective function. A random processing time is generated uniformly within [10 .. 60] time units to execute each task on a machine and stored in matrix P. For instance, matrix P would be a 9×5 matrix in the first job comprising nine processes. The element at position (3,2) in this matrix represents the processing time of the second process on the third machine. The same structure is also considered for the costs incurred by the user in the job process. Another feature considered for each job, according to the model, is the idle time between processes. The structure for storing these times is a square matrix whose size is equal to the number of processes, repeated as many times as the number of machines, such that the entry (2,3,4) represents the idle time between processes two and three on machine four. Among other features considered for each job is the priority number of each task. Identical priorities will have the same number. Tasks with higher priority will be assigned a larger number. The last feature considered for jobs is the job type. For this purpose, two job types are assumed in the proposed model, and two different types are

considered for the machines. This classification for jobs and machines ensures that machines perform jobs they are capable of, and the type of job they cannot perform is not included in the execution cycle.

### 4.2 The Simulation Parameters

The proposed model was implemented in MATLAB on a Windows 10 operating system, utilizing a computer with a Core i3 2100 MHz processor and 8 GB of RAM. Values for other parameters are provided in Table 1 that were selected based on observations of the Pareto front. If the Pareto region exhibits changes in the final iterations, the number of iterations increases; otherwise, the selected number is considered appropriate or reducible. Considering the aforementioned points and through multiple algorithm executions, this iteration count was determined via trial and error. The population size was adjusted using a similar approach. By recording the start, end, and execution times of processes, it is possible to select an arbitrary member from the Pareto region in each iteration and plot its corresponding Gantt chart.

**Table 1**. The proposed PSO parameters

| Parameter | Value | | |
|---|---|---|---|
| | Example 1 | Example 2 | Example 3 |
| No. of Initial Particles | 50 | 75 | 150 |
| No. of Iterations | 200 | 250 | 300 |
| Inercia (w) | 0.85 | 0.90 | 0.97 |
| C1 | 2 | 2 | 2 |
| C2 | 2 | 2 | 2 |
| Minimum Velocity | 0.5 | 0.6 | 0.8 |
| Maximum Velocity | -0.5 | -0.6 | -0.8 |

### 4.3 The Simulation Results

Figure 4 presents a Gantt chart illustrating the task execution of the first example. Rectangular bars denote individual tasks in the Gantt chart associated with an arbitrary Pareto-optimal solution. Tasks allocated to the same row are assigned to a single machine for execution. The first example utilizes five machines, as evident from the number of rows in the Gantt chart. Vertical lines demarcate the completion times of tasks. Expressly, the initial vertical line at time 222 signifies completing the first process within the first job. Figure 5 presents the Gantt chart illustrating the cost implications of this scenario.

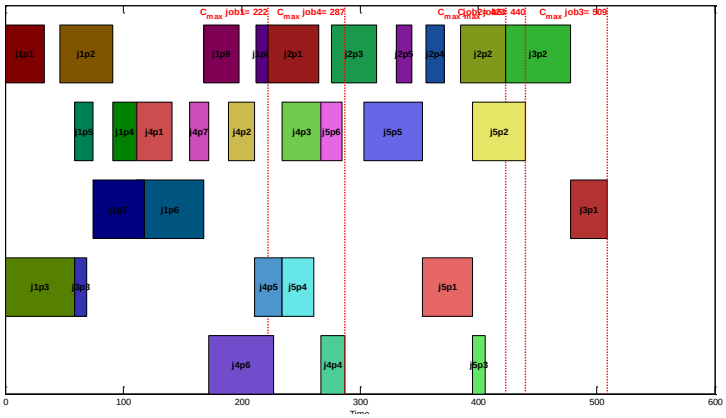



**Figure 4**. Temporal representation of tasks using a Gantt chart in Example 1

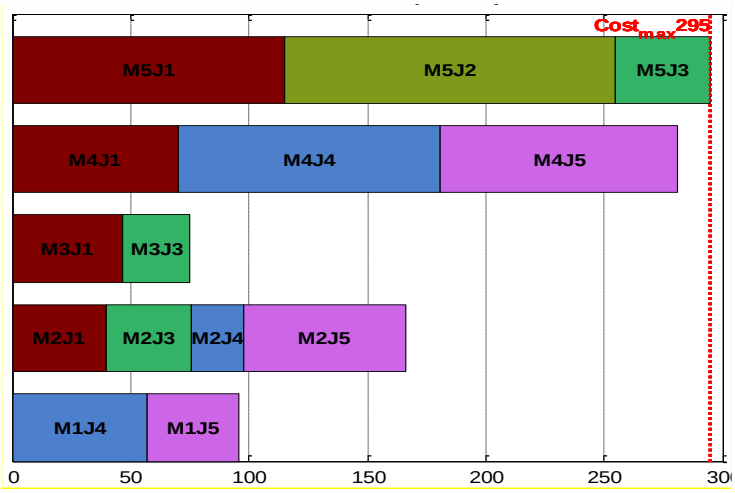



**Figure 5**. Gantt chart illustrating the cost breakdown of activities in Example 1

When visualized in terms of cost, the Gantt chart illustrates allocating tasks to different machines. For instance, the rectangles in the third row represent tasks processed by the third machine. Tasks 1 and 3 are assigned to this machine. There can be several reasons for a machine to be idle: (1) a mismatch between the task type and the machine's capabilities (e.g., a type A task assigned to a type B machine), (2) a prohibitively high cost of processing the task on that particular machine, or (3) incomplete optimization in the early stages of the algorithm. If processing time, setup cost, and task-machine compatibility constraints are relaxed, the optimal solution would trivially involve assigning all tasks to all machines. During the initial stages of optimization, the algorithm may produce infeasible solutions. A penalty function is introduced to guide the search towards feasible regions. To address the multi-objective nature of the problem, a weighted sum approach is employed, transforming it into a single-objective equivalent. The goal is to identify the Pareto front, representing a set of non-dominated solutions that approximate the optimal trade-offs among the objectives. At each iteration of the algorithm, a Pareto front is constructed based on the current population. No other solution dominates the members of this front. Due to the concept of dominance, additional regions not represented in the visualization may exist. If the population of the Pareto front is smaller than the overall population at a given iteration, then individuals for the subsequent generation are selected from these other regions. The selection mechanism for these additional individuals mirrors that of the Pareto front. This phenomenon is common in the early stages of the algorithm as the solution space is still being explored. Figure 6 illustrates the Pareto front of the first example before the objective functions are aggregated. Considering the problem's constraints, two types of Pareto fronts exist. In the first case, the population members in this region include any solution. Solutions in this region may not satisfy the type-matching constraints. This situation is depicted in Figure 7(a). The resulting Pareto region fulfills all model conditions by eliminating these solutions, as shown in Figure 7 (b).

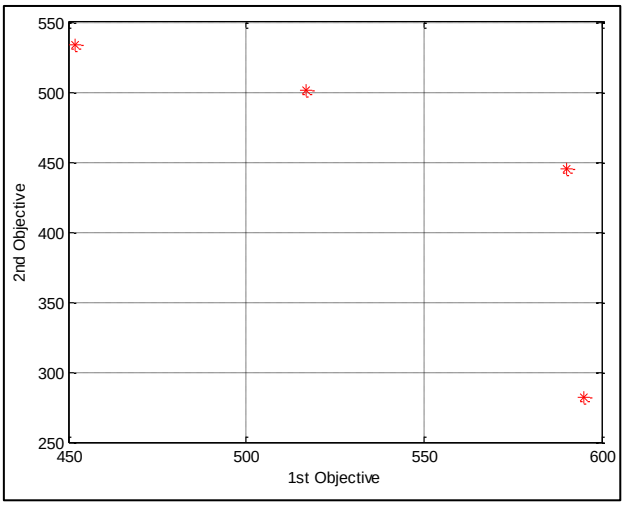
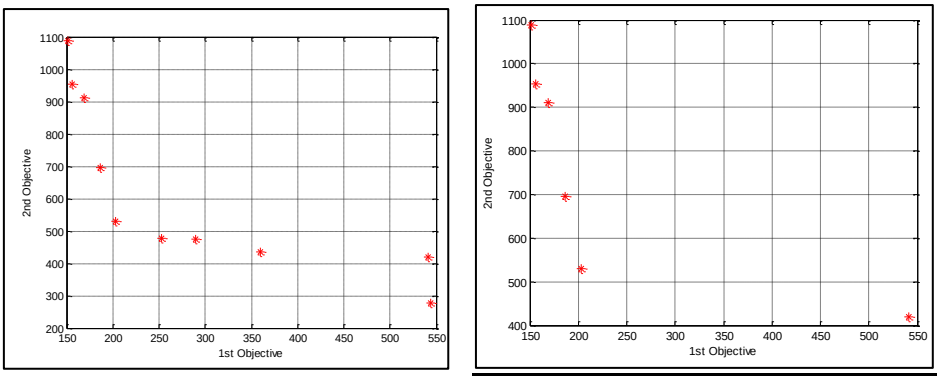


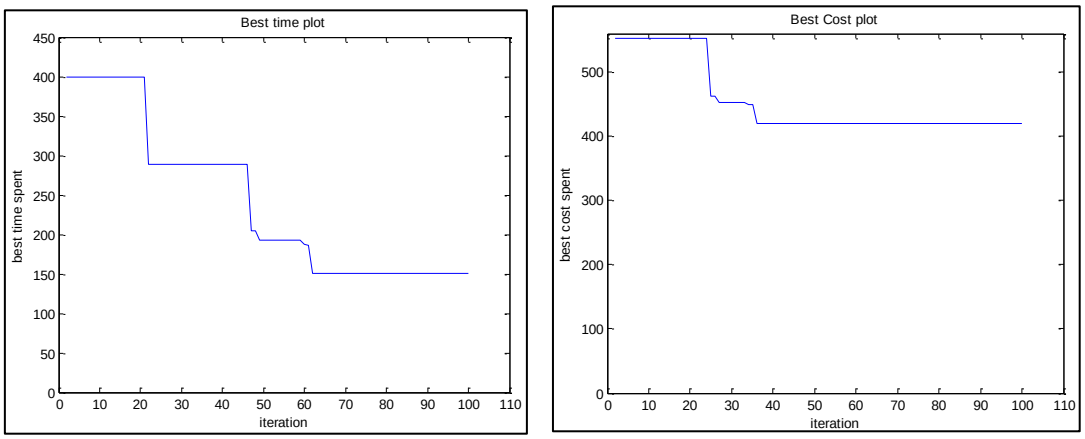

**Figure 6.** The Pareto frontier generated by the algorithm in the early stages of Example 1




(a) (b)

**Figure 7**. (a) The final Pareto frontier includes feasible and infeasible solutions, (b) eliminating dominated solutions and reconstructing the Pareto frontier.

Given the sole objective of minimizing either task execution time or machine costs, the optimal solution is chosen accordingly. In the case of time minimization, the solution with the shortest overall execution time is selected. The solution incurring the lowest total machine cost is chosen for cost minimization. Employing this approach, two Pareto-optimal solutions corresponding to minimal time and cost are identified and retained in each iteration. Based on these stored solutions, two graphs are constructed. The horizontal axis of these graphs represents the iteration number, while the vertical axis signifies either the average machine cost or the maximum task completion time. These graphical representations are depicted in Figure 8 (a) and (b), respectively.




(a) (b)

**Figure 8**. (a) emporal convergence diagram of tasks optimized within the Pareto frontier. (b) Cost convergence diagram associated with the Pareto-efficient frontier

According to the figures above, it can be inferred that the algorithm has attempted to minimize the objective function values. Stabilizing the best execution time and the lowest average cost in the final iterations indicates that the chosen value for the number of population iterations was appropriate. The figures mentioned above result from selecting two specific members from the Pareto front region at each stage. In most cases, these two particular points are not adopted as the solution, as one solution excessively minimizes the time value. At the same time, the other prioritizes reducing the average machine costs. Considering the weights the management assigns to the importance of objective function values for the project, a point from the solution space is selected. Assuming equal significance for the objective functions (w1= w2= 0.5), the convergence diagram of the objective function f for Example 1 is obtained, as shown in Figure 9.

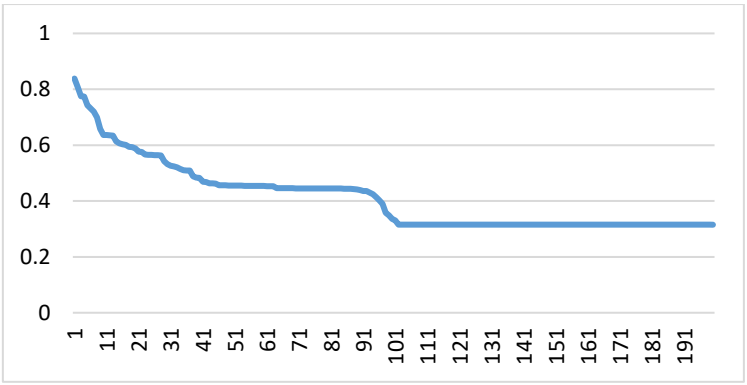



**Figure 9.** Convergence diagram reaching the final solution in example 3

Due to the stochastic nature of metaheuristic search methods, another crucial factor in evaluating the performance of such algorithms is their stability. This concept means that the solutions obtained from multiple runs of the algorithm should be close to each other and exhibit a low standard deviation. Figure 10 illustrates the stability of the proposed method in 50 different runs of Example 3. According to this figure, the proposed method found a solution of 0.315 in 88% of cases. The variance of the obtained solutions is 0.001282, and their standard deviation is 0.035798. Therefore, it can be concluded that the proposed method exhibits good stability.

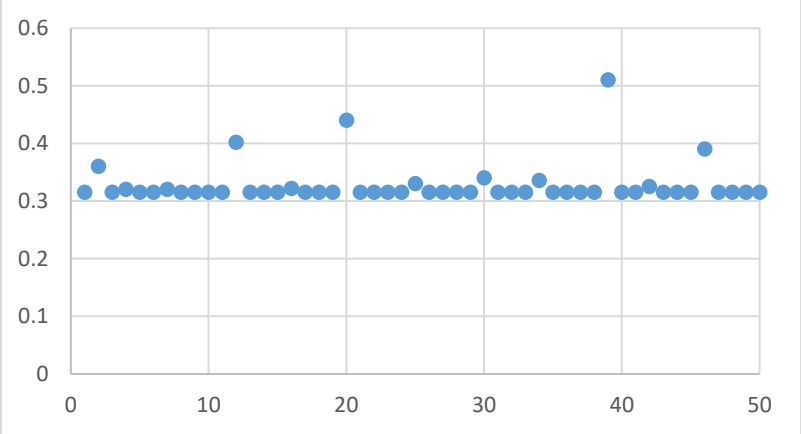



**Figure 10.** Stability diagram of the proposed method in 50 different runs of Example 3

**Table 2.** displays the results obtained from solving examples 1, 2, and 3 by implementing the suggested approach.

**Table 2**. The simulation results of all scenarios

| Scenario | f1 | f2 | f | Avg. Computation Time (sec.) |
|---|---|---|---|---|
| Example 1 | 150 | 418 | 0.292 | 15 |
| Example 2 | 236 | 566 | 0.294 | 25 |
| Example 3 | 408 | 741 | 0.315 | 44 |

## 5. Conclusions

This research focuses on the job scheduling problem, which involves determining the start time of each job on the most suitable machine. It is assumed that each job consists of several tasks of different priorities. Additionally, there are precedence relationships between the processes within a job, which a set of parallel machines must execute. These machines are heterogeneous, and some may be unable to process specific tasks. This study aims to optimize two conflicting objectives simultaneously: minimizing the overall completion time of all jobs and minimizing the operational cost of using the machines. Finding an optimal schedule is challenging, given the constraints of priority, precedence, and process-machine compatibility. A two-objective mathematical model has been

developed to address this problem. Considering all the problem constraints, the proposed model provides solutions that minimize both the execution time and the cost of executing the jobs.

Due to the model's non-linearity and the proposed method's computational complexity, a particle swarm optimization algorithm has been developed and implemented in MATLAB. The algorithm was applied to three benchmark instances of different sizes to evaluate the algorithm's performance. The computational results and stability diagrams demonstrate the efficiency and robustness of the proposed model. The limitations of the proposed model in capturing the complexities of real-world, industrial parallel machine scheduling problems highlight the need for further research. To bridge this gap, future studies should focus on:

- Grounding the problem in industrial case studies
- Employing empirical data to validate theoretical findings
- Comparing and contrasting various solution methodologies to identify the most effective approach

Moreover, the following limitations can be highlighted due to the model's constraints. The proposed model assumes a strict sequential order of tasks, meaning each task must commence after the preceding one is completed. This assumption might not hold in all industrial contexts. A more generalized model could relax this constraint to accommodate flexible task sequencing. Additionally, the model simplifies machine setup times by considering only inter-process setup. Machine setup times often occur between tasks and can be sequence-dependent. A more comprehensive model would incorporate variable machine setup times, increasing its applicability to various industrial scenarios. The proposed model assumes a homogeneous process type for all tasks within a job, requiring machines to have matching process capabilities. This assumption can be relaxed to accommodate heterogeneous process types within a job, enhancing the model's realism. Additionally, the model does not account for job arrival times, limiting the range of potential objective functions. Future research can explore more flexible objective functions, such as early and tardy penalty costs, to address these limitations.

**References**

Abdyazdan, M., Rahmani, A. (2007). Task scheduling in multiprocessor systems using a new priority genetic algorithm based on the number of children. 13th Annual Conference of Computer Society of Iran. (in Persian)

Adler, D. (1993). Genetic algorithms and simulated annealing: A marriage proposal. IEEE International Conference on Neural Networks, 2, 1104-1109, DOI: 10.1109/ICNN.1993.298712.

Ak, B., Koc, E. (2012). A Guide for Genetic Algorithm Based on Parallel Machine Scheduling and Flexible Job-Shop Scheduling. Procedia-Social and Behavioral Sciences, 62, 817-823.

Aggoune, R., (2004). Minimizing the Makespan for the Flow Shop Scheduling Problem with Availability Constraints, European Journal of Operational Research, 153(3), 534–543.

Ayough, A., Khorshidvand, B. (2019). Designing a manufacturing cell system by assigning workforce. Journal of Industrial Engineering and Management, 12(1), 13-26. doi: https://doi.org/10.3926/jiem.2547

Ayough, A., Khorshidvand, B. (2023). Robust optimization for the integrated worker-cell assignment and sequencing problem in a lean U-shaped assembly line, Computers & Industrial Engineering, 178, 109139, https://doi.org/10.1016/j.cie.2023.109139.

Ayough, A., Rabieh, M., Javadi, M.N., and Khoshidvand, B. (2024). An MINLP model for project scheduling with feeding buffer, International Journal of Applied Decision Sciences, 17(6), 710-732.

Garey, H.M., Johnson, D., Sethi, R. (1976). The complexity of flow shop and job shop scheduling, Mathematics of Operation Research, 1, 117-129.

Garshasbi, M. (2012). Optimization of task scheduling on parallel multiprocessor systems using genetic algorithms. 2nd Lahijan National Conference on Software Engineering. (in Persian)

Gholipour Kanani, Y., Tavakkoli Moghaddam, R., Tabari, M., Jafari Zarandini, Y., & Aryanezhad, M. B. (2011). Solving a Novel Multi-Objective Scheduling Problem in a Cellular Manufacturing System by a Hybrid Algorithm. Research in Production and Operations Management, 2(2), 1-18. (in Persian)

Hao, G., Sam, K., Baojie F., Ran W. (2014). A Hybrid Particle-Swarm Tabu Search Algorithm for Solving Job Shop Scheduling Problems, Industrial Informatics, IEEE Transactions, on, 10(4), 2044 – 2054.

Haouari, M., Gharbi, A., Jemmali, M. (2006). Tight bounds for the identical parallel machine scheduling problem. International Transactions in Operational Research, 13(6), 529-548.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks, 4, 1942-1948.

Javani, M., Fardad, Kh., Medadian, M. (2010) Scheduling Static Multiprocessor Systems with Multi-Agent Memetic Algorithm. 4th Iran Data Mining Conference (IDMC), Tehran. (in Persian)

Joulaei, F., Tavakoli Moghadam, R., & Azadian, F.. (2006). Parallel Machine Scheduling For Split Jobs With Sequence Dependent Setup Times. Journal Of Faculty Of Engineering (University Of Tehran), 40(4 (98)), 495-506. (in Persian)

Mattfeld, D., Bierwirth, C. (2014). An efficient genetic algorithm for job shop scheduling with tardiness objectives, European Journal of Operational Research, 155, 616–630.

Mousavi, S., Mahdavi, I., Rezaeian, J., Zandieh, M. (2018). Bi-objective scheduling for the re-entrant hybrid flow shop with learning effect and setup times. Scientia Iranica, 25(4), 2233-2253.

Mousavipoor, H., Farughi, H., Ahmadizar, F. (2019). Job shop scheduling problem based on learning effects, flexible maintenance activities and transportation times. Journal of Industrial and Systems Engineering, 12(3), 107-119.

Naderi, B., Zandieh, M., Ghomi, S.F. (2009). Scheduling sequence-dependent setup time job shops with preventive maintenance, The International Journal of Advanced Manufacturing Technology, 43(1-2), 170-18,

Namazi, A., Golmakani, H. (2013). Multiple Route Job Shop Scheduling with Maintenance Activity Constraints. International Journal of Industrial Engineering and Production Management. 23(4), 459-470.

Naseri, H., Khalili, F., Taghinezhad, N., Taleshian, F. (2013). Modeling the open workshop scheduling problem considering machine downtime, parameter availability, and time frame, Journal of Operations Research and Applications. 2, 131-143. (in Persian)

Parsa, S., Izadkhah, H., Hosseinzade,A. (2007). Combining object migration algorithm and genetic algorithm for task graph scheduling in multiprocessor architecture. 3rd International Conference on Information and Knowledge Technology, Tehran, (in Persian)

Pezzella, F., Morganti, G., Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem, Computers and Operations Research. 35(10), 3202-3212.

Pinedo. M.L. (2008). Scheduling, theory, algorithms, and systems. Springer, 3rd edition.

Rahmati, S. H. A., Zandieh, M. (2012). Developing two multi-objective algorithms for solving multi-objective flexible job shop scheduling problem considering total consumed power per month. Industrial Management Studies, 10(27), 118-143.

Roohnavazfar, M., Manerba, D., Fotio Tiotsop, L. (2021). Stochastic single-machine scheduling problem as a multi-stage dynamic random decision process. Computational Management Science, 18, 267–297.

Statsny, J., Skorpil, V., Balogh, Z., Klein, R., (2021). Job Shop Scheduling Problem Optimization by Means of Graph-Based Algorithm, Applied Science, 11.

Tavakkoli Moghaddam, R., Khodadagan, Y., Haghnevis, M. (2005). Presenting a combinatorial model for selecting parallel machines and scheduling jobs considering setup times. 4th International Industrial Engineering Conference, Tehran. (in Persian)

Wang, C.S., Uzsoy, R. (2012). A genetic algorithm to minimize maximum lateness on a batch processing machine, Computer and Operations Research, 9(12), 1014-1021.

Geurtsen, M., Didden, J., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review," European Journal of Operational Research., 305(2), 501-529.

Ying, K.C., Pourhejazy, P., Huang, Z.Y, (2024). Revisiting the development trajectory of parallel machine scheduling, Computers & Operations Research, 168, 106709.

Liu, C.L., Tseng, C.J., Huang, T.H., Wang, J.W. (2023). Dynamic Parallel Machine Scheduling With Deep Q-Network." IEEE Explore, 53(11), 6792-6804.