Journal of Quality Engineering and Production Optimization

Vol. 6, No. 2, Summer & Autumn 2021

http://jqepo.shahed.ac.ir

Research Paper

DOI: 10.22070/JQEPO.2021.14612.1191

Minimizing the sum of earliness and tardiness in single-machine scheduling

Marjan Esmaeili [1], Fardin Ahmadizar [2*], Heibatolah Sadeghi [3]

[1] *M.S. in Industrial Engineering, University of Kurdistan, Sanandaj, Iran*

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

[3] *Assistant Professor, Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran*

**** Corresponding Author:*** *Fardin Ahmadizar (Email: f.ahmadizar@uok.ac.ir)*

Abstract *– Today, the concept of JIT production has usage in production management and inventory control widely. In such an environment, tardiness or earliness is essential. Therefore, scheduling tries to minimize the sum of earliness and tardiness, which represents customer satisfaction, as well as inventory control. Most studies in scheduling adopt the assumption that machines are continuously available during the planning horizon. But in the real world, some machines may be temporarily unavailable for reasons such as breakdowns or preventive maintenance activities. So, considering the unavailability as a constraint is necessary for scheduling problems in the JIT production system. In this study, the unavailability constraint has been investigated with two flexible modes on a single machine. In each period, the duration of unavailability corresponding to the continuous working time of the machine changes in a discrete manner and can adopt two different values. Since the objective function is irregular, unforced idleness may be useful, increasing the complexity of the problem. First, a binary integer mathematical programming model is presented. Due to the NP-Hardness of the problem under consideration, a genetic algorithm is proposed to solve the problem in large dimensions. To examine the performance of the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), several problem instances are generated and solved, and the obtained results are compared with those obtained from solving the mathematical model with the GAMS software. The computational results indicate the proposed algorithm has a good performance with an average deviation of 0.87% and a reasonable computational time.*

Keywords– *Single machine scheduling, Earliness and Tardiness, Flexible periodic availability constraints, Genetic Algorithm.*

I. INTRODUCTION

Nowadays, in a competitive global market, having effective scheduling industrial advances is necessary. Scheduling models with earliness and tardiness metrics have a close relationship with JIT production and supply chain concepts in production systems. Determining the sequence and scheduling of jobs is a type of decision variable that plays an important role in industrial and service environments. Tardiness costs cause losing the customer and finishing the job early, causing maintenance costs, warehouse space, insurance and taxes, stagnant capital, and product corruption. Therefore, machines must be inspected and checked periodically to be imposed low cost to the system. Therefore, the issue of unavailability as a constraint in JIT production to minimize earliness and tardiness in the single machine environment, which is the most basic and in practice a special case of other environments, is essential in scheduling.

Manuscript Received:12- July -2021 & Revised:20- October -2021 & Accepted: 21- Nov -2021
ISSN: 2423-3781

Tsai (2007) considered minimizing the sum of earliness and tardiness in a single machine scheduling problem with distinct delivery time and setup time. They proposed a genetic algorithm for finding the best solution for the proposed model. Benmansour et al. (2014) examined minimizing the maximum earliness and tardiness weighted in the single machine scheduling problem with the periodic preventive maintenance periodic with normal and limited delivery dates. They used a CPLEX 12.4 for finding the solution of the proposed model. Ahmadizar and Farhadi (2015) presented a single machine scheduling problem in which jobs are published at different times and presented to the customer in batches. They provided a mathematical model for the minimized sum of earliness, tardiness, holding, and delivery costs. Lou et al. (2015) examined a single machine scheduling problem with variable maintenance activity; while maintenance should begin before a certain deadline and the maintenance period increases with the start time, preemption is not allowed. They proposed a heuristic algorithm in polynomial time for minimizing the maximum tardiness and makespan and the number of tardy jobs, and the total completion time. M'Hallah and Alhajraf (2016) considered a single machine scheduling problem with the objective function of minimizing the sum of weight earliness and tardiness. They used Ant Colony Algorithm (ACO), Approximation Solving (ASV), Variable Neighborhood Search (VNS) for solving the proposed model. Low et al. (2016) examined single machine scheduling for minimizing the sum of earliness and tardiness with common delivery date and the unavailability of distance and solve the problem. They proposed an integer linear programming model and used dynamic programming for solving this model. Sajadi et al. (2009) proposed a dynamic programming approach for lot sizing scheduling problems with periodic demand and finite horizon time. Niroomand et al. (2016) used a hybrid greedy algorithm for minimization tardiness/earliness in a single machine scheduling problem with a fuzzy delivery date.

Ganji et al. (2017) examined a single machine scheduling with minimizing maximum earliness and also minimizing the number of tardy jobs with a period of flexible unavailability with resumable jobs. They used a heuristic and branch-and-bound algorithm that was able to solve problems. Jayanthi and Anusuya (2017) proposed a PSO algorithm to minimize the sum of weight earliness and tardiness in single machine scheduling by considering the constraint of the size jobs. Touat et al. (2017) studied the problem of single machine scheduling with flexible availability constraints with the limited human resource with objective functions minimizing the total tardiness while preemption is not allowed and used a genetic algorithm for finding the optimal solution. Yuce et al. (2017) used a hybrid genetic-bee algorithm for minimizing earliness and tardiness penalties in the single-machine scheduling problem. Genetic algorithm operators have been used in the overall global search phase to increase bee algorithm (BA) search. Ahmadizar and Eteghadipour (2017) studied the two-agent single-machine scheduling problem with the objective function of minimizing the sum of earliness and tardiness. They used two greedy improvement algorithms to solve the proposed problem.

Xiong et al. (2018) considered a single-machine scheduling problem and due date assignment. The objective function in this paper includes the earliness, tardiness, and due date assignment costs. They used the pseudo-polynomial-time solution algorithms and randomized adaptive search algorithms to solve the proposed model. Pacheco et al. (2018) used a heuristic method based on the Variable Neighborhood Search (VNS) for sequencing jobs in a single machine with programmed preventive maintenance and sequence-dependent set-up times. Sadeghi (2019) considered the single machine production planning with periodic demand, operation cost, and periodic order quantity policy. Lin et al. (2019) considered a single machine with a restrictive common due window to minimize the total weighted earliness-tardiness penalties, which conform to just-in-time (JIT) manufacturing. They used the simulated annealing (BSA) algorithm for solving the proposed model.

Kellerer et al. (2020) studied a single machine scheduling problem to minimize the total weighted earliness and tardiness about a nonrestrictive common due date. They presented a fully polynomial-time approximation scheme (FPTAS). Unavailability of the machine is one of the constraints that manufacturing systems always face. Mozaffariyan and Sahraeian (2020) studied single-machine scheduling with linear earliness and tardiness costs considering the work failure, energy consumption restriction, and the allowed idleness. They presented a nonlinear mathematical model and used a genetic algorithm for solving this problem. Khanh Van and Van Hop (2021) studied the scheduling problem of parallel machines with minimizing the sum of weight earliness and tardiness and the maximum completion time. They

proposed a genetic algorithm based on sequence-dependent preparation time. Sadeghi et al. (2021) applied a small lot delivery strategy through the JIT system for an integrated production-inventory model with multiple discrete deliveries. Chen et al. (2021) investigated the problem of single-machine scheduling with minimizing total tardiness, unavailability of the machine, and preventive maintenance constraints. They presented a genetic algorithm and showed the results of computational tests, the effectiveness and efficiency of the algorithm. Wang et al. (2021) investigate the single-machine common due-window assignment problem with generalized earliness/tardiness penalties and a rate modifying activity with two types of processing time for jobs, one with constant values and another with time-dependent processing times, also considering the constant maintenance activity and the goal of obtaining the optimal sequence. The Objective function and solution algorithms of existing research for single-machine scheduling are summarized in Table I.

Table I: Summary of the features of surveyed publications

Author	*Availability*	*Non resumable*	*Objective function*	*Approach*	*Denote*
Feldmann and Biskup (2003)	-	-	total earliness and tardiness penalties	Simulated Annealing	$1\|d_j = d\|\sum \alpha_j E_j + \beta_j T_j$
Mahnam et al. (2013)	-	-	the sum of maximum earliness and tardiness	Branch and bound-GA-PSO	$1\|rj,dj,I\| ET_{max}$
Ahmadizar and Farhadi (2015)	-	-	Sum of earliness, tardiness, holding, and delivery costs	Combined algorithm	$1\|sij$, B, $D\|\Sigma\alpha ijEij+\beta ijTij+hijHij+yibDj$
Lou et al. (2015)	Variable Maintenance	✓	tardiness and makespan, total completion time	Heuristic	$1\|nr\text{-}v\|L Lmax, \Sigma Tj, Cmax, \Sigma Cj$
Mashkani and Moslehi (2016)	flexible bimodal periodic	✓	the total completion time	Branch and bound	$1\|nr\text{-}fpa, bm\|\sum C_i$
M'Hallah and Alhajraf (2016)	-	-	Sum of weight earliness and tardiness	(ACO), (ASV), (VNS)	$1\|d_j\|\Sigma w_j E_j+\Sigma w_j T_j$
Shahriari et al. (2016)	Periodic preventive maintenance	✓	Total earliness and tardiness, and makespan	MOPSO	
Low et al. (2016)	-	-	the sum of earliness and tardiness	ILP	$1,h_1\|d_j = d \mid \sum(T_j + E_j)$
Jayanthi and Anusuya (2017)	-	-	the sum of weight earliness and tardiness	PSO	$1\|\|\Sigma(\alpha_j E_j+\beta_j T_j)$
Yuce et al. (2017)	-	-	earliness and tardiness penalties	GA- BA	$1\|\|\Sigma(\alpha Ej+\beta Tj)$
Ahmadizar and Eteghadipour (2017)	-	-	the sum of earliness and tardiness	Greedy Algorithms	$1\|\|\Sigma(Ej+Tj)$
Ganji et al. (2017)	Flexible activity	✓	maximum earliness	Heuristic	$1\|nr\text{-}fa\|E_{max}$
Touat et al. (2017)	Flexible Periodic Activities	✓	total tardiness	GA	$1\|nr\text{-}fpa,r_i,d_i\|\Sigma\tilde{T}_i$
Kellerer et al. (2020)	-	-	total weighted earliness-tardiness	(FPTAS)	$1\|dj = d, p(N) \le d\|$ $\sum w_j(E_j + T_j)$

Continue Table I: Summary of the features of surveyed publications

Author	*Availability*	*Non resumable*	*Objective function*	*Approach*	*Denote*
Mozaffariyan and Sahraeian (2020)	-	-	total earliness and tardiness penalties	GA	$1\|\| \sum(\alpha E_j+\beta T_j)$
Khanh Van and Van Hop (2021)	-	✓	total earliness and tardiness and makespan	GA-ISETP	$M\|\|\lambda Cmax + \sum(Ej+Tj)$
Chen et al. (2021)	Preventive maintenance	-	total tardiness	GA	$1\|$ PM $\|\sum T_i$
Wang et al. (2021)	Fixed maintenance	-	sums of earliness/tardiness penalties, the weighted number of early/delayed jobs	Heuristic	$1\|pj = aj, pj = \lambda jaj, CONW\|\sum (\alpha E_j + \beta T_j + \theta_j u_j + \delta_j v_j + \gamma d_1 + \omega D)$
This article	Flexible bimodal periodic	✓	Sum of earliness and tardiness	GA	$1\|fpa,bm\| \Sigma(E_j+T_j)$

This paper considers single-machine scheduling with the aim of minimizing the sum of earliness and tardiness with flexible periodic availability constraints. This paper assumed that the maximum time for jobs is fixed, and two different values are considered for it. Therefore, based on the mentioned cases, two different values for the maximum continuous working time and the period of unavailability have two different modes. These modes are called periodic dual-mode available restrictions. Tasks scheduled between two periods of unavailability are called categories. Due to the high complexity of the problem, the genetic optimization algorithm is used to solve the proposed model, and to find the efficiency of the genetic algorithm, minor problems are modeled and solved in GAMS software. The results are compared with the results of the genetic algorithm. Therefore, based on what has been said, the innovation of the paper can be summarized as follows.

1. It is considered a coordinated production system in which goods and services are delivered just in time when needed.
2. Reducing the costs of maintaining products in the production line and reducing defective products and reprocessing by having a stop in the production line.
3. Reduce delay times due to inaccessibility of the machine and stoppage of the production line
4. Use just in time strategy with production line stops
5. Controlling stops and preventing breakdowns due to a minor breakdown and reducing the number of equipment failures and unplanned expectations.

II. PROBLEM DESCRIPTION AND MODEL FORMULATION

This paper examines to minimize the sum of earliness and tardiness with flexible periodic availability constraints. In each period, the duration of the unavailability period changes according to the continuous working time of the machine in a discrete manner and can adopt two different values. The concept of flexibility is defined as the interval for unavailability period occurrence and the start time of unavailability in this interval is a decision variable. In this model, it is assumed that the maximum continuous working time of the machine and the duration of unavailability have fixed and predetermined values that the unavailability time is proportional to the continuous working time of the machine. A set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ is processed in a machine at zero times. During each period, the maximum continuous working time of the machine can adopt two values T_1 and T_2, which is $(T_2 > T_1)$, and the duration of each unavailability period, which depends on the maximum continuous working time of the machine, can adopt two values, take W_1 and W_2, which are $(W_2 > W_1)$. In each period, since the maximum continuous working time of the machine and the duration of unavailability have two different modes, these modes are called bimodal flexible periodic availability

constraints. The scheduled jobs between any two unavailability periods are called batches. The objective function is to minimize the sum of earliness and tardiness. The problem of a single machine scheduling with the flexible bimodal periodic availability constraints is shown in Figure 1. The values of q1 to q4 are the sum of the processing times of the scheduled jobs, which represent the first to fourth batches, respectively (Mashkani and Moslehi, 2016). Therefore, the problem of scheduling a single machine with the objective function of minimizing the sum of earliness and tardiness with the flexible bimodal periodic availability constraints in the condition which preemption is not allowed is represented by the symbol $1|fpa, bm|\sum_{i=1}^{n}(T_i + E_i)$, which in that bm shows the unavailability constraint of two-modes.

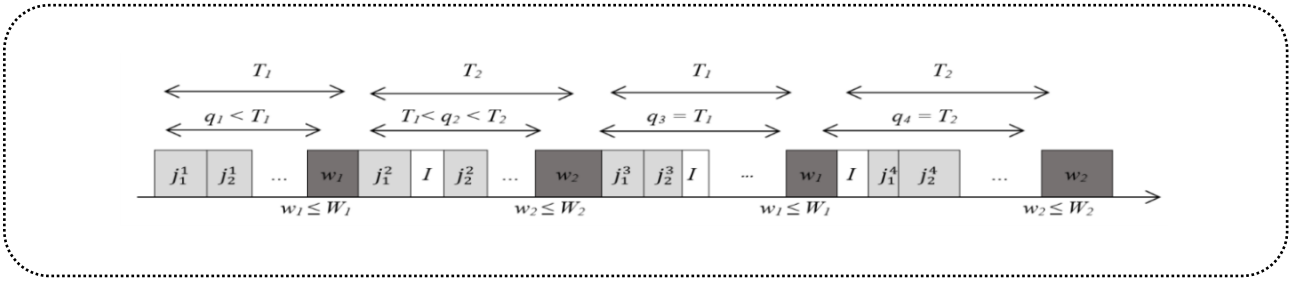

Fig.1. The problem of single machine scheduling with the flexible bimodal periodic availability constraints

It is possible flexible bimodal availability constraints to the flexible multiple-modes availability constraints $1|fpa, mm|\sum_{i=1}^{n} T_i + E_i$ generalized. In this case, it is assumed that during each period, the maximum continuous working time of the machine can take several values of T_p, that $T_{p+1} > T_p$, and the duration of each period of unavailability, which depends on the maximum continuous working time of the machine, can take the values W_p that $W_{p+1} > W_p$. So as the maximum working time of the machine increases so does the unavailability period. Since each period has a maximum continuous working time of the machine and a period of unavailability, there have several different modes, and these modes are called multi-mode flexible periodic availability constraints.

The assumptions made in this paper are as follows:

- Jobs are available in zero time.
- Only one machine is available for processing jobs.
- The machine can only process one job at any time.
- Preemption is not allowed from the start to the end of the machining process on the machine.
- The machine is not constantly available for processing during the schedule.
- Setup times are independent of sequence and are included in processing times.
- Each job has a determined due date.
- Tardies and earlies have punishable, and costs depend on the time of tardies and earlies.
- Unforced idleness may be useful.
- All data are deterministic.

Property 1. Problem $1|fpa,bm|\sum_{i=1}^{n}(T_i + E_i)$ is strongly NP-hard.

Researchers have long considered the computational complexity of the problem $1||\sum T_j$ until they proved the nonlinearity and complexity of this problem in 1989 (Du and Leung, 1990). Years later, Wan and Yuan (2013) proved that the problem of a single machine with minimizing the sum of earliness and tardiness with the symbol $1||\sum T_j + E_j$ is a strongly NP-hard problem. Since the Van and Yan problem is a special case of minimizing the sum of earliness and tardiness with flexible bimodal periodic availability constraints problem in the single machine environment with resumable jobs, then the complexity of the problem under study in this paper is With the addition of constraint at least that's the size of Van and Jan's problem. It can be concluded that problem $1|fpa, bm|\sum T_{i+}E_i \sum_{i=1}^{n}(T_i + E_i)$ is a strongly NP-hard problem.

Notations that are used in the model are listed as below:

In this section, the symbols and variables used in the problem are first defined, and then a linear mathematical model of the integer is presented.

Index

i Index for jobs, i =1,…, n.

k Index for batches, k=1… K.

P Index for unavailability, P=1,2.

Parameters

w_P The duration of the unavailability period

T_P The maximum continuous working time

p_i Processing time of job i

d_i Due date of job i

M A very large positive number.

Decision variables

s_k The starting time of kth batch

c_i The completion time of job i

T_i The tardiness of job i

E_i The earliness of job i

u_{ij} A binary variable equal to 1 If the job j in the sequence within the batch is before job i; otherwise, it is 0. i,j = 1,2,…,n and $j \neq i$

x_i^k A binary variable. If job i is scheduled in batch k, its value will be 1; otherwise, it is 0. i = 1,2,…,n and k = 1,2,…,K

y_p^k A binary variable equal to 1 if the maximum continuous working time of the machine in kth batch equals T_p; otherwise, it is 0. P = 1,2 and k = 1,2,…,K

According to the introduced symbols and the expressed assumptions, the mathematical model of the problem is formulated as follows.

$$\min Z = \sum_{i=1}^{n} (T_i + E_i) \tag{1}$$

s.t

$$\sum_{k=1}^{K} x_i^k = 1 \qquad i = 1.2.3. \ldots . n \tag{2}$$

$$\sum_{i=1}^{n} p_i x_i^k \leq \sum_{P=1}^{2} T_P y_P^k \qquad k = 1.2.3. \ldots . K \tag{3}$$

$$S_1 = 0 \tag{4}$$

$$S_k \geq \sum_{i=1}^{n} \sum_{r=1}^{k-1} p_i x_i^k + \sum_{r=1}^{k-1} \sum_{P=1}^{2} w_P y_P^r \qquad \forall\, k = 2.3. \ldots . K \tag{5}$$

$$S_k \leq S_{k+1} \qquad \forall\, k = 1.2.3. \ldots . K \tag{6}$$

$$C_i \geq S_k + p_i - M\left(1 - x_i^k\right) \qquad \forall\, k = 1.2.3. \ldots . K \qquad \forall i = 1.2.3. \ldots . n \tag{7}$$

$$C_i \leq S_{k+1} - \sum_{P=1}^{2} w_p y_p^k + M\left(1 - x_i^k\right) \qquad \forall\, k = 1.2.3. \ldots . K - 1 \qquad \forall\, i = 1.2.3. \ldots . n \tag{8}$$

$$C_i \leq C_j - p_j + M\left(u_{ij}\right) \qquad \forall\, j = 1.2.3. \ldots . n \qquad \forall\, i = 1.2.3. \ldots . n \tag{9}$$

$$C_j \leq C_i - p_i + M\left(1 - u_{ij}\right) \qquad \forall\, j = 1.2.3. \ldots . n \qquad \forall\, i = 1.2.3. \ldots . n \tag{10}$$

$$\sum_{P=1}^{2} y_i^k = 1 \qquad \forall\, k = 1.2.3. \ldots . K \tag{11}$$

$$T_i = \max\{0. d_i - C_i\} \qquad \forall\, i = 1.2.3. \ldots . n \tag{12}$$

$$E_i = \max\{0. C_i - d_i\} \qquad \forall\, i = 1.2.3. \ldots . n \tag{13}$$

$$x_i^k . y_p^k . u_{ij} \in \{0. 1\} \qquad P = 1.2 \qquad \forall\, k = 1.2.3. \ldots . K \qquad \forall\, i = 1.2.3. \ldots . n \tag{14}$$

Constraint (1) indicates the objective function of minimizing the sum of earliness and tardiness. Due to constraints (2), each job can only be scheduled in one batch. Due to constraint (3), the sum of processing times for scheduled in the kth batch is less than the maximum continuous working time of the machine in that batch. Constraint (4) indicates that the start time of the first batch is zero. Constraint (5) calculates the start time for the *k*th batch, where *k* = 2,3, ..., *K*, and constraint (6) prevent interference from jobs within batches. The set of constraints (7) and (8) calculate the completion time of job *i*, which is equal to the sum of the start times of the *k*th batch and the processing time of the *i* job, and ensures that the jobs do not interfere with the unavailability time. The set of constraints (9) and (10) prevents the interference of two jobs on the machine and determines the order of jobs in the sequence. In these constraints, *M* is a very large positive value. Constraints (11) The maximum continuous working time of the machine in each batch can have a certain amount. Constraints (12) and (13) calculate the earliness and tardiness times. Finally, constraint (14) indicates that the decision variables are binary.

In this model, the maximum number of *k* that represents the number of batches for *n* jobs is equal to *n*. The minimum job that can be scheduled in a batch is obtained by dividing the shortest time of work continuity time by the largest time of processing job. The maximum job that can be scheduled in a batch is equal to setting the processing time to ascending order and the number of jobs in which the total processing time is equal to or less than the largest continuous working time. The following formula is used to obtain *M*, which is a large positive value (Mashkani and Moslehi, 2016).

$$\mathrm{M} = \sum_{i=1}^{n} p_i + W_2 \times (n - 1) \tag{15}$$

The proposed model is solved by GAMS software using the CPLEX solver, and its accuracy has been verified. The GAMS is able of optimal solution for a small size, then for solving the problem in a larger size, the genetic algorithm is used, and the results are examined.

III. PROPOSED ALGORITHM APPROACH

Evolutionary algorithms are part of optimization algorithms, most of which are random and inspired by nature. These algorithms are suitable for searching in complex and very large spaces and are less likely to be trapped in local optimal. The genetic algorithm is a general method of metaheuristic for discrete optimization that is suitable for solving scheduling problems. This method was invented in 1975 by Holland. Genetic algorithms use Darwin's principles of natural selection to find the optimal formula for predicting or matching a pattern. This method is a kind of neighbor search method that uses genetic evolution as a problem-solving pattern. As shown, the problem of a single machine with the objective function minimizing the sum of earliness and tardiness with flexible bimodal periodic availability constraints is an NP-hard problem. The mathematical model is only able to solve the problem in small dimensions, so a metaheuristic is used to solve the problem in large dimensions. Here, the steps of the genetic algorithm are fully explored and used to optimize the problem.

A. Chromosome structure

The first step in the genetic algorithm is to show the answers to the problem in the form of a chromosome. Each chromosome is a string or sequence of bits that represents a feasible solution problem. In most cases, the initial answer is randomly generated, and in some cases, it is generated using a heuristic algorithm, and in this case, it is random. In this paper, the coding method is that there is a chromosome with two arrays, one of which is for displaying jobs and the other for displaying the type of unavailability. The first array that shows the sequence of jobs uses a set of non-repetitive natural numbers. Each number corresponds to a job specified by the job counter. The second array is the unavailability type. Since we have two types of unavailability, they are marked with the numbers 1 and 2, which respectively indicate the unavailability of the first and second types. Also, due to the jobs being categorized, and each job can only be placed in one batch, the upper limit for the number of the batch is considered the number of jobs. So the number of existing jobs is not unavailability. In this paper, the length of chromosomes is considered as the number of batches, which is equal to the number of jobs. Figure 2 shows an example of chromosome coding for six jobs.

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

Fig. 2. Display of chromosome coding for six jobs

The parameters for the six jobs to show the chromosome structure of the problem are given in Tables II and III. W_p, which is related to unavailability, is selected according to the T_p parameter, which is continuous working time.

Table II: Processing time and due date for six jobs

i	1	2	3	4	5	6
Processing time	5	6	5	8	1	3
Due date	21	12	6	18	12	20

Table III: Parameters T_p and W_p for six jobs

P	1	2
T_P	10	14
W_P	5	7

In each chromosome, information about a point in the input space is coded. To identify the point to which the chromosome refers, it must be decoded. The chromosome must be separated into constructive genes to decode, and then each gene must be decoded. Here, each gene represents a job, and since, based on one of the assumptions that all jobs are available at zero times, the first gene on the chromosome is scheduled at zero times as much as the processing time, and the next jobs in the chromosome are arranged in sequence. Then, according to the second dimension of the chromosome, which indicates the type of unavailability, the processing time of the jobs is summed from the first job until the total processing times are smaller or equal to a continuous working time proportional to unavailability shown in the first gene of the second dimension. The unavailability shown in the first gene is then placed in sequence and continued until all jobs have been performed, respectively. Figure 3 shows an example of chromosome decoding.

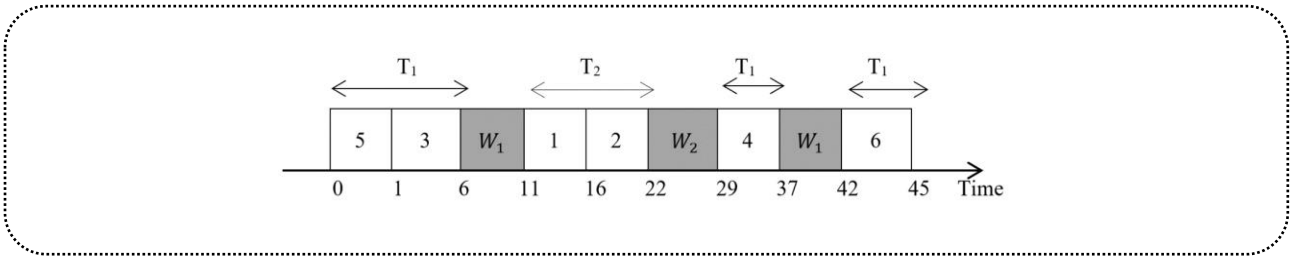

Fig. 3. display of chromosome decoding for six jobs

B. Production of the primary population

The second step in implementing the genetic algorithm is to generate a series of primary solutions called populations. One of the genetic algorithm characteristics is that instead of focusing on one point in the search space or one chromosome, it focuses on a population of chromosomes. Each population or generation of chromosomes is the same size. The primary population is produced only once at the beginning of the first generation of the genetic algorithm. Determining the appropriate population size is important in implementing a genetic algorithm. In this paper, the initial population includes sequences whose number of sequences is equal to the size of the population along each chromosome, which is equal to the number of jobs. In each sequence, inside each of the genes, a number representing the job counter is randomly placed. Depending on the type of unavailability, the jobs are categorized, each batch including a number of jobs whose sum of processing times must be commensurate with the type of unavailability provided in the second dimension of the chromosome. In other words, the sum of the processing times of the jobs within each batch should be less than or equal to continuous working time, which is proportional to the type of unavailability provided. Jobs are added in the order specified in the first dimension of the chromosome and so on until there is nothing to add. The objective function of the problem under study is an irregular objective function. Sometimes unforced idleness times may be useful for irregular functions and can improve the response to the problem. By stating a particular condition, it can be shown that intentional idleness time can improve the objective function. Consider that the nth job is the last job in the last batch. If the n job has earliness, the amount of the objective function can be improved by keeping the job idleness until it can be completed on time. Here, a proposed algorithm is used to apply inserted idleness to the genetic algorithm, which is listed below. Idleness algorithms are applied after obtaining a sequence of jobs and placing unavailability. In this way, tardy and early jobs are calculated, and from the end of the sequence, the number of tardy or on-time jobs and early jobs is counted. If the number of early jobs is greater than the number of tardy and on-time jobs, they are shifted forward to the size of the smallest number of early jobs. Because the idleness algorithm is applied from the end to the beginning of the sequence, the first applied idleness is kept as the lowest amount of idleness, which is measured each time with the new idleness amount, and the lowest is selected. When a job in the sequence has the conditions for applying newly inserted idleness, if idleness has been applied before, the idleness of the previous job is updated, that is, the amount of idleness entered from it is reduced. The setting of the parameters for the proposed genetic algorithm is given in Table IV.

Table IV: parameters for the genetic algorithm

parameters	value
Chromosome length	*N*
Population size	100
Probability of crossover	0.8
Probability of mutation	0.08
The number of repetitions	10**N*

The steps in the idleness algorithm are as follows.

C. Idleness algorithm

Step 0) Specify the sequence of jobs. Put the unavailable in the sequence. Consider the amount finish time to job consider be *f* and put it in a list called F. Then, put the number of tardinesses in T. Put the number of earliness jobs to E. Put idleness time amount in a list called I. Consider all the values in List I at first to be zero. Put the delivery time difference and finish time to *L*.

Step1) Consider the counter *i* equal to the last job. Set the value of T and E to zero. Put the idleness time amount in Id. Consider the smallest value of L equal to min. Then consider Id = -min. Put j = *i* + 1.

Step 2) Continue while *i* > = 0.

Step 3) If it was L > = 0, set T = T + 1. Put *i* = *i* - 1 and go to step two. Otherwise, go to the fifth step.

Step 4) If the list I in position *i* has idleness, then consider the value of *j* = *i*. Take Id equal to the idleness amount of position *i* in List I. Set the value of T and E to zero again. Put *i* = *i* - 1 and go to step two. Otherwise put *i* =*i* - 1.

Step 5) If it was L < 0, put E = E + 1. Then select the minimum value of Id from between the Id in List I and the value of *L*.

Step 6) If it was T < E, then add Id to the finish time of jobs from *i* to *j* (*f* = *f* + *Id*). Update F list. Put Id on the idleness I list, if they had idleness on the I list before, then subtract the new idleness amount from the previous idleness amount and update the I list. Then go to step one. Otherwise:

Step 6-1) If Id ≠ 0. Then go to the fourth step.

Step 6-2) If Id = 0. Then set *i* = *i* - 1 and go to step two.

D. Fitness objective function assignment

Whether the answer is appropriate or not is measured by the criterion obtained from the objective function—the more appropriate the answer, the greater the fitness value. In optimization problems, the fitness function is the same as the objective function used to evaluate and reproduce new chromosomes, called "offspring of later generations." Depending on the objective function, the fit of the chromosomes must be calculated by two parameters, the start time and the completion time of each job. By calculating these parameters, the tardiness and early of each job are obtained according to the relations (12) and (13) of the model. In the end, the objective function is calculated by summing the values of tardiness and earliness. The fitness function is obtained from the following equation.

$$Z_{co}^{'} = max\, Z_{CO} - Z_{CO} \qquad co = 1.2.\ldots.n \tag{16}$$

In this case, the objective is to minimize tardiness and earliness, and the fitness function is obtained by distinguishing each value of the objective function for each chromosome from the largest value of the objective function. In this formula, *Co* counts the chromosomes, Z'_{co} is the value of the fitness function. The value of max Z_{CO} is the largest value of the objective function among population chromosomes. Z_{CO} is also the value of the objective function for each of the population chromosomes.

E. Selection strategy

The choice is the process of selecting two parents for the act of intersection. To select more suitable parents for the production of children with high fitness. In this research, the roulette wheel method has been used, so the chromosomes of the population in the form of a sequence wrought, then the total value of each chromosome is calculated with the value of all previous chromosomes. Then the relative cumulative fitness value of the chromosomes is calculated. A random number α is created between zero and one. Among chromosomes, the first chromosome with a relative cumulative value greater than α is selected.

F. Genetic operators

After determining the suitability of the parent population chromosomes, the most suitable chromosomes are passed on to the next generation, produce the next generation, crossover, and mutation operators on the parents are performed as pairs from populations. This mechanism forms a new population of existing chromosomes and the size of the current population. The size of the population remains constant throughout the process.

G. Intersection operator

It is a process that involves two parents and produces a new child. A crossover operator is used on production ponds in the hope that a better child will be produced. This operation is done in three steps:

1. The reproductive operator selects a pair of parents from the reproductive pool.
2. A point of intersection is randomly selected along the string.
3. Finally, the values of the strings change according to the intersection point.

In this paper, a method similar to a two-point intersection is used, with the difference that one of the points is used instead of a two-point and does not use the one-point method, because that is the form it is common, which one point to be randomly determined that to the right of one parent and the left of another parent selected is not effective here, because repetition occurs. Here, after selecting the parents, a point is considered, from the left, means from zero to that point, from the first parent, and to continue the string to the size of the chromosome from the next parent from the first point examines everywhere, every gene that was not repetitive is added to the child. Figure 4 below shows the crossover operator.

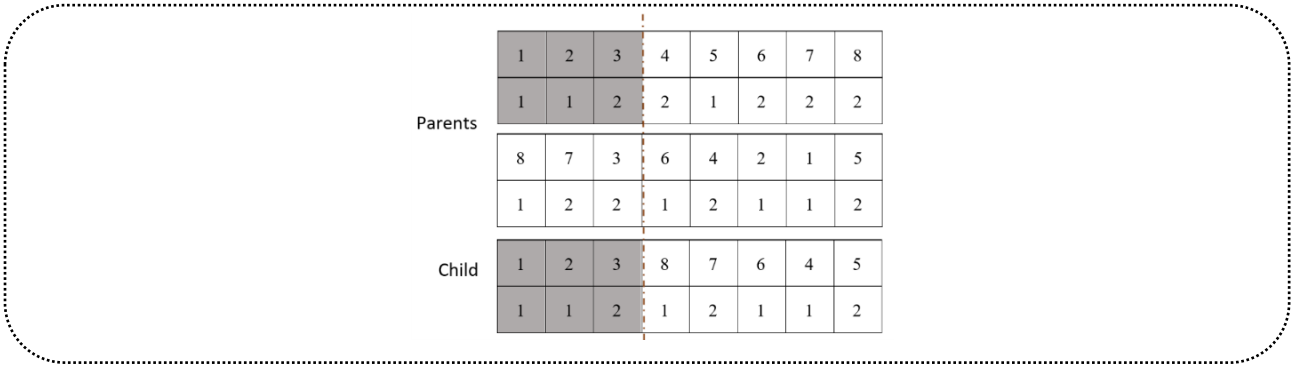

Fig. 4. Crossover operator of the one-point method.

H. mutation operator

After the crossover, the chromosomes undergo a mutations operator. Mutations are designed to help search the entire search space and prevent the algorithm from falling into the local optimal. In this research, the replacement method has been used. In this type of mutation, two random positions of a string are selected, and their related values are exchanged. Figure 5 shows an example of this type of mutation.

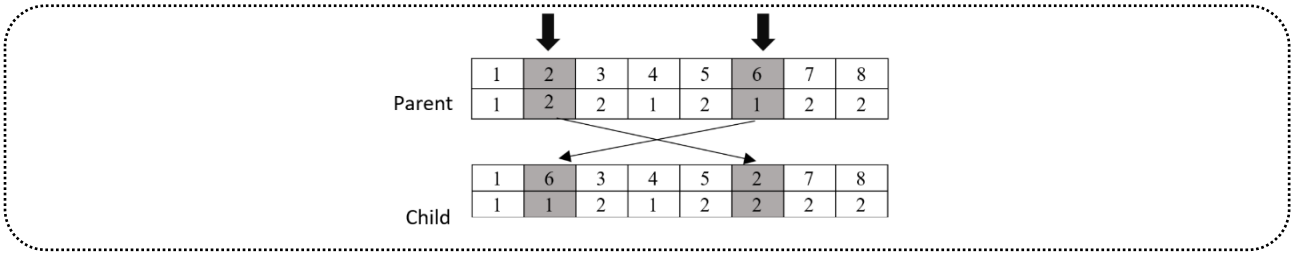

Fig.5. an example of this type of mutation

I. Stop criterion

There are several rules for stopping the genetic algorithm. Maximum generation, time spent, and lack of improvement in fitness. This paper uses the rule of maximum generation. Thus, the genetic algorithm stops when a certain number of generations have occurred. For example, if a generation generator has reached a certain number, the algorithm stops.

For example, 11 work with the numbers i =1,2,3, ...,11 and in order of processing time {6, 8, 5, 8, 4, 3, 4, 4, 9, 5, 3} and delivery time {60, 46, 44, 50, 63, 43, 49, 34, 33, 52, 58} with two values of T_1= 15, T_2 =21 for maximum continuous working time of the machine and also two values of W_1 = 5, W_2 = 7 for periods unavailability is considered. Using the genetic algorithm, the optimal answer is 97. For this example, the best sequence is shown in Figure 6, while any other sequence will have more values in the objective function, and consequently, they will have more delay and early times costs.

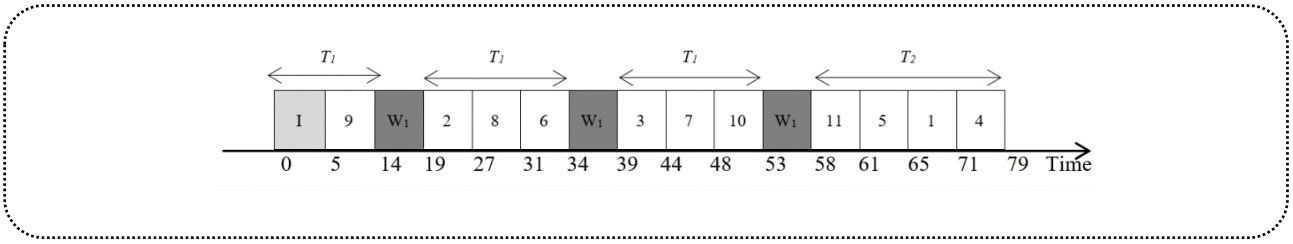

Fig. 6. display the optimal solution for 11 jobs

IV. SENSITIVE ANALYSES

To evaluate the problem, 70 samples were produced. The jobs in the produced samples have values of {8, 9, 10, 11, 12, 15, 18, 21, 24, 27, 30, 35, 40, 40, 50}. Repeat five times for each job at each level. The processing time of the jobs is a random number obtained using a discrete uniform distribution in the interval [1, 10]. Delivery time is also randomly obtained from a discrete uniform distribution from the following range.

$$d_i \sim [(1-R-Q).\sum_{i=1}^{n} p_i \,.\, (1-R+Q).\sum_{i=1}^{n} p_i] \tag{17}$$

In relation (16), the parameter R is the tardiness factor, which is obtained randomly from the set {0.2 and 0.6}. Q is the domain of delivery dates, which is obtained randomly from the set {0.2 and 0.6.}. $\sum_{i=1}^{n} p_i$ is the sum of processing times. The T_1 parameter is randomly selected from the sets {10, 15, 20} و, and the value of W_1 is 5. Also, the parameter T_2 is obtained randomly from the set {1.4T_1 and 1.8T_1}. The value of W_2 is also obtained from 1.6W_1. All of the above are mentioned as input parameters for this algorithm; they have been experimentally obtained and are based on extensive research on various problems.

Tables V: Comparing the result of genetic algorithm ,PSO and GAMS for small sizes programming

n	*i*	T_1	T_2	W_1	W_2	*R*	*Q*	*GAMS result*		*GA result*				*PSO result*			
								OV²	*T¹*	*Min*	*Average*	*Max*	*T*	*Min*	*Average*	*Max*	*T*
8	1	15	27	5	8	0.2	0.2	0	167.98	0	0	0	9.59	0	0	0	14.26
	2	15	27	5	8	0.6	0.6	0	6.18	0	0	0	5.68	0	0.75	1.51	9.75
	3	15	21	5	8	0.2	0.6	0	8.03	0	0	0	9.25	0	2.08	4.16	11.90

Continue Tables V: Comparing the result of genetic algorithm ,PSO and GAMS for small sizes programming

n	*i*	T_1	T_2	W_1	W_2	*R*	*Q*	*GAMS result*		*GA result*				*PSO result*			
								OV[2]	*T*[1]	*Min*	*Average*	*Max*	*T*	*Min*	*Average*	*Max*	*T*
	4	20	28	5	8	0.6	0.6	0	9.72	0	0	0	5.67	0	0.70	1.04	10.10
	5	15	21	5	8	0.2	0.6	0	137.24	0	0	0	9.55	0	1.38	2.77	13.27
Average								0	65.83	0	0	0	7.94	0	0.98	1.89	11.85
9	1	15	21	5	8	0.2	0.6	0	259.05	0	0	0	8.16	6.15	6.92	7.69	13.94
	2	15	21	5	8	0.6	0.6	0	543.20	0	0	0	5.83	1.10	1.38	1.65	10.38
	3	10	18	5	8	0.2	0.2	0	936.44	0	0	0	6.89	2.29	4.02	5.74	12.84
	4	20	28	5	8	0.2	0.6	0	194.83	0	0	0	12.65	9.43	12.26	15.09	15.04
	5	15	27	5	8	0.6	0.6	0	283.85	0	0	0	6.03	0	1.33	2.66	10.92
Average								0	443.47	0	0	0	7.91	3.79	5.18	6.56	12.62
10	1	20	28	5	8	0.6	0.2	0	7200	0	0	0	6.92	4.63	5.29	5.96	12.04
	2	20	36	5	8	0.2	0.2	0	747.02	0	0	0	13.46	10	13	16	20.39
	3	15	27	5	8	0.2	0.2	0	540.2	0	0	0	11.97	2.53	3.79	5.06	16.96
	4	20	36	5	8	0.6	0.6	0	1149	0	0	0	6.43	3.92	4.90	5.88	12.07
	5	20	28	5	8	0.2	0.6	0	65.81	0	0	0	14.56	21.42	23.80	28.57	22.45
Average								0	1940.44	0	0	0	10.66	8.5	10.15	12.29	16.78
11	1	15	21	5	8	0.6	0.6	0	5341	0	0	0	7.15	6.25	7.03	7.81	13.09
	2	10	14	5	8	0.2	0.6	0	7200	0	0	0	7.67	11.81	15.74	19.68	14.53
	3	20	28	5	8	0.6	0.6	0	3041	0	0	0	6.64	0	6.66	13	11.53
	4	10	18	5	8	0.2	0.2	0	7200	0	0	0	8.23	2.61	4.57	6.53	15.28
	5	10	14	5	8	0.6	0.6	0	990.7	0	0	0	8.51	3.61	4.81	6.62	12.64
Average								0	4754.54	0	0	0	7.64	4.85	7.76	10.72	13.41
12	1	15	27	5	8	0.2	0.2	0	7200	0	0	0	12.33	9.27	11.34	13.40	23.39
	2	15	27	5	8	0.2	0.6	0	7200	0	0	0	16.67	29.70	30.19	30.69	23.51
	3	10	14	5	8	0.2	0.6	1.9	7200	0	0	0	9.18	14.64	16.56	18.47	18.24
	4	20	28	5	8	0.2	0.2	2	7200	0	0	0	9.83	14.60	15.73	16.85	18.90
	5	15	21	5	8	0.2	0.2	1	7200	0	0.3	0.4	10.09	2.39	3.82	5.26	21.42
Average								0.98	7200	0	0.06	0.08	11.62	14.12	15.52	16.93	21.09
Total average								0.19	2880.85	0	0.01	0.01	9.15	6.25	7.91	10.51	20.66

In this section, the relative deviation percentage of the values is calculated according to the value obtained from the objective function. Then, the values obtained in Tables V and VI are compared for the mathematical model and the proposed algorithm. The relative deviation percentage of each of the results is calculated as Eq.(18).

$$gap\% = \frac{solution - best\ solution}{best\ solution} \times 100 \tag{18}$$

In Eq. (18) "solution", the answer obtained from the objective function is each of the problem solving, and "best solution" is the best value of the objective function obtained from that problem from between the genetic algorithm and the mathematical model.

Table V solved small size of the problems and found the optimal solution by GAMS software, then solved these problems again by genetic algorithm.

As it is known, the solutions of the genetic algorithm are the same as the optimal solution (GAMS), and this shows the high efficiency of the genetic algorithm for solving the proposed problem. Then the problem is solved for a large size, and its results are shown in Table VI.

The results of solving the samples were calculated according to 70 samples, each of which was dissolved five times by the genetic algorithm and the values of the objective function obtained in the three levels of minimum, maximum and average, and the same function has been performed for the PSO algorithm. In Tables VI, "*" sign indicates that the GAMS software is not able to solve the problem in 7200 seconds. Examining the results shown in Table V and Table VI, it can be seen that due to the complexity of the problem and the limitation of memory, the mathematical model can optimize up to 11 jobs and also, due to the solution of the mathematical model in GAMS software and numerous time reviews that showed that GAMS software can solve the model up to 28 jobs, even with the change of time from 7200 seconds to 10800 seconds and also to 14400 seconds, the results showed any things did not change. And this result for the PSO algorithm showed that the average deviation is 36.24%, of the best solution obtained in the average deviation time is 99.40%. In contrast, the proposed genetic algorithm can solve problems in a shorter time and is a better solution than the mathematical model. Also, due to the complexity of the problem, the mathematical model can solve jobs in small sizes and cannot solve medium and large samples if the algorithm has the ability to solve the samples in small, medium, and large dimensions at a better time and provides an optimal and near-optimal solution. Also, the average deviation in the genetic algorithm is 0.87% of the best solution obtained in a relatively appropriate time. In comparison, this value is 39.98% for GAMS software. Calculation time is higher in some samples due to the implementation of the idleness algorithm for data in which idleness is applied, and this data is more for samples whose value *R* (tardiness factor) is 0.2 because, in this case, the delivery time of jobs is close to each other. Also, samples in which the continuous working time has the highest value job takes longer time to solution, because in times of greater continuous working time, the number of batches decreases, so the number of periods of unavailability decreases, and this causes to jobs by taking idleness the value of an objective function be better. At the same time, the values related to the absence of unforced idleness are listed in the table as "NI" which indicates that unforced idleness is useful for some instances.

Tables VI: Comparing the GA, PSO and GAMS result for large sizes problems

n	*Data*							*GAMS result*		*GA Result*					*PSO Result*			
	i	T_1	T_2	W_1	W_2	*R*	*Q*	*OV*	*T*	*NI*	*Min*	*Average*	*Max*	*T*	*Min*	*Average*	*Max*	*T*
15	1	10	14	5	8	0.6	0.2	0	7200	*	0	0	0	8.52	8.61	8.80	8.98	17.35
	2	15	27	5	8	0.6	0.2	0	7200	*	0	0	0	8.52	16.11	16.66	17.22	18.03
	3	15	27	5	8	0.6	0.2	0.9	7200	*	0	0	0	8.45	9.52	11.90	14.28	19.44

Continue Tables VI: Comparing the GA, PSO and GAMS result for large sizes problems

n	*Data*							*GAMS result*		*GA Result*					*PSO Result*			
	i	T_1	T_2	W_1	W_2	*R*	*Q*	*OV*	*T*	*NI*	*Min*	*Average*	*Max*	*T*	*Min*	*Average*	*Max*	*T*
15	4	15	21	5	8	0.2	0.2	13	7200	12.1	0	0	0	12.29	15.26	16.31	17.36	27.91
	5	10	14	5	8	0.6	0.6	0.04	7200	*	0	0.1	0.6	8.22	15.43	16.88	18.32	16.93
Average								2.78	7200		0	0.02	0.12	9.2	12.98	14.11	15.23	19.93
18	1	15	21	5	8	0.2	0.6	6.7	259.05	1.53	0	0.5	1.3	11.28	7.26	8.89	12.23	37.62
	2	15	21	5	8	0.6	0.6	13.5	543.20	*	0	0.8	1.5	9.59	12.63	15.78	18.94	35.71
	3	10	18	5	8	0.2	0.2	10.5	936.44	*	0	0	0	9.55	20.33	22.56	24.79	36.59
	4	20	28	5	8	0.2	0.6	21.7	194.83	*	0	0.6	1	9.62	28.04	30.23	32.43	36.91
	5	15	27	5	8	0.6	0.6	2.2	283.85	*	0	0.2	0.7	9.69	32.95	39.20	45.83	35.71
Average								10.92	443.47		0	0.42	0.9	9.94	20.24	23.33	26.84	36.50
21	1	20	28	5	8	0.6	0.2	20	7200	*	0	0.4	1	10.79	15.68	18.12	20.57	40.29
	2	20	36	5	8	0.2	0.2	60.7	747.02	36.05	0	1.9	3.1	41.19	51.36	56.97	62.58	60.97
	3	15	27	5	8	0.2	0.2	49	540.2	*	0	0.6	1.6	10.84	24.32	27.16	30	20.92
	4	20	36	5	8	0.6	0.6	25.2	1149	23.14	0	1.9	4.7	35.42	51.71	53.71	55.71	57.99
	5	20	28	5	8	0.2	0.6	94.7	65.81	63.37	0	0.4	0.5	41.54	84.79	90.35	95.90	60.04
Average								46.32	1940.44		0	1.04	2.18	27.95	45.57	49.26	52.95	48.04
24	1	15	21	5	8	0.6	0.6	27.4	5341	*	0	1.8	2.5	14.88	13.37	15.71	18.05	29.23
	2	10	14	5	8	0.2	0.6	17.1	7200	*	0	1.2	3.9	13.50	27.48	29.31	31.14	21.97
	3	20	28	5	8	0.6	0.6	140.7	3041	27.27	0	1.2	2	45.36	46.66	53.18	59.69	91.12
	4	10	18	5	8	0.2	0.2	21.7	7200	*	0	1.3	2.4	13.66	18.12	20.27	22.41	21.77
	5	10	14	5	8	0.6	0.6	124.1	990.7	*	0	0.8	2	13.50	61.13	66.58	72.02	22.67
Average								66.2	4754.54		0	1.26	2.56	20.18	33.35	37.01	40.66	37.35
27	1	15	27	5	8	0.2	0.2	47.7	7200	*	0	2	3.6	14.28	25.18	26.36	27.55	22.74
	2	15	27	5	8	0.2	0.6	204.1	7200	*	0	0.5	1.4	15.67	13.84	15.09	16.16	26.01
	3	10	14	5	8	0.2	0.6	47.1	7200	*	0	0.7	2.1	15.24	18.93	20.11	21.46	47.05
	4	20	28	5	8	0.2	0.2	5.4	7200	4.2	0	1.3	2.7	39.79	50.93	55.01	60.69	78.39
	5	15	21	5	8	0.2	0.2	64.1	7200	*	0	1.1	2.5	16.78	58.68	61.37	64.06	70.30

Continue Tables VI: Comparing the GA, PSO and GAMS result for large sizes problems

n	*Data*							*GAMS result*		*GA Result*					*PSO Result*			
	i	T_1	T_2	W_1	W_2	*R*	*Q*	*OV*	*T*	*NI*	*Min*	*Average*	*Max*	*T*	*Min*	*Average*	*Max*	*T*
Average								73.68	7200		0	1.12	2.64	20.35	33.51	35.58	37.98	48.89
30	1	20	36	5	8	0.2	0.6	*	*	17.85	0	2.1	5.7	67.28	66.12	68.57	71.03	194.78
	2	20	36	5	8	0.2	0.6	*	*	32.98	0	0.8	2.5	67.52	57.54	62.98	68.42	195.97
	3	20	28	5	8	0.2	0.6	*	*	22.18	0	1	2.1	63.67	64.72	69.09	73.45	182.10
	4	10	14	5	8	0.2	0.2	*	*	*	0	1	2.6	17.42	26.46	28.43	30.40	74.46
	5	15	21	5	8	0.2	0.2	*	*	*	0	0.8	1.7	21.20	20.73	22.42	24.11	122.54
Average								*	*		0	1.14	2.94	47.38	47.11	50.29	53.48	153.97
35	1	15	27	5	8	0.6	0.2	*	*	*	0	0.9	1.7	16.85	21.83	23.86	25.89	55.88
	2	20	36	5	8	0.2	0.2	*	*	8.64	0	0.5	1.3	45.40	25.63	27.68	29.74	220.39
	3	10	14	5	8	0.2	0.6	*	*	*	0	1.2	2.6	17.69	43.57	44.48	45.39	68.59
	4	10	18	5	8	0.6	0.2	*	*	*	0	1.1	1.8	16.73	10.50	11.47	12.43	54.89
	5	20	28	5	8	0.6	0.6	*	*	*	0	0.9	2.4	16.96	59.52	63	66.41	53.02
Average								*	*		0	0.92	1.96	22.72	32.21	34.09	35.97	90.55
40	1	10	18	5	8	0.2	0.6	*	*	*	0	2.4	3.81	25.54	47	50.56	53.63	99.51
	2	15	27	5	8	0.2	0.2	*	*	*	0	0.3	0.8	46.67	18.80	20.56	22.32	244.44
	3	20	28	5	8	0.2	0.6	*	*	21.78	0	1.1	3.7	93.69	79.11	82.12	85.14	289.54
	4	10	14	5	8	0.6	0.2	*	*	*	0	1.1	2	18.96	19.06	19.77	20.48	59.68
	5	15	21	5	8	0.2	0.6	*	*	4.99	0	1.4	2.3	58.49	67.99	69.60	71.21	217.10
Average								*	*		0	1.26	2.52	48.67	46.39	48.52	50.55	182.05
50	1	15	21	5	8	0.6	0.6	*	*	*	0	0.3	0.7	23.45	45.02	46.33	47.64	187.23
	2	15	21	5	8	0.2	0.6	*	*	*	0	0.9	2	31.20	44.03	45.15	46.27	530.27
	3	15	21	5	8	0.6	0.2	*	*	*	0	0.5	0.8	22.10	17.03	18.34	19.66	198.62
	4	20	36	5	8	0.2	0.2	*	*	14.01	0	1	1.46	91.23	18.27	18.30	18.34	284.13
	5	15	27	5	8	0.6	0.6	*	*	*	0	0.8	1.6	21.52	38.93	41.94	44.95	186.41
								*	*		0	0.7	1.31	37.90	32.65	34.01	35.37	277.33
Total average								39.98	7200		0	0.87	1.88	27.14	33.77	36.24	38.78	99.40

V. CONCLUSION

In this paper, the problem of scheduling a single machine with the objective function of minimizing the sum of earliness and tardiness with flexible periodic availability constraints has been investigated. Today, one of the most important features of production is the extensive application of the concept of JIT production in the management of production and inventory, in which each job should be completed as close as possible to its delivery date. In such environments, it is an important period of time that a job is an earliness or tardiness. Therefore, scheduling programs are responsible for minimizing the total earliness and tardiness, which indicates customer satisfaction as well as the level of product performance in terms of inventory. In most research, one of the most common assumptions is that machines are always available along the planning horizon. But in the real world, a machine may not be available for a variety of reasons, such as breakdowns or the need for preventive maintenance or change tools. In this paper, the unavailability constraint of two flexible modes has been investigated. According to this definition, in each period, the duration of unavailability changes according to the continuous working time of the device in a discrete manner and can take two different values. Jobs are available in zero time, and preemption is not allowed; the data is also deterministic. Because the objective function is irregular unforced idleness may be helpful, which this subject increases the complexity of the problem. In this paper, for this problem, a mathematical model of mixed-integer linear programming has been proposed to obtain the optimal solution of the problem, which in GAMS plan was evaluated with a CPLEX solver in that the model was only able to solve the problem optimally up to 11 dimensions. Therefore, due to that, the problem is NP-hard, and also due to that the model was not able to solve the problem in medium and large dimensions, a genetic algorithm has been proposed to solve it. In order to evaluate the performance of the proposed genetic algorithm, a PSO algorithm is designed and compared with the genetic algorithm. In the following, to examine the performance of the proposed algorithms, a number of sample problems were designed and solved, and compared with the mathematical model and PSO algorithm. Finally, the results of generating 70 sample problems showed that the proposed genetic algorithm provided optimal values in small samples in a much shorter time and medium and large sizes more appropriate values than the model at a reasonable time with a relative error value of 0.87% percent Provides.

Following extensive studies in the field of single machine scheduling with the objective function of minimizing the sum of earliness and tardiness with bimodal flexible periodic availability constraints and doing the present study, the following are some suggestions for future research:

1. Special and numerous conditions can be created in the scheduling problem, which has not been observed in the investigated studies. These include restrictions such as preconditions, blocking, etc., along with availability constraints.
2. Availability constraints examined in this paper are flexible. Considering other types of unavailability introduced in this paper, this problem can be done with variable unavailability constraint that has been less studied.
3. Given that preemption is not allowed in this problem and the non-resumable Jobs, as well as few studies, have been done on semi-resumable jobs, this problem can be developed by considering semi-resumable jobs.
4. Heuristic and metaheuristic algorithms are commonly used to solve scheduling problems. In most scheduling research, algorithms are used individually or in combination. Due to the NP-Hard nature of these problems, we can use other metaheuristic algorithms to improve computational time reduction and better responses.

REFERENCES

Ahmadizar, F. and Eteghadipour, J. *Single-machine earliness–tardiness scheduling with two competing agents and idle time.* Engineering Optimization, 2017. 49(3): p. 499-512.

Ahmadizar, F. and Farhadi, S. Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. Computers & Operations Research, 2015. 53: p. 194-205.

Benmansour, R., et al., *Minimizing the weighted sum of maximum earliness and maximum tardiness costs on a single machine with periodic preventive maintenance*. Computers & Operations Research, 2014. 47: p. 106-113.

Chen, L., Wang, J. and Yang, W. *A single machine scheduling problem with machine availability constraints and preventive maintenance.* International Journal of Production Research, 2021. 59(9): p. 2708-2721.

Du, J. and Leung, J.Y.-T. *Minimizing total tardiness on one machine is NP-hard.* Mathematics of operations research, 1990. 15(3): p. 483-495.

Feldmann, M. and Biskup, D. *Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches*. Computers & Industrial Engineering, 2003. 44(2): p. 307-323.

Ganji, F., Moslehi, G. and Ghalebsaz Jeddi, B. *Minimizing maximum earliness in single-machine scheduling with flexible maintenance time.* Scientia Iranica, 2017. 24(4): p. 2082-2094.

Jayanthi, S. and Anusuya, S. *Minimization of total weighted earliness and tardiness using PSO for one machine scheduling.* International Journal of Pure and Applied Mathematical Sciences, 2017. 10(1): p. 35-44.

Kellerer, H., Rustogi, K. and Strusevich, V.A. *A fast FPTAS for single machine scheduling problem of minimizing total weighted earliness and tardiness about a large common due date. Omega*, 2020. 90: p. 101992.

Khanh Van, B. and Van Hop, N. Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness-tardiness. *Journal of Industrial and Production Engineering*, 2021. 38(1): p. 18-28.

Lin, S.-W., et al., *Single machine job sequencing with a restricted common due window*. IEEE Access, 2019. **7**: p. 148741-148755.

Low, C., Li, R.-K. and Wu, G.-H. *Minimizing total earliness and tardiness for common due date single-machine scheduling with an unavailability interval*. Mathematical Problems in Engineering, 2016. 6: p. 1-12.

Luo, W., Cheng, T.E. and Ji, M. *Single-machine scheduling with a variable maintenance activity.* Computers & Industrial Engineering, 2015. 79: p. 168-174.

Mahnam, M., Moslehi, G. and Ghomi, S.M.T.F. *Single machine scheduling with unequal release times and idle insert for minimizing the sum of maximum earliness and tardiness*. Mathematical and Computer Modelling, 2013. 57(9-10): p. 2549-2563.

Mashkani, O. and Moslehi, G. *Minimising the total completion time in a single machine scheduling problem under bimodal flexible periodic availability constraints*. International Journal of Computer Integrated Manufacturing, 2016. 29(3): p. 323-341.

M'Hallah, R. and Alhajraf, A. *Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem*. Journal of Scheduling, 2016. 19(2): p. 191-205.

Mozaffariyan, S. and Sahraeian, R. *Single-machine scheduling considering carryover sequence-dependent setup time, and earliness and tardiness penalties of production*. Journal of Industrial and Systems Engineering, 2020. 13(Special issue: 16th International Industrial Engineering Conference): p. 112-120.

Niroomand, S., et al., *Hybrid greedy algorithms for fuzzy tardiness/earliness minimisation in a special single machine scheduling problem: case study and generalisation*. International Journal of Computer Integrated Manufacturing, 2016. 29(8): p. 870-888.

Pacheco, J., et al., *Variable neighborhood search with memory for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times*. Knowledge-Based Systems, 2018. 145: p. 236-249.

Sadeghi, H., *A forecasting system by considering product reliability, POQ policy, and periodic demand.* Journal of Quality Engineering and Production Optimization, 2019. 4(2): p. 133-148.

Sajadi, S., Arianezhad, M.G. and Sadeghi, H.A. *An Improved WAGNER-WHITIN Algorithm.* 2009. 20(3): p. 117-123.

Sadeghi, H., Golpîra, H. and Khan, S.A.R. *Optimal integrated production-inventory system considering shortages and discrete delivery orders.* Computers & Industrial Engineering, 2021. 156: p. 107233.

Shahriari, M., et al., *JIT single machine scheduling problem with periodic preventive maintenance.* Journal of Industrial Engineering International, 2016. 12(3): p. 299-310.

Touat, M., et al., *A hybridization of genetic algorithms and fuzzy logic for the single-machine scheduling with flexible maintenance problem under human resource constraints*. Applied Soft Computing, 2017. 59: p. 556-573.

Tsai, T.-I., *A genetic algorithm for solving the single machine earliness/tardiness problem with distinct due dates and ready times*. The International Journal of Advanced Manufacturing Technology, 2007. 31(9-10): p. 994-1000.

Wan, L. and Yuan, J. *Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard.* Operations Research Letters, 2013. 41(4): p. 363-365.

Wang, J.-B., Hu, Y. and Zhang, B. *Common due-window assignment for single-machine scheduling with generalized earliness/tardiness penalties and a rate-modifying activity*. Engineering Optimization, 2021. 53(3): p. 496-512.

Xiong, X., et al., *Single-machine scheduling and common due date assignment with potential machine disruption.* International Journal of Production Research, 2018. 56(3): p. 1345-1360.

Yuce, B., et al., *Hybrid Genetic Bees Algorithm applied to single machine scheduling with earliness and tardiness penalties*. Computers & Industrial Engineering, 2017. 113: p. 842-858.