Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Flexible job-shop scheduling problem with the number of workers dependent processing times

Busra Tutumlu[a*] and Tugba Saraç[b]

[a]*Kutahya Dumlupinar University, Turkey*
[b]*Eskisehir Osmangazi University, Turkey*

C H R O N I C L E

Article history:
Received September 14 2024
Received in Revised Format December 2 2024
Accepted January 29 2025
Available online January 29 2025

Keywords:
Flexible Job-Shop Scheduling Problem
The Number of Workers Dependent Processing Times
Mixed-Integer Programming
NSGA-II

A B S T R A C T

Studies in the literature on flexible job-shop scheduling problems (FJSP) generally assume that one worker is assigned to each machine and that processing times are constant. However, in some industries, multiple workers with cooperation can process complex operations faster than one worker. If the possibility of completing jobs in a shorter time with worker cooperation is not taken into account, the opportunity to create more effective schedules may not be taken advantage of. Therefore, it is essential to consider the flexibility of collaboration between employees. However, to increase labor efficiency in businesses, jobs are also expected to be done with the minimum number of workers possible. This study considers the FJSP with both machine and number of workers dependent processing times. The objectives are minimizing the total tardiness and the total number of workers. A bi-objective mathematical model and an NSGA-II algorithm for large-sized problems have been proposed. The performance of the proposed solution approaches is demonstrated by using randomly generated test problems. For each problem, the most successful Pareto solution among the obtained solutions by the mathematical model and the NSGA-II algorithm was determined using the TOPSIS method. Furthermore, the effect of the total number of workers on the total tardiness is examined. The performance of proposed solution approaches, and when the worker number increases, the total tardiness of jobs can be reduced by an average of 75.88%, have been shown through comprehensive experimental studies.

1. Introduction

Effective production management is essential for businesses, offering a strategic advantage in today's rapidly evolving and complex manufacturing landscapes. One of the crucial topics for developing production strategies is job-shop scheduling. A flexible job-shop scheduling problem (FJSP) is a planning process that determines how to organize the jobs and machines in a job-shop under certain constraints. In this problem, having more than one machine that can do the same job in production environments creates flexibility. One of the main objectives is to optimize production processes by using resources efficiently. One of the most important resources in a production process is labor.

The processing times of some complex jobs may vary directly depending on the number of workers performing these jobs. This is an important factor to be considered especially in the planning and scheduling stages of the jobs in production and operations management. Surprisingly, despite its importance, the worker factor in the FJSP problem has been largely neglected in the existing literature. This study aims to fill this gap and highlight the need to consider this factor in the FJSP.

When scheduling, determining the number of workers to be assigned to each machine is a critical problem in terms of increasing efficiency and balancing costs. Assigning more workers to the machines can speed up the completion times of the jobs and reduce possible tardiness and the costs that will arise from this. However, this increase in the number of workers will also cause labor costs to increase. It is necessary to develop an appropriate strategy to achieve this balance, that is, to minimize tardiness while keeping total labor costs under control. Therefore, this study aims to optimize both total tardiness and the number of workers assigned to the machines.

* Corresponding author
E-mail busra.tutumlu26@gmail.com (B. Tutumlu)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)

doi: 10.5267/j.ijiec.2025.1.007

Recently, there has been an increasing focus on integrating the worker factor into the FJSP, as evidenced by emerging studies in this area. Studies addressing the FJSP in the literature have considered machine constraints and worker flexibility in recent years. In the literature, this problem is defined as either worker-constrained or dual resource-constrained. The available studies in the literature on FJSP with worker-constrained constraints are examined. The objective functions, the worker number assigned to a machine simultaneously, and the processing time characteristics of these studies are given in Table 1.

Table 1
Studies on FJSP with worker-constrained

		Processing time			***Worker Number***		
Addressed by	***Objective function(s)***	*Machine dependent*	*Worker dependent*	*Worker number dependent*	<1	1	>1
This study	TT, TW	✓		✓			✓
Shi et al., 2023	$C_{max,}$ HWJS	✓	✓			✓	
Luo et al., 2023	C_{max}, MWM, MWW	✓					✓
Grumbach et al., 2023	C_{max}, TT	✓	✓		✓		
He et al., 2022	$C_{max,}$ TT	✓	✓			✓	
Luo et al., 2022	C_{max}, MWM, TMW	✓				✓	
Zhang et al., 2022	C_{max}	✓	✓			✓	
Defersha et al., 2022	C_{max}	✓			✓		
Lou et al., 2022	$C_{max,}$ MWM, MWW	✓	✓			✓	
Yan et al., 2022	C_{max}	✓				✓	
Vital-Soto et al., 2022	C_{max}, MWW, WT	✓	✓			✓	
Gnanavelbabu et al., 2021	C_{max}	✓	✓			✓	
Liu 2021	C_{max}, TWC, GPF	✓	✓			✓	
Gong et al., 2021	C_{max}, TWC, GPF	✓	✓			✓	
Tan et al., 2021	MWF, C_{max}	✓	✓			✓	
Renna et al., 2020	TMW	✓				✓	
Gong et al., 2020	C_{max}	✓	✓			✓	
Kress et al., 2019	C_{max}, TT	✓	✓			✓	
Meng et al., 2019	TEC	✓	✓			✓	
Gong et al., 2018	C_{max}, MWM, TMW	✓	✓			✓	
Zheng et al., 2016	C_{max}	✓	✓			✓	

TT: Total tardiness, TW: Total number of workers, C_{max}: Makespan, HWJS: Higher worker job satisfaction, MWM: Maximum machines' workload, MWW: Maximum workers' workload, TMW: Total machine workload, WT: Weighted tardiness, TWC: Total worker costs, GPF: The green production factors, MWF: Maximum worker fatigue, TEC: Total energy consumption

As can be seen from Table 1, in most studies, the problem is addressed as multi-objective. The most commonly considered objective functions are C_{max}, TT, MWM, TMW, and MWW. In addition, In the literature, studies that consider the worker factor and define the processing time depending on the worker have only examined the situation where a time difference occurs due to the performance of the assigned worker. None of these studies have considered the effect of the number of workers assigned to a job on the processing time. In studies, the worker number assigned to a machine simultaneously is generally assumed to be one. However, due to the characteristics of production, there are situations where a worker is assigned to more than one machine or more than one worker is assigned to a machine.

Some businesses have automatic machines, and therefore, in automatic machines, there is no need for the worker to wait at the machine throughout the process. Once a worker has started the process on the machine, he/she can be assigned to other secondary activities. Studies addressing this situation include Grumbach et al. (2023) and Defersha et al. (2022) can be given as examples.

On the other hand, in some businesses, more than one worker can be assigned to a machine/bench simultaneously. This situation is frequently seen, especially in businesses where complex jobs are carried out. Luo et al. (2023) can be given as an example. Luo et al. (2023) filled the gap in the literature for situations where one or more workers are required for machines simultaneously in FJSP, where worker flexibility is also included. However, they assumed that the worker number on the machines is known in advance and the processing times are dependent only on the machines.

This study discusses an FJSP with worker constraints. More than one worker can be assigned to the machines simultaneously. The required worker number for the machines is not known in advance and is a decision variable. Processing times depend on both the machine and the number of workers. Considering the accessible literature, processing times depending on the number of workers are considered for the first time in this study. A multi-objective mathematical model is proposed to solve the addressed problem. The objective functions of the proposed model are minimizing the total tardiness and the worker number. In addition, the NSGA-II algorithm has been developed to solve large-scale problems.

The study's next section explains the considered problem and the proposed mathematical model. Then, in the third section, the proposed NSGA-II algorithm is presented. The fourth section contains the experimental results, while the final section offers conclusions and recommendations.

2. Considered problem and mathematical model

In the considered problem, there are *n* jobs and *m* machines. Every job consists of operations. Operations must be performed in a particular order. This sequence is called the route. Each operation has a subset of machines to which it can be assigned. A machine is capable of processing only one operation at a time. When an operation is started on a machine, it continues on the same machine until it is completed; in other words, no operation splitting is allowed. At the beginning of the planning period, all jobs are ready to assign the machines. A different number of workers can be assigned to each machine. Processing times depend on the machine and the number of workers assigned. The problem is aimed to minimize the total tardiness and the total number of workers simultaneously.

The mathematical model proposed for the considered problem is given below.

Indices:

i, r*:* indices of jobs. $i, r \in \{1,2,3 \dots N\}$

v, s*:* indices of operations. $v, s \in \{1,2,3 \dots N_i\}$

k, a*:* indices of machines. $k, a \in \{1,2,3 \dots M\}$

l, b*:* indices of sequences. $l, b \in \{1,2,3 \dots G\}$

j*:* index of number of workers. $j \in \{1,2,3 \dots Q\}$

Parameters:

N*:*Number of jobs.

Q*:* Maximum number of workers that can be assigned to a machine.

M*:* Number of machines.

w_i: Number of operations of the job *i.*

p_{ivkj}: Processing time of operation *v* of job *i* with *j* workers on the machine *k.*

d_i: Due date of the job *i.*

u_{ivk}: If machine *k* is capable of processing operation *v* of job *i,* 1; otherwise, 0.

β*:* Sufficiently large positive number.

Decision Variables:

x_{ivkl}: Equals 1 if operation *v* of job *i* is processed on machine *k* in the position *l* and otherwise, equals 0.

z_{kj}: Equals 1, if *j* number of workers are assigned to the machine *k* and otherwise equals 0.

T_i: Tardiness of job *i.*

C_{ivk}: Completion time of operation *v* of job *i* on the machine *k.*

Objective Functions:

$$min\ f_1 = \sum_i T_i \quad (1)$$

$$min\ f_2 = \sum_j \sum_k j z_{kj} \quad (2)$$

constraints:

$$\sum_k \sum_l x_{ivkl} = 1 \qquad \forall i, v | v \le w_i \qquad (3)$$

$$\sum_k \sum_l x_{ivkl} = 0 \qquad \forall i, v | v > w_i \qquad (4)$$

$$\sum_i \sum_{v|v \le w_i} x_{ivkl} \le 1 \qquad \forall k, l \qquad (5)$$

$$\sum_l x_{ivkl} \le u_{ivk} \qquad \forall i, v, k | v \le w_i \qquad (6)$$

$$l - b \ge 1 - M(2 - x_{ivkl} - x_{iskb}) \qquad \forall i, v, k, l, s, b \; | v > s, v \le w_i, l \ne b \qquad (7)$$

$$\sum_i \sum_{v|\, v \le w_i} x_{ivkl} - \sum_r \sum_{s|s \le w_r} x_{rsk(l-1)} \le 0 \qquad \forall k, l | l > 1 \qquad (8)$$

$$C_{ivk} + M\left(2 - x_{ivkl} - z_{kj}\right) \ge p_{ivkj} \qquad \forall i, v, k, l, j \;\; | l = 1, v = 1 \qquad (9)$$

$$C_{ivk} + M\left(3 - x_{ivkl} - x_{rsk(l-1)} - z_{kj}\right) \ge C_{rsk} + p_{ivkj} \qquad \forall\, i, v, r, s, k, l, j \; | l > 1, v = 1, i \ne r \qquad (10)$$

$$C_{ivk} + M\left(3 - x_{ivkl} - x_{i(v-1)ab} - z_{kj}\right) \ge C_{i(v-1)a} + p_{ivkj} \qquad \forall i, v, k, l, a, b, j \; | l = 1, v > 1, v \le w_i \qquad (11)$$

$$C_{ivk} + M\left(3 - x_{ivkl} - x_{rsk(l-1)} - z_{kj}\right) \ge C_{rsk} + p_{ivkj} \qquad \forall i, v, r, s, k, l, j \; | l > 1, \; i \ne r, v \le w_i, \; s \le w_r \qquad (12)$$

$$C_{ivk} + M\left(3 - x_{ivkl} - x_{i(v-1)ab} - z_{kj}\right) \ge C_{i(v-1)a} + p_{ivkj} \qquad \forall i, v, k, l, a, b, j \; | v > 1, j \le w_i \qquad (13)$$

$$T_i \ge C_{ivk} - d_i \qquad \forall i, v, k \; | v \le w_i \qquad (14)$$

$$\sum_j z_{jk} = 1 \qquad \forall k \qquad (15)$$

$$C_{ivk} \ge 0 \qquad \forall i, v, k \qquad (16)$$

$$T_i \ge 0 \qquad \forall i \qquad (17)$$

$$x_{ivkl} \in \{0,1\} \qquad \forall i, v, k, l \qquad (18)$$

$$z_{kj} \in \{0,1\} \qquad \forall j, k \qquad (19)$$

Eq. (1) and Eq. (2) show the objective functions. Eq. (1) is the minimization of the total tardiness of jobs, and Eq. (2) is the minimization of the total number of workers. Eq. (3) and Eq. (4) ensure that the operations of each job are assigned to a sequence of a machine and operations that are not related to the job are not assigned. Eq. (5) guarantees the assignment of at most one job to a sequence of a machine, while Eq. (6) is the constraint related to machine eligibility. Eq. (7) and Eq. (8) ensure that the operations of the jobs are processed sequentially in the machines. Equation (9) calculates the completion time when the first operations of the jobs are assigned to the first sequence. Eq. (10) calculates the completion time when the first operation of jobs processed in the second or later sequence. Eq. (11) calculates the completion time of the jobs when the second or later operations are processed in the first sequence. Eq. (12) calculates the completion time of operations other than the first sequence of jobs. Eq. (13) calculates the completion time of the second or later operations of the jobs. Eq. (14) calculates the tardiness of the jobs. Eq. (15) determines the worker number on each machine. Eqs. (16)-(19) set the type of the variables.

3. Proposed NSGA-II Algorithm

One of the multi-objective optimization algorithms is NSGA-II. This algorithm is a fast and efficient algorithm that can find a large number of effective solutions without the need for weight sets. NSGA-II is based on a genetic algorithm that finds Pareto-optimal solutions. Different from the steps of the genetic algorithm, a non-dominated sorting approach and crowding distance calculation procedure are applied. For each objective function, every individual in the population is compared to every other individual using the non-dominated sorting approach. Individuals dominated by each individual and individuals dominating this individual are recorded. If no individual dominates an individual, that individual is included in the F1 class, which is the best class. If only one individual dominates over the individual, it is included in the F2 class. In this way, all individuals are classified according to the non-dominated sorting approach.

Individuals for each objective are first sorted from smallest to largest to calculate the crowding distance. The crowding distance of the individuals with the smallest and largest values is taken as infinity. Then, the crowding distance value (CD_i) of the other individuals, excluding the largest and smallest individuals for that objective, is calculated. The formula for CD_i is given in Eq. (20). In the formula, *g* and *i* are the objective function and individual indexes, respectively.f_i^g is the value of objective function *g* of individual *i*.

$$CD_{\text{i}} = \sum_{g=1}^{n} \frac{|f_{i+1}^g - f_{i-1}^g|}{f_{enb}^g - f_{enk}^g} \qquad \forall_i \qquad (20)$$

The NSGA-II steps are as follows:

Step 0. Determine population size (*ps*), number of iterations (*T*), crossover rate (P_c), and mutation rate (P_m).
Step 1. Create the initial population (P_0). *ps*(P_0)=*N,* iteration number *(t)=0.*
Step 2. Apply the crossover and mutation operator. Q_t is the population after crossover and mutation. *ps*(Q_t)=*N*.
Step 3. Obtain a population of size *2N* (R_t) with the populations obtained in *Step 1* and *Step 2.* ($R_t = P_t U Q_t$)
Step 4. Calculate the value of each individual's fitness function.
Step 5. Apply the non-dominated sorting approach to the 2N population.

Step 6. Once the sorting is complete, individuals of size N will be added to the new population, starting with class F_1. Calculate the crowding distance if the size of the population exceeds N when all individuals in the last class to be included are added ($F_1+F_2+...+F_n>N$).
Step 7. If the crowding distance values were calculated in *Step 6*, sort these values from largest to smallest and include individuals, starting with the individual with the largest value until the population size is N.
Step 5. Increase the number of t by one ($t = t+1$), and if the termination criterion is met, stop. If not, update the last population to P_t and return to *Step 2*.

3.1. Solution Representation

The solution of the proposed problem is represented by two structures, both job-machine and worker-machine. The first one, the job-machine representation, shows the machines to which the operations of the jobs are assigned and their sequence in the machines. This representation's length is determined by the total number of operations, and the number of repetitions of jobs in the first row indicates which operation is involved. The second one, worker-machine representation, shows the number of workers in the machines. The number of machines determines this representation's length. Figure 1 shows the representation of an example with four jobs and three machines. Where O_{iv} is express operation v of job i.

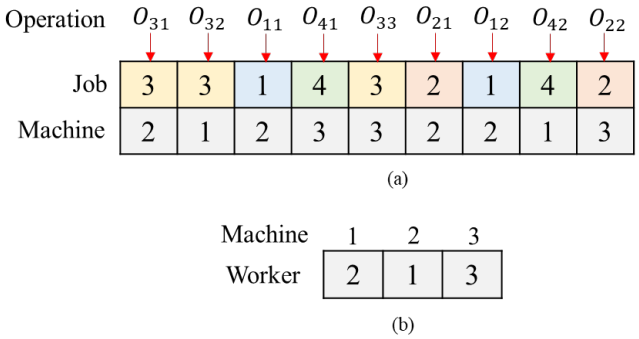

Fig. 1. Representations
a-)Job-Machine Representation b-) Worker-Machine Representation

When Fig. 1 is examined, it is seen that according to (a), job 1, job 2, and job 4 have two operations, and job 2 has three operations. When (a) is decoded, O_{32} and O_{42} are processed in Machine 1, O_{31}, O_{11}, O_{21} and O_{12} in Machine 2, and finally O_{41}, O_{33} and O_{22} in Machine 3 respectively. When (b) is decoded, there are two workers in Machine 1, one in Machine 2 and three in Machine 3.

3.2. Generating the Initial Population

As with all population-based metaheuristic algorithms, the first step in NSGA-II is to generate an initial population. The following steps are followed to generate the population of the addressed problem,
Step 0. Determine the population size (*ps*) and define t=0.
Step 1. Update t=t+1 and define a list (L) for the job row of the t^{th} solution of the population. Add the index of each job to list L as many times as its total number of operations. For example, if job 1 and job 2 have 3 operations and job 3 has two operations, the list L is created as L=[1,1,1,2,2,2,3,3].
Step 2. Shuffle the list elements L=[3,1,2,3,1,1,2,2]. This new L is the job row.
Step 3. Create the machine row by randomly selecting for each operation one of the machines to which that operation can be assigned. *Step 4*. To create the worker row, assign a random number of workers in the range [1, Q] for each machine.
Step 5. If t=*ps*, stop. If not, return to Step 1.

3.3. Operators

In this study, crossover and mutation operators are used. The JBX crossover operator in the literature has been applied for the job row in the proposed representation structure. The steps of this operator are as follows.
Let P_1 and P_2 be the chromosomes to which the crossover operator is applied, and P_1' and P_2' be the new chromosomes formed at the end of the crossover.
Step 1: Divide the job set $J = \{J_1, J_2, ..., J_n,\}$ into two random subsets ($jobset_1$, $jobset_2$).
Step 2: Position the members belonging to $jobset_1$ in P_1 to the same places in P_1', and the members belonging to $jobset_2$ in P_2 to the same places in P_2'.
Step 3: Add the members belonging to $jobset_2$ in P_2 to the empty places in P_1', and the members belonging to $jobset_1$ in P_1 to the empty places in P_2' from left to right.

An approach based on the JBX crossover operator has been applied for the machine row. The machines on which the operations of jobs in $jobset_1$ are assigned in P_1, to P_2', and the machines of jobs in $jobset_2$ to which operations are assigned in P_2 , to P_1' are transferred. For the worker row, the one-point crossover operator is used. Figure 2 shows the application of crossover operators.

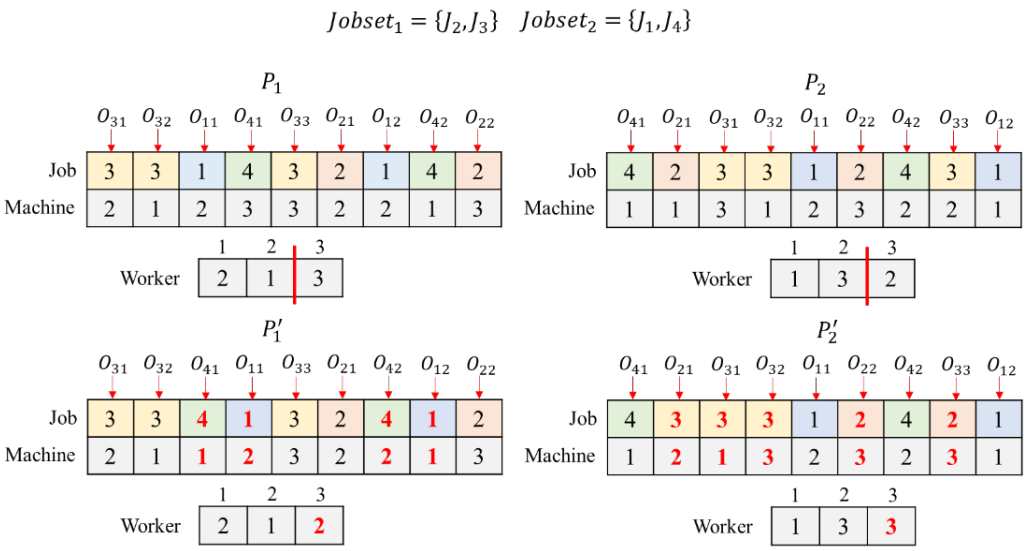

Fig. 2. Crossover operators

The swap operator is applied as the mutation in the job row. A machine is randomly chosen from those available to handle the changing job's operations within the job row, thereby updating the machine information for the corresponding job operation in the machine row. In the worker row, a random integer is derived from the range [1, Q], and the worker information of any machine is replaced with this number. Fig. 3 shows the application of mutation operators.

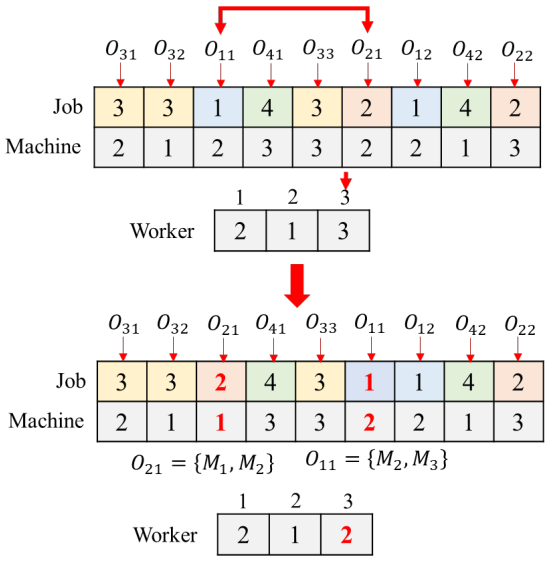

Fig. 3. Mutation operators

4. Experimental Results

Various test problems are generated in different sizes to assess the performance of the proposed mathematical model and NSGA-II. A desktop with an Intel(R) Core(TM) i5-1240P processor running at 1700 Mhz, backed by 16 GB of RAM, and operating on the Windows 11 platform is used for tests. The proposed model has been coded with the CPLEX solver of GAMS 44.4.0. NSGA-II has also been coded with PYTHON 3.9.

4.1. Generation of Test Problems

Eight problems are generated with four different sizes and two instances from each size. Test problems are named "number of jobs-number of operations-instance number." They are 7-3-x, 10-4-x, 12-5-x and 50-6-x respectively. In these problems, approximately 21, 40, 60, and 300 operations are scheduled, respectively. In the generated problems, the maximum number

of workers (Q) that can be assigned to a machine and the machines' maximum number (m_{iv})on which the jobs' operation is three. The machine numbers of the problems are 5 of 7-3-x, 6 of 10-4-x, 7 of 12-5-x and 8 of 50-6-x. The processing times which the number of workers is two (p_{ivk2}) are derived as regards the uniform distribution in the range of [1, 100]. For processing times where the number of workers is one (p_{ivk1}), it is obtained by multiplying p_{ivk2} with a random number between 0.5 and 0.8. For the case where the number of workers is three (p_{ivk3}), it is obtained by multiplying $p_{ivk2}$with a random number between 1.1 and 1.3. The due dates of the jobs have been derived by adapting the formula used by Saraç and Ozcelik (2023) in the unrelated parallel machine problem. The formula is given in Eq. (21).

$$d_i \sim \left[\sum_v \frac{\sum_k p_{ivk2}}{m_{iv}} \; x \; \left(1 - b_1 - \frac{b_2}{2}\right), \sum_v \frac{\sum_k p_{ivk2}}{m_{iv}} \; x \; \left(1 - b_1 + \frac{b_2}{2}\right) \right] \tag{21}$$

b_1 and b_2 in Eq. (21) are the parameters used to control the tightness of due dates. These parameter values are taken as 0.6 and 0.4, respectively.

4.2. Parameters of NSGA-II

The basic parameters of NSGA-II are crossover rate (P_c), mutation rate (P_m), population size (ps), and number of iterations (t). An experimental design is made to determine the values of these parameters. The parameters affecting the success of the algorithm are examined using the 10-4-1 problem. Factors and levels of factors are given in Table 2.

Table 2
Factors and levels of factors

	Factors			
Levels of Factors	P_c	P_m	*ps*	*t*
1	0.8	0.01	50	500
2	0.9	0.03	100	750
3	1	0.05	200	1000

Analysis of variance was performed with Minitab. The results are shown in Table 3.

Table 3
Results obtained by variance analysis

Variance analysis					
Source	*DF*	*Adj SS*	*Adj MS*	*F-Value*	*p-Value*
ps	2	236.34	118.168	19.11	0.000
P_c	2	8.98	4.490	0.73	0.485
P_m	2	27.47	13.733	2.22	0.112
t	2	37.75	18.877	3.05	0.050
Error	153	946.03	6.183		
Total	161	1256.57			

According to *p*-values in Table 3, *ps* and *t* are critical. The main effect plot of the variance analysis results is given in Fig. 4.

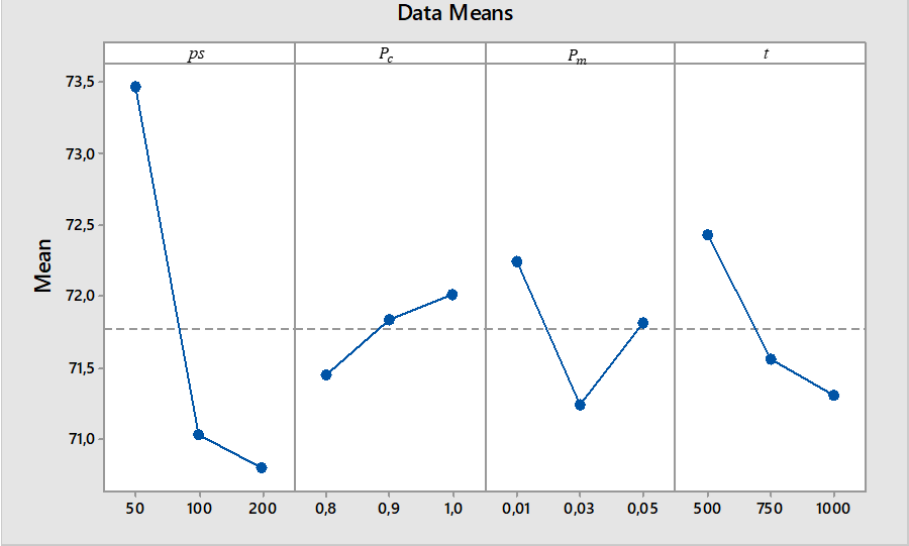

Fig. 4. The main effect plot

As can be seen from Fig. 4, *ps*, *pc*, *Pm*, and *t* should be 200, 0.8, 0.03, and 1000, respectively. In this study, these values are used.

4.3. Toy Problem

The toy problem consists of six jobs and five machines. The maximum number of workers (Q) that can be assigned to a machine is two. The number of operations (w_i) and due dates of the jobs (d_i) are given in Table 4.

Table 4
w_i and d_i values

i	w_i	d_i
1	2	63
2	2	64
3	1	38
4	2	40
5	1	17
6	2	57

The machines where the jobs' operations can be processed ($u_{ijk} = 1$) and the processing times (p_{ijkv}) according to the number of workers on these machines are given in Table 5.

Table 5
p_{ivkj} values for $u_{ijk} = 1$

i	v	k	j	p_{ivkj}	i	v	k	j	p_{ivkj}
1	1	1	1	100.1	4	1	1	3	63
1	1	1	2	77	4	1	2	1	26
1	1	1	3	53.9	4	1	2	2	20
1	1	3	1	84.5	4	1	2	3	14
1	1	3	2	65	4	2	2	1	37.7
1	1	3	3	45.5	4	2	2	2	29
1	2	1	1	58.3	4	2	2	3	20.3
1	2	1	2	53	4	2	4	1	35.2
1	2	1	3	42.4	4	2	4	2	44
1	2	2	1	39.6	4	2	4	3	35.2
1	2	2	2	33	5	1	2	1	70.8
1	2	2	3	16.5	5	1	2	2	59
1	2	3	1	63.8	5	1	2	3	35.4
1	2	3	2	58	5	1	3	1	59.4
1	2	3	3	46.4	5	1	3	2	54
2	1	4	1	115.2	5	1	3	3	27
2	1	4	2	96	5	1	4	1	13
2	1	4	3	48	5	1	4	2	10
2	2	2	1	3.9	5	1	4	3	6
2	2	2	2	3	6	1	1	1	62.7
2	2	2	3	2.1	6	1	1	2	57
2	2	3	1	25.2	6	1	1	3	39.9
2	2	3	2	21	6	2	1	1	45.5
2	2	3	3	16.8	6	2	1	2	35
3	1	1	1	72.6	6	2	1	3	17.5
3	1	1	2	66	6	2	4	1	47.3
3	1	1	3	33	6	2	4	2	43
4	1	1	1	99	6	2	4	3	34.4
4	1	1	2	90	4	1	1	3	63

The toy problem has been solved with the proposed model using all epsilon values in the range [4, 12]. The obtained pareto solutions are given in Table 6.

Table 6
The obtained pareto solutions

ε	f_1	f_2	Time (sec.)
4	311.3	4	4.55
5	281.9	5	8.39
6	186.3	6	21.83
7	164.1	7	50.08
8	121.8	8	29.73
9	96.9	9	9.00
10	71.4	10	21.89
11	51.9	11	11.34
12	33.5	12	1.31

To examine how much increasing the number of workers by one unit reduces total tardiness, the ε-constraint method has been used for solving the proposed mathematical model. This method is one of the methods for solving multi-objective problems frequently used in the literature. One of the objectives is selected as the objective function, and the other objectives are edited as constraints in the ε-constraint method. In this study, the first objective function (f_1) was chosen as the objective function, and the second objective was converted to the ε-constraint ($f_2 \leq \varepsilon$). As can be seen from Table 6, total tardiness decreases when the total number of workers increases. This shows that the two objective functions contradict each other. The graph drawn to examine the effect of each unit increase in the number of workers on the total tardiness is given in Fig. 5.

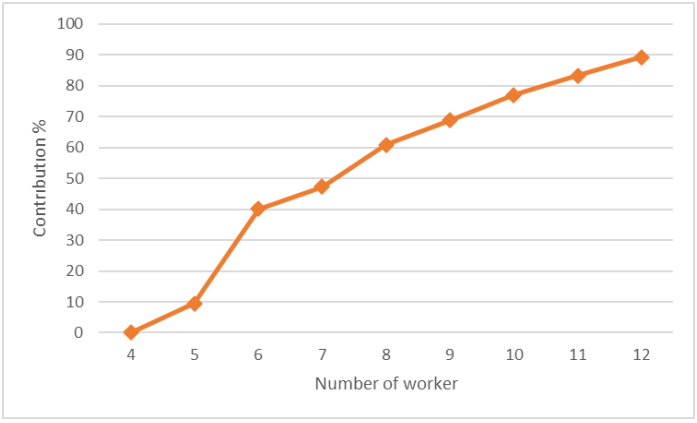

Fig. 5. The percentage contribution of jobs to the total tardiness by increasing the number of workers

As can be seen from Fig. 5, the percentage contribution at which increasing by one worker reduces the total tardiness is not always the same. For example, when the total number of workers increased from 4 to 5, the total tardiness decreased by about 10%. In contrast, when the total number of workers increased from 5 to 6, there was a significant reduction of about 30%. The total tardiness continues to decrease steadily for subsequent increases in the number of workers, but the percentage reduction has gradually reduced. When the total number of workers reaches 12, the total tardiness is reduced by 90% compared to the solution with four workers. This graph is also an essential tool for the decision-maker to determine the optimal number of workers. For example, the decision maker may adopt 6 workers because it has the largest percentage reduction.

The solution obtained for the value ε = 9 has been examined in detail. The machines where the jobs are processed, and their sequences in the machines are shown in the Gantt Chart in Fig. 6.

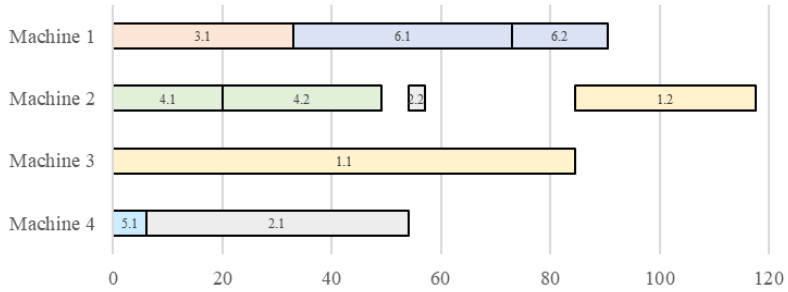

Fig. 6. Gantt Chart of the solution obtained for the value ε = 9

The number of workers on the machines is 3, 2, 1, 3 respectively. There are nine workers in total. The tardiness of the jobs is given in Table 7.

Table 7
Tardiness of the jobs

i	T_i
1	54.5
2	-
3	-
4	9
5	-
6	33.4

When Table 7 is examined, it is seen that the second job is not delayed at all, and the total tardiness is 96.9 seconds.

4.4. Test Results

All test problems have been solved with both the proposed mathematical model and NSGA-II for each ε value. CPU time is limited to 7200 seconds for GAMS/Cplex. Additionally, NSGA-II is run 10 times for each problem. The graphs of the non-dominated points obtained by both the mathematical model and NSGA-II for all test problems are given in Fig. 7. The points obtained with NSGA-II are non-dominated points among all points obtained as a result of 10 runs.

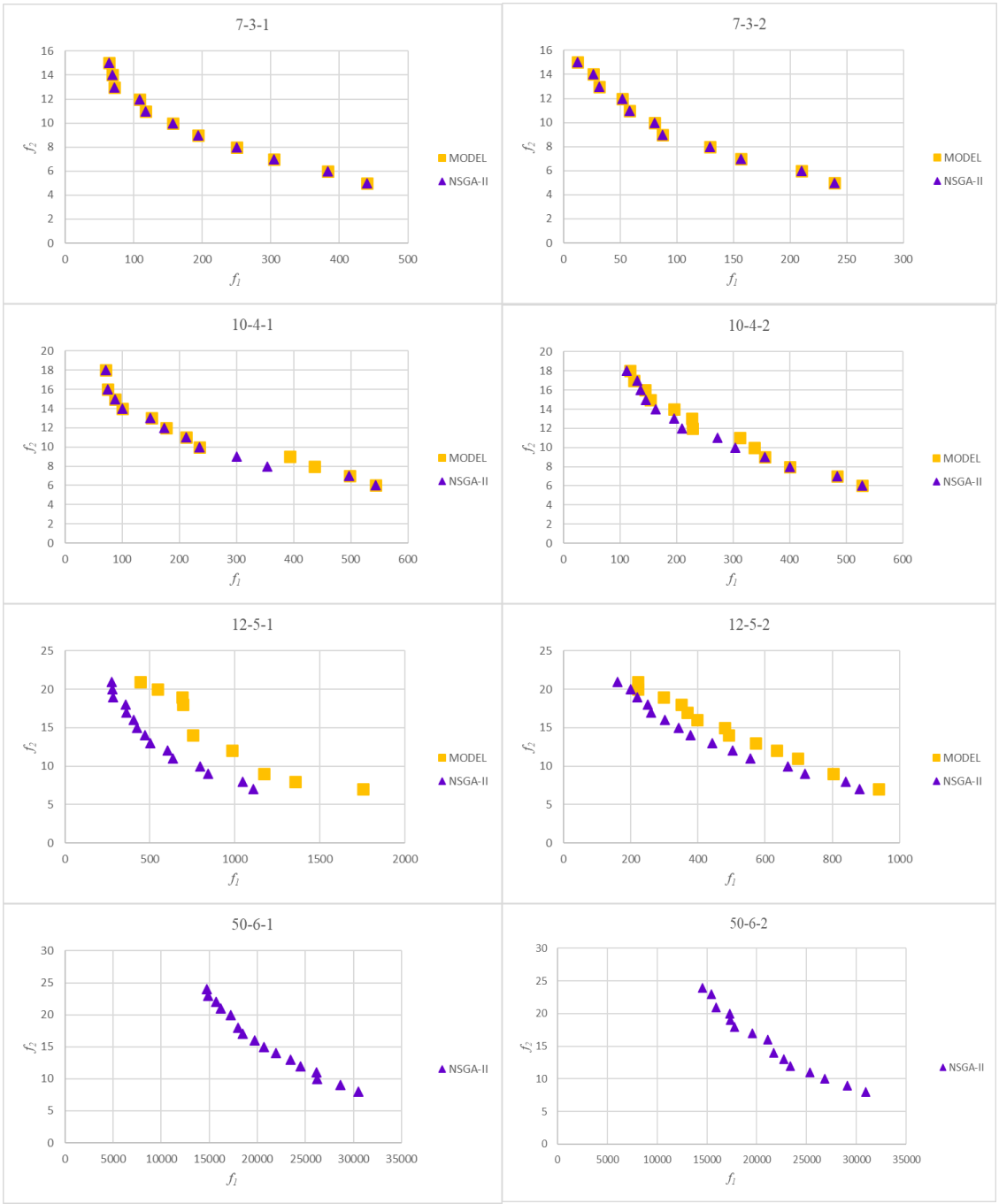

Fig. 7. The graphs of the non-dominated points

As we can see from Fig. 7, the proposed model and NSGA-II find the same points in small-size problems (*7-3-x*). The mathematical model and NSGA-II have been found the same points for some of the 10-4-x test problems, while in others, NSGA-II has been found better solutions. The most striking difference between the solutions found by the model and NSGA-II can be seen in the graph of the *12-5-x* problems. The solutions obtained with NSGA-II are more successful than the solutions obtained with the model within time limits. In large-sized problems (*50-6-x*), no feasible solution has been found with the proposed mathematical model within the time limits since the considered problem is NP-hard. However, solutions have been obtained with NSGA-II.

The CPU times of the proposed mathematical model and NSGA-II are shown in Table 8. The CPU time of the proposed mathematical model is the total time solved with all epsilon values. The CPU time of NSGA-II is also the total time of all runs. The CPU times of the proposed mathematical model are compared to the CPU times of the NSGA-II. The percentage improvement value has been calculated using the formula given in Eq. (22).

$$\%\, imp_{Model-NSGA-II} = \frac{CPU_{Model} - CPU_{NSGA-II}}{CPU_{Model}} \times 100 \tag{22}$$

Table 8
The CPU times of the proposed model and NSGA-II

Problem	CPU_{Model}*(sec.)*	$CPU_{NSGA-II}$ *(sec.)*	$imp_{Model-NSGA-II}$
7-3-1	7849.41	2777.51	%64.61
7-3-2	7218.10	1891.58	%73.79
10-4-1	81983.84	3248.18	%96.04
10-4-2	93600.00	2927.64	%96.87
12-5-1	64800.00	3770.24	%94.18
12-5-2	93600.00	2601.54	%97.22
50-6-1	-	12640.92	-
50-6-2	-	6790.67	-

When Table 8 is examined, there is an essential difference between the CPU times of the proposed model and NSGA-II. NSGA-II has been found solutions in a shorter time than the proposed model. Since the concept of time is very valuable in today's competitive environment, the success of NSGA-II is too critical to be ignored.

The most successful Pareto solution from the solutions obtained with both the mathematical model and the NSGA-II algorithm was determined using the TOPSIS method. These solutions are given in Table 9. The TOPSIS method requires the weights of the criteria when determining the best alternative. Three different weight sets were used in this study. In the first weight set f_1, in the third set of weights f_2, is three times more important than the other objective. In the second set of weights, both objectives are of equal importance.

Table 9
The most successful Pareto solution obtained with the TOPSIS method

problem	$(w_1; w_2)$	f_1	f_2	*model*	*NSGA-II*
7-3-1	(0.75; 0.25)	72.3	13	*	*
	(0.50; 0.50)	117.4	11	*	*
	(0.25; 0.75)	304.3	7	*	*
7-3-2	(0.75; 0.25)	31.4	13	*	*
	(0.50; 0.50)	58.6	11	*	*
	(0.25; 0.75)	156.6	7	*	*
10-4-1	(0.75; 0.25)	100.8	14	*	*
	(0.50; 0.50)	173.2	12		*
	(0.25; 0.75)	354.1	8		*
10-4-2	(0.75; 0.25)	145.84	15		*
	(0.50; 0.50)	163.1	14		*
	(0.25; 0.75)	400	8	*	*
12-5-1	(0.75; 0.25)	362.4	17		*
	(0.50; 0.50)	500.8	13		*
	(0.25; 0.75)	844	9		*
12-5-2	(0.75; 0.25)	161.4	21		*
	(0.50; 0.50)	301.7	16		*
	(0.25; 0.75)	719.1	9		*
50-6-1	(0.75; 0.25)	16163	21		*
	(0.50; 0.50)	26248.5	10		*
	(0.25; 0.75)	28662.1	9		*
50-6-2	(0.75; 0.25)	15923.5	21		*
	(0.50; 0.50)	23348.4	12		*
	(0.25; 0.75)	29076.3	9		*

In Table 9, the most successful Pareto solutions selected with three different weight sets with TOPSIS for each test problem are given, and the solution method by which these solutions were obtained is marked with *. As seen from the table, while successful solutions could be achieved only for small-sized problems with the mathematical model, the most successful solutions were achieved with all weight sets for all problems with the NSGA-II algorithm. The results obtained revealed the success of the proposed NSGA-II algorithm.

Furthermore, to examine the effect of the total number of workers on the total tardiness, the tardiness values obtained with the minimum and maximum number of workers from the Pareto solutions achieved by NSGA-II were compared for each test problem. The percentage contribution of the increase in the number of workers to the decrease in tardiness was calculated using the formula given in Eq. (23). The obtained values are given in Table 10.

$$contribution\ \% = \frac{tardiness\ for\ minimum\ \#worker - tardiness\ for\ maximum\ \#worker}{tardiness\ for\ minimum\ \#worker}\ 100 \tag{23}$$

Table 10
The percentage contribution in the tardiness of jobs in the results obtained with NSGA-II

problem	*minimum #worker*	*tardiness*	*maximum #worker*	*tardiness*	*contribution %*
7-3-1	5	440	15	64.2	85.41
7-3-2	5	238.9	15	12.7	94.68
10-4-1	6	543.9	18	70.8	86.98
10-4-2	6	528.1	18	112.4	78.72
12-5-1	7	1107.6	21	276.7	75.02
12-5-2	7	881.3	21	161.4	81.69
50-6-1	8	30501.2	24	14729.1	51.71
50-6-2	8	30911.3	24	14579.7	52.83

As can be seen from Table 10, when the number of workers increases, the total tardiness of jobs reduces significantly. In all test problems, the total tardiness has been reduced by an average of 75.88%.

In summary, when we solve all the test problems, the result is that the total tardiness decrease when the number of workers increases. It has advantages in solving the problems with NSGA-II. The benefits are as follows: it can find the same Pareto solutions as the model in small-sized problems, and it can solve large-scale problems and find solutions in all problem sizes in a short time.

5. Conclusion

This study considers an FJSP with both machine and worker number dependent processing times. In most studies addressing FJSPs, the worker factor is ignored. However, in businesses, considering that human labor is intensive, jobs can be completed earlier when the number of workers on the machines increases. Therefore, in this study, more than one worker can be assigned to a machine simultaneously. Processing times also depend on both the machine and the number of workers. A multi-objective mathematical model is proposed to solve the addressed problem. The objective functions are to minimize the total tardiness of jobs and total the number of workers. The NSGA-II algorithm has also been proposed for large problems since the considered problem is NP-hard. Randomly generated test problems have been solved to show the performance of the proposed solution approaches. According to the obtained solutions, as the number of workers increases, total tardiness is reduced significantly. The results of the mathematical model and NSGA-II are compared. NSGA-II finds successful solutions in a very short time to large-sized problems.

In future studies, scheduling problems with the number of workers dependent processing times in different machine environments can also be considered. Moreover, worker costs may also be included.

Acknowledgements

There is no conflict of interest with any person/institution in the paper.

References

Defersha, F.M., Obimuyiwa, D., & Yimer, A.D. (2022). Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem. *Computers & Industrial Engineering, 171*, 108487. https://doi.org/10.1016/j.cie.2022.108487

Gnanavelbabu, A., Caldeira, R.H., & Vaidyanathan, T. (2021). A simulation-based modified backtracking search algorithm for multi-objective stochastic flexible job shop scheduling problem with worker flexibility. *Applied Soft Computing, 113*, 107960. https://doi.org/10.1016/j.asoc.2021.107960

Gong, G., Deng, Q., Gong, X., & Huang, D. (2021). A non-dominated ensemble fitness ranking algorithm for multi-objective flexible job-shop scheduling problem considering worker flexibility and green factors. *Knowledge-Based Systems, 231,* 107430. https://doi.org/10.3390/app12010205

Gong, G., Chiong, R., Deng, Q., & Gong, X. (2020) A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility. *International Journal of Production Research, 58*(14), 4406-4420. https://doi.org/10.1080/00207543.2019.1653504

Gong, X., Deng, Q., Gong, G., Liu, W., & Ren, Q. (2018). A memetic algorithm for multi-objective flexible job-shop problem with worker flexibility. *International Journal of Production Research, 56*(7), 2506-2522. https://doi.org/10.1080/00207543.2017.1388933

Grumbach, F., Badr, N.E.A., Reusch, P., & Trojahn, S. (2023). A memetic algorithm with reinforcement learning for sociotechnical production scheduling. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3292548

He, Z., Tang, B., & Luan, F. (2022). An improved African vulture optimization algorithm for dual-resource constrained multi-objective flexible job shop scheduling problems. *Sensors, 23*(1), 90. https://doi.org/10.3390/s23010090

Kress, D., Müller, D., & Nossack, J. (2019). A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. *OR spectrum, 41*, 179-217. https://doi.org/10.1007/s00291-018-0537-z

Liu, C., Yao. Y., & Zhu, H. (2021). Hybrid salp swarm algorithm for solving the green scheduling problem in a double-flexible job shop. *Applied Sciences, 12*(1), 205. https://doi.org/10.3390/app12010205

Luo, Q., Deng, Q., Xie, G., & Gong, G. (2023). A Pareto-based two-stage evolutionary algorithm for flexible job shop scheduling problem with worker cooperation flexibility. *Robotics and Computer-Integrated Manufacturing, 82*, 102534. https://doi.org/10.1016/j.rcim.2023.102534

Luo, Q., Deng, Q., Gong, G., Guo, X., & Liu, X. (2022). A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm. *Expert Systems with Applications, 207*, 117984. https://doi.org/10.1016/j.eswa.2022.117984

Lou, H., Wang, X., Dong, Z., & Yang, Y. (2022). Memetic algorithm based on learning and decomposition for multiobjective flexible job shop scheduling considering human factors. *Swarm and Evolutionary Computation, 75*, 101204. https://doi.org/10.1016/j.swevo.2022.101204

Meng, L., Zhang, C., Zhang, B., & Ren, Y. (2019). Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access, 7*, 68043-68059. https://doi.org/10.1109/ACCESS.2019.2916468

Renna, P., Thürer, M., & Stevenson, M. (2020). A game theory model based on Gale-Shapley for dual-resource constrained (DRC) flexible job shop scheduling. *International Journal of Industrial Engineering Computations, 11*(2), 173-184. https://doi.org/10.5267/j.ijiec.2019.11.001

Saraç, T., & Ozcelik, F. (2023). A matheuristic algorithm for multi-objective unrelated parallel machine scheduling problem. *Journal of the Faculty of Engineering and Architecture of Gazi University, 38*(3), 1953-1966. https://doi.org/10.17341/gazimmfd.873295

Shi, J., Chen, M., Ma, Y., & Qiao, F. (2023). A new boredom-aware dual-resource constrained flexible job shop scheduling problem using a two-stage multi-objective particle swarm optimization algorithm. *Information Sciences, 643*, 119141. https://doi.org/10.1016/j.ins.2023.119141

Tan, W., Yuan, X., Wang, J., & Zhang, X. (2021). A fatigue-conscious dual resource constrained flexible job shop scheduling problem by enhanced NSGA-II: An application from casting workshop. *Computers & Industrial Engineering, 160*, 107557. https://doi.org/10.1016/j.cie.2021.107557

Vital-Soto, A., Baki, M.F., & Azab, A. (2022). A multi-objective mathematical model and evolutionary algorithm for the dual-resource flexible job-shop scheduling problem with sequencing flexibility. *Flexible Services and Manufacturing Journal*, 1-43. https://doi.org/10.1007/s10696-022-09446-x

Yan, Q., Wang, H., & Wu, F. (2022). Digital twin-enabled dynamic scheduling with preventive maintenance using a double-layer Q-learning algorithm. *Computers & Operations Research, 144*, 105823. https://doi.org/10.1016/j.cor.2022.105823

Zhang, S., Hou, T., Qu, Q., Glowacz, A., Alqhtani, S.M., Irfan, M., & Li, Z. (2022). An improved mayfly method to solve distributed flexible job shop scheduling problem under dual resource constraints. *Sustainability, 14*(19), 12120. https://doi.org/10.3390/su141912120

Zheng, X.L., & Wang, L. (2016). A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem. *International Journal of Production Research, 54*(18), 5554-5566. https://doi.org/10.1080/00207543.2016.1170226