Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

Celal Bayar University Journal of Science

A New Hybrid Scatter Search Method for Solving the Flexible Job Shop Scheduling Problems

Safa Külahlı[1], Orhan Engin[2]*, İsmail Koç[3]

[1] Management and Organization Department, Vocational School, Selçuk University, Konya, Turkey
[2] Industrial Engineering Department, Engineering and Natural Science Faculty, Konya Technical University, Konya, Turkey
[3] Software Engineering Department, Engineering and Natural Science Faculty, Konya Technical University, Konya, Turkey
* orhanengin@yahoo.com
* Orcid: 0000-0002-7250-0317

Received: 23 April 2021
Accepted: 11 November 2021
DOI: 10.18466/cbayarfbe.926756

Abstract

Flexible job shop scheduling (FJSS) is derived by inheriting the features of the job-shop scheduling problem. It has an extra routing sub-problem of the job-shop scheduling. FJSS is well known as an NP-hard problem in the literature. A new hybrid scatter search method is proposed to solve the FJSS problem. The objective function of the proposed hybrid scatter search method is minimizing the maximum completion time (C_{max}) for FJSS problems. The proposed hybrid scatter search method is integrating a local and global search for generating an initial population. The performance of the proposed new hybrid scatter search method is dependent on the selected parameters. These parameters are the size of the initial population and reference set; the number of subsets, reference set updating and population sub updating; reproduction, crossover, and mutation operators, and their ratio. A full factorial experimental design is made to determine the best values of control parameters and operators for the proposed new hybrid scatter search to solve the FJSS problems. The proposed new hybrid scatter search method is tested on a set of the well-known benchmark flexible job shop scheduling instances from the literature. The computational results indicated that the proposed new hybrid scatter search is an effective method for solving the FJSS problems.

Keywords: Flexible job shop scheduling problem, Full factorial experimental design, Hybrid scatter search method, Makespan.

1. Introduction

Job shop scheduling (JSS) is the hardest problem in this domain [1]. The flexible job shop scheduling is a much more complicated version of the JSS. In the FJSS problem, processes are permitted to be committed on any machine chosen from inside a set of available machines. The FJSS is an extremely NP-hard problem [2]. FJSS can be solved via two main approaches called integrated and hierarchical. While in the integrated approach allocating and ordering of problems are concurrently considered, in hierarchical approaches, these two operations are processed one by one [3]. Though designing integrated approaches are more difficult, their results are better than the other in general [4,5]. Brucker and Schlie [6] studied first on the FJSS problems. They developed a polynomial algorithm for FJSS problems. Kacem et al. proposed an evolutionary optimization method for solving the FJSS problem [7]. Tay and Wibowo studied the representation of the four different chromosomes for the evolutionary algorithms to solve the FJSS [8]. They indicated that the representation of the chromosome also plays an important role in the solution success of FJSS problems. Ong et al. developed an algorithm with an integrated approach attempting based on the clonal selection mechanism for solving the FJSS problems with the repetition processing [9]. Ho et al. generated a genetic architecture method for FJSS problems [10]. Gao et al. presented a hybrid genetic algorithm for FJSS problems [11]. Fattahi et al. developed a heuristic approach for solving the FJSS problems [12]. They also presented the mathematical model of the problems. Gholami and Zandieh integrated the simulated annealing and genetic algorithm approaches for the dynamic FJSS problems [13]. Xing et al. developed a search method for the

multi-objective FJSS problem [14]. Zhang et al. presented a particle swarm optimization method for multi-objective FJSS problems [15]. Bagheri et al. developed an artificial immune algorithm based on an integrated approach for FJSS problems [16]. Guohui et al. hybridized a tabu search and genetic algorithm approaches for the FJSS [17]. Wang and Yu considered an FJSS with constraints of the machine [18]. Zhang et al. proposed a genetic algorithm to solve the FJSS problems with minimizing the makespan value [1]. They determined global and local selection to produce a high-grade starting population in the stage of initialization in their method. Birgin et al. presented a MILP model for the FJSS [19]. Demir and İşleyen compiled four of the most used formulations of the FJSS and proposed a time-indexed model for FJSS [20]. Yuan and Xu proposed a hybrid differential evolution algorithm to solve FJSS with the makespan minimization criterion [21]. Demir and İşleyen considered the FJSS with overlapping in operations [22]. Abdelmaguid dealt with the makespan minimization problem in scheduling FJSS whenever there exist distinguishable sequence-dependent setup times [23]. He presented a randomized neighborhood searching function, and he experimentally specified its best parameters using modified FJSS benchmark samples. Gao et al. developed an artificial bee colony algorithm (ABCA) to schedule and reschedule with a new job(s) to minimize makespan for FJSS [24]. Besides, they proposed a new rule for initializing the bee colony populations. González et al. developed influential neighborhood structures for the FJSS containing feasibility and non-improvement circumstances, as well as procedures for fast prediction of the quality of neighbors [25]. They included these neighborhoods in a scatter search (SS) algorithm that employs tabu search and path relinking. They determined a new dissimilarity measurement to construct these meta-heuristics. Ishikawa et al. proposed a novel optimization mechanism for distributing genetic algorithms to solve FJSS problems [26]. Singh and Mahapatra developed a particle swarm optimization for solving FJSS [27]. They introduced mutation operators used commonly in genetic algorithms. Zabihzadeh and Rezaeian presented the integer linear programming model for the FJSS [28]. Li et al. proposed a hybrid ABCA for solving FJSS problems [29]. They integrated tabu search and bee colony algorithm. Shen et al. considered the FJSS problem with sequence-dependent setup times [30]. Min et al. proposed a genetic algorithm for solving the multi-objective FJSS problem with transportation constraints [31]. Li et al. generated a java algorithm for solving the FJSS problem [32]. Currently, there is no reported paper on the hybrid scatter search (HSS) method for solving the FJSS problems. The main contributions of this paper are summarized as follows:

- In this paper, the basic scatter search method is hybridized first by new strategies and new methods to obtain a good balance between exploration and exploitation in SS.
- At the proposed HSS, local and global search methods are used to determine the initial populations.
- Also, at the proposed HSS, the crossover and mutation operators and ratios are used first.
- The proposed new HSS method is the first used to solve the FJSS problems. The objective of the proposed HSS is to minimize the C_{max}.
- The proposed HSS method found the new best C_{max} values for four benchmark problems from the literature.
- A full experimental design is done for determining the best parameter sets of the new HSS method for solving the FJSS problems.
- The proposed new HSS method is tested on a set of benchmark FJSS problems.

The proposed new hybrid scatter search method (HSS) is analyzed. The results of HSS are compared to the Fattahi et al. [12], Özgüven et al. [33], Bagheri et al. [16], and Birgin et al. [19] from the literature.

The paper is organized as follows. Section 2 gives the FJSS formulation. Section 3 provides the proposed HSS method. Section 4 gives the computational results. Finally, the conclusion and future research are presented in Section 5.

2. Flexible job shop scheduling

This paper considers FJSS problems. The problem can be denoted as FJc / / C_{max}, where FJc denotes the flexible job shop with *c* work center. Each work center has several identical machines in parallel. C_{max} indicates the performance measure, makespan. The objective is to minimize the C_{max} values for FJSS problems. The FJSS problem has *m* machine and *n* job. Each job consists of a sequence of operations. The assumptions in the FJSS problems are given as follow:

- The jobs and machines numbers are known at the beginning of the schedule,
- All processors, machines, and jobs are available,
- The processing time of each job on the machine is given before,
- The processing time are including the setup time of the jobs,
- The processing sequence of each job on the machine is known previously,
- The objective function is minimizing the makespan values.

The FJSS problem is formulated as a mixed integer linear programming model as follow [12, 33, 19].

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

Parameters;

n	: Jobs number $(j = 1,...,n)$
m	: Machines number $(i = 1,...,m)$
k	: Number of operations $(k = 1,...k_j)$
$M_{j,k}$	: Job j requires operation k on Machine
$O_{j,k}$	: k th operation of the j th job
$Z_{i,j,k}$	: Capable machines set $M_{j,k}$ assigned to operation $O_{j,k}$
$t_{i,j,k}$	: Processing time of operation $O_{j,k}$ if performed on machine i $(t_{i,j,k} > 0)$
LN	: Large number
C_{max}	: Makespan
$S_{j,k}$	: Start time of operation $O_{j,k}$
p_i	: The assigned operations number on machine i; $(p = 1,...p_i)$
$SWT_{i,p}$	: Start of working time for machine i in priority p $(p = 1,...p_i)$
$T_{j,k}$	: Processing time of operation $O_{j,k}$ after select a machine

Decision variables;

$$X_{i,j,k,p} = \begin{cases} 1 & if\ O_{j,k}\ is\ performed\ on\ machine\ i\ in\ priority\ p \\ 0 & otherwise \end{cases}$$

$$Y_{i,j,k} = \begin{cases} 1 & if\ machine\ i\ is\ selected\ for\ operation\ O_{j,k} \\ 0 & otherwise \end{cases}$$

$$Z_{i,j,k} = \begin{cases} 1 & if\ O_{j,k}\ can\ be\ performed\ on\ machine\ i \\ 0 & otherwise \end{cases}$$

The mixed integer linear programming model;

The objective function is to minimize the makespan.

Min C_{max}

Subject to

$$C_{max} \geq S_{j,k_j} + T_{j,k_j} \ ; (j=1,...,n) \quad \textbf{(2.1)}$$

$$\sum_i (Y_{i,j,k})(t_{i,j,k}) = T_{j,k} \ ;(j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.2)}$$

$$S_{j,k} + T_{j,k} \leq S_{j,k+1} \ ;(j=1,...,n);\ (k = 1,...k_j\text{-}1) \quad \textbf{(2.3)}$$

$$SWT_{i,p} + (T_{j,k})(X_{i,j,k,p}) \leq SWT_{i,p+1} \ ;\ (i = 1,...,m)\ (j=1,...,n);\ (k = 1,...k_j);\ (p = 1,...p_i\text{-}1) \quad \textbf{(2.4)}$$

$$SWT_{i,p} \leq S_{j,k} + (1 - X_{i,j,k,p}).LN \ ;\ (i = 1,...,m)\ (j=1,...,n);\ (k = 1,...k_j);\ (p = 1,...p_i) \quad \textbf{(2.5)}$$

$$SWT_{i,p} + (1 - X_{i,j,k,p})LN \geq S_{j,k} \ ;\ (i = 1,...,m)\ (j=1,...,n);\ (k = 1,...k_j);\ (p = 1,...p_i) \quad \textbf{(2.6)}$$

$$Y_{i,j,k} \leq Z_{i,j,k} \ ;(i = 1,...,m);\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.7)}$$

$$\sum_j \sum_k X_{i,j,k,p} = 1 \ ;\ (i = 1,...,m);\ (p = 1,...p_i) \quad \textbf{(2.8)}$$

$$\sum_i Y_{i,j,k} = 1 \ ;(j=1,...,n);\ \ (k = 1,...k_j) \quad \textbf{(2.9)}$$

$$\sum_p X_{i,j,k,p} = Y_{i,j,k} \ ;(i = 1,...,m)\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.10)}$$

$$S_{j,k} \geq 0 \ ;\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.11)}$$

$$T_{j,k} \geq 0 \ ;\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.12)}$$

$$SWT_{i,p} \geq 0 \ ;\ (i = 1,...,m);\ (p = 1,...p_i) \quad \textbf{(2.13)}$$

$$X_{i,j,k,p} \epsilon \{0,1\} \ \ ;\ (i = 1,...,m);\ (j=1,...,n);\ (k = 1,...k_j);\ (p = 1,...p_i) \quad \textbf{(2.14)}$$

$$Y_{i,j,k} \epsilon \{0,1\} \ \ ;\ (i = 1,...,m);\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.15)}$$

$$Z_{i,j,k} \epsilon \{0,1\} \ \ ;\ (i = 1,...,m);\ (j=1,...,n);\ (k = 1,...k_j) \quad \textbf{(2.16)}$$

Constraint (2.1) determines the makespan (C_{max}). Constraint (2.2) provides the processing time of operation $O_{j,k}$. Constraint (2.3) determines each job to follow a specified operation sequence. Constraint (2.4) defines each machine to process one operation at a time. Constraints (2.5) and (2.6) denote each operation $O_{j,k}$ can be started after its assigned machine is idle. Constraint (2.7) defines the capable machines for each operation. Constraint (2.8) determines the operations to a machine and sequences assigned operations on all machines. Constraints (2.9) and (2.10) define each operation can be performed only on one machine at one priority. Constraints (2.11); (2.12) and (2.13) denote the non-negative parameters. Constraints (2.14); (2.15) and (2.16) permitted the variables $X_{i,j,k,p}$; $Y_{i,j,k}$; $Z_{i,j,k} \epsilon \{0,1\}$ or equivalently are binary.

3. Proposed hybrid scatter search method

3.1. Scatter search method

Scatter search (SS) is one of the evolutionary methods [34]. The SS method was proposed in the 1970s [35]. Contrary to other evolutionary methods, SS is established on the premise which systematic designs and methods to generate new solutions afford significant benefits. It utilizes strategies for search diversification and concentration that have verified efficiency in various optimization problems [36].

SS is based on an approach to solution generation and recombination [37]. The fundamental characteristic of

SS is the diversification of optimization solutions. The steps of the SS method are given in Fig.1.

1. An initial population is created,
2. A reference set is generated from the population,
3. A subset is selected from the reference set,
4. A combination procedure is applied to the subset,
5. An improvement procedure is applied to the combinations,
6. The reference set is updated,
7. Step 3 to Step 6 are repeated until a new reference set is needed,
8. Step 2 to Step 7are repeated until a population is needed,
9. Step 1 to Step 8 are repeated until the stopping criterion is met.

Figure 1. The steps of the SS method. [38, 39]

The SS strategy is implemented by utilizing six procedures and three stopping criteria for solving an optimization problem. These six procedures are based on five components. SS has five main methods namely diversification generator, improvement, reference set update, subset generation, and solution combination method [40, 41, 42, 43]. A diversification generator is a means to start the search procedure. Improvement method is the procedure where local search procedure is defined. The reference set update method is the procedure by which the elements of the *RefSet* are selected. The Subset generation method declares the orders to choose the pairs of parents which will be combined later to constitute new solutions combination method. The method aims to generate new solutions by using the existing solutions.

3.2. The Hybrid scatter search method

In this study, we hybridized the basic SS by new strategies and new methods to obtain a good balance between exploration and exploitation in SS. These strategies are local and global search methods to determine the initial population. This global search method is used to escape from local optimum traps in the solution space. These strategies are good for exploration. Also, the new methods for basic SS are crossover and mutation operator and ratios. These new methods are good at exploitation.

The outline of the proposed HSS is given as follows:

Step 1. Setting initial parameters of HSS
Number of the initial population,
Number of reference set,
Number of subsets,
Number of updating initial population,
Number of updating subset,
Method of crossover,
Method of mutation,
The ratio of crossover,
The ratio of mutation,
Set CPU time.
Step 2. Solving problem,
Determining initial population by the local and global search,
Generate reference set,
Select subset,
Choose two job sequences for crossover from the subset,
Do crossover
If new job sequences are feasible and the objective function is better than before;
Add they subset
Else
Eliminate they
Until crossover ratio
Choose a job sequence for mutation from the subset,
Do mutation
If the new job sequences are feasible and the objective function is better than before
Add it subset
Else
Eliminate it
Until mutation ratio
Until the number of subsets
Update the reference set
Until the number of references set
Until the stopping criterion met
Step 3. Output the minimum makespan (C_{max})

3.2.1. Initial population

The initial population is a crucial task in the scatter search algorithm [44]. In this study, two methods are presented to solve the first sub-problem by assigning each operation to a suitable machine.
The steps of the first method are given as follows [1]:

Step 1: A new array is created and starts each member to *0*.
Step 2: Choose a job randomly.
Step 3: Add the processing time and the time of the corresponding machine.
Step 4: Compare the added time to determine the shortest time.
Step 5: Set the allele which corresponds with the current process in the machine selection part to *k*.
Step 6: Add the processing time of the current chosen machine.
Step 7: Select the next process of the current job and execute Step 3 to Step 6 until all operations of the current job are chosen, then go to Step 8.
Step 8: Until all jobs are selected once, go to step 2.

A local search is used as a second method.

3.2.2. Generating reference set and subset selection

The individual in the reference set comprises selecting the best individuals in the initial population. The individuals in the reference set rank from the worst to the best according to the determinate objective function. The objective function of this study is to minimize the makespan (C_{max}). The sorted individuals are selected by starting from the best ones according to the predefined

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

reference set generation rate and the reference set is generated. The subsets are constituted by choosing the best individuals from individuals in the reference set because the combination and improvement procedures will be implemented over the best ones.

3.2.3. Crossover

Crossover is described as new individuals' generation from between two individuals selected from within the whole population according to the given rules [45]. The objective of the crossover method is to facilitate achieving the optimal result by providing the formation of better individuals. There exist different methods used in the solution of several problems. In addition to these, some methods are privately developed according to the characteristic structure of the problems. In this study, the four crossover operators are considered, namely priority-based sequential, single point, operation sequence change, and alternative machine change crossover. In the priority-based sequential crossover method, two jobs are randomly selected from the first parent individual and all the processes which belong to those jobs are selected. However, the other jobs are transferred into the child individual by selecting from the second parent individual. In the example below, 1st and 3rd jobs are randomly selected from the first parent individual (p1); the other jobs (2nd and 4th jobs) are selected from the first parent individual (p2). Then, a new child individual (c1) is generated by using p1 and p2 as shown in Fig. 2.

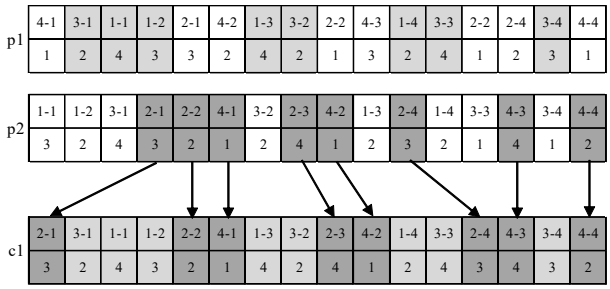

Figure 2. An example of priority-based sequential crossover operator.

Single point crossover, a point that is randomly determined on any selected individual.

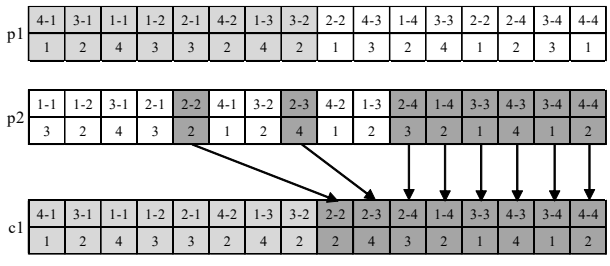

Figure 3. An example of a single-point crossover operator.

It is replaced by splitting the chromosome into two parts from that selected point. This method is used by modifying in this study.
The part of the first chromosome until the predefined cut point is transferred into the new chromosome from the first one. However, after the genes transferred from the first chromosome are extracted from the second individual, the others are transferred into the new chromosome in order. A sample of single-point crossover is shown in Fig. 3.

In the operation sequence change crossover method, it is operated on any single chromosome. Two jobs together with their whole processes are randomly selected over a single chromosome. Selected jobs are implemented by replacing over the same chromosome. An example of operation sequence change crossover is presented in Fig. 4.

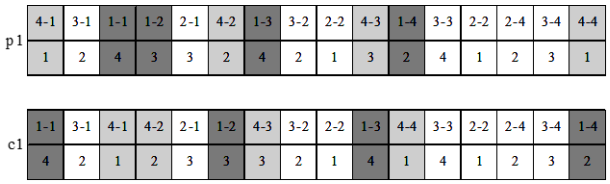

Figure 4. An example of operation sequence change crossover operator.

In the alternative machine change crossover method, the genes are randomly selected from the first chromosome. The same selected genes are chosen from the second chromosome, too. The machines of the selected genes over the second chromosome are assigned to the selected genes over the first chromosome. In sort, the method is applied by changing the machine assignment components of the same genes. An example an alternative machine change crossover is presented in Fig. 5.

3.2.4. Mutation

The mutation operator is implemented after crossover. Mutation used to provide diversity in the population is generally employed in small proportions because that mutation rate is great can destroy the good individuals.

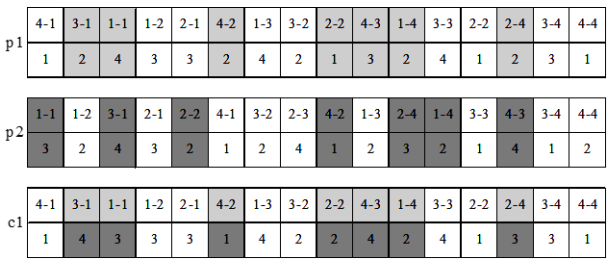

Figure 5. An example of an alternative machine change crossover operator.

In mutation, it is altered over the genes of only one individual. It is decided whether mutation will be implemented over the individual according to a defined possibility value. In this study, the four mutation operators are considered, namely random assignment machine, operation relocation, best machine change, and last operation relocation.

The random assignment machine method, a machine where a process randomly selected from the chromosome would be operated is randomly changed. This operation is performed regardless of the processing time on the machine. For example, while the machine, where the process 2-3 would be operated, is the 1st machine before mutation, the machine is changed as the 4th machine at the end of the mutation. An example of a random assignment machine is presented in Fig. 6.

Before	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	1
After	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	4

Figure 6. An example of a random assignment machine mutation operator.

In Operation Relocation Method, a process is randomly selected on the chromosome. The position of this process is changed by considering its predecessor and subsequent processes of this selected process. In the example below, the randomly selected process 1-2 is replaced with a position that is determined randomly between the process 1-1 and 1-3. A sample of operation relocation is shown in Fig. 7.

Before	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	1
After	1-1	2-1	2-2	3-1	1-2	3-2	1-3	2-3
Mutation	2	1	4	3	3	2	3	4

Figure 7. An example of operation relocation mutation operator.

In the best machine change method, a machine where a randomly selected process would be operated is changed by considering the processing time. In Fig. 8, the machine where process 2-3 would be operated is changed by detecting which machine process 2-3 is operated in the shortest time on.

Before	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	1
After	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	3

Figure 8. An example of the best machine change mutation operator.

While the present machine where this process is operating in the 1st machine, the 3rd machine was assigned to operate this process after mutation. A sample of the best machine change is presented in Fig. 8. The last process on the chromosome which is randomly selected and will be mutated is determined. The position of the determined last process is randomly replaced by considering the position of the predecessor process. A sample of the last operation relocation is given in Fig. 9.

Before	1-1	2-1	1-2	2-2	3-1	3-2	1-3	2-3
Mutation	2	1	3	4	3	2	3	1
After	1-1	2-1	1-2	2-2	3-1	2-3	1-3	3-2
Mutation	2	1	3	4	3	4	3	2

Figure 9. An example of last operation relocation mutation operator.

3.2.5. Local search and reference set updating procedure

In the local search procedure, before two chromosomes which would be crossover are subjected to the crossover method, the value C_{max} is calculated according to the available machine assignment and recorded, to perform the assignment of the machine where the processes would be operated in the shortest time according to the determined process array. Then, indices of all the machines are reset. By considering the process sequences of the chromosomes in which the index of the machine has been reset, these chromosomes are subjected to crossover.

New chromosomes generated at the end of the crossover are assigned to the machine which has the earliest completion time, by starting from the beginning of the chromosome via the local search procedure. In this procedure, for the process which would assign, the algorithm assigns the machine that has the earliest completion time by scanning all the alternative machines.

The steps of the local search procedure are given as follows:

Step 1: Reset the indices of machines in the chromosome.
Step 2: Select operations from the chromosome depending on the order of the processes.
Step 3: Determine the processing times on the alternative machines where the selected operation can be processed.
Step 4: Find the completion time of the previous operation from the selected operation.
Step 5: Find enough time spans according to the empty time span and the processing time on the alternative machine.

Step 6: Assign the machine which has the earliest completion time and enough time spans as the machine where the operation will be processed.
Step 7: Go to Step 2 until all the processes are selected.

The main objective of the local search procedure is to determine the span where the operations at the chromosome with the process sequence which occurs in the end of the crossover and mutation are earliest processed. Therefore, by assuming that the end of the previous operation is the lower limit, the whole alternative machines should be scanned for finding the machine where the operation would be earliest processed when assigning.

The solutions occurring at the end of the local search are compared to the solutions in the reference set and after new solutions are included in the reference set, the worst solutions in the reference set are removed.

3.2.6. Reference set population updating and stopping criteria

The values C_{max} obtained in the result of mutation are compared to the values in the reference set, if the new value is the better solution, by including it in the reference set the worst result is removed from the set and therefore the reference set is updated. This loop is executed until a new reference set is needed.
The generation and updating procedure of both population and the reference set included in the SS method contains 3 stopping criteria. These are given as follows [38]:

1. *New Reference Set Criterion*; contains the decision about when a new reference set should be created from the population.

2. *New Population Criterion*; contains the decision about when a new initial population should be generated.

3. *Stopping Criterion*; contains the decision about when the whole searching operations set should be stopped.

4. Computational results

4.1. Parameter test

A full factorial experimental design is made to determine the best values of control parameters and operators for the proposed new HSS to solve the FJSS problems.

The developed new HSS algorithm is examined on the benchmark instances. First studies are conducted on the problem group that belongs to Fattahi et al. [46]. Besides, the algorithm is also carried out for the problem groups formed by Kacem et al. [7,47] and Fattahi et al. [12].

The parameters used in the HSS method are utilized as fixed in all the problems. The optimization where all the levels are examined is carried out by dividing 20 parts which have Intel Xenon CPU E5-1650 3Ghz and 8GB RAM. The parameters and the ratio (levels) are given in Tables 1, 2, 3, and 4.

Table 1. The ratios of the global, local, and random methods.

Method	Ratio			
Global search	0.6	0.5	0.4	0.3
Local search	0.3	0.4	0.5	0.6
Random	0.1	0.1	0.1	0.1

The best values of global, local, and random search methods rations are found as follows:

- Global search method : 0.6
- Local search method : 0.3
- Random method : 0.1

Table 2. Parameter ratios.

Parameter	Ratio (Level)								
Initial population size	20	30	40	50	60	70	80	90	100
Size of reference set (%)	20	30	40	50	60	70	80	90	100
Number of subsets	2	4	6						
Number of reference set updating	50	100	150	200	250				
Number of population set updating	50	100	150	200	250				

Table 3. Parameter ratios for crossover methods.

Crossover Methods	Ratio								
Priority-based sequential	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Single point	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Operation sequence change	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Alternative machine change	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

Table 4. Parameter ratios for mutations methods.

Mutation methods	Ratio		
Random assignment machine	0.05	0.1	0.15
Operation relocation	0.05	0.1	0.15
Best machine change	0.05	0.1	0.15
Last operation relocation	0.05	0.1	0.15

The best parameter values are found by the full factorial experimental design. These parameters are given in Tables 5, 6, and 7.

Table 5. Initial parameter values.

Initial parameters of proposed HSS	
Size of the initial population	40
Size of reference set (%)	30
Number of subsets	6
Number of reference set updating	200
Number of population set updating	40

Table 6. The best ratios for crossover methods.

Crossover methods	Ratio
Priority-based sequential	0.6
Single point	0.2
Operation sequence change	0.2
Alternative machine change	0.7

Table 7. The best ratios for mutation methods.

Mutation methods	Ratio
Random assignment machine	0.05
Operation relocation	0.1
Best machine change	0.1
Last operation relocation	0.05

4.1.1. Performance comparison of HSS

The objective function of the proposed HSS is minimizing the maximum completion time (C_{max}). The algorithm is examined on two different benchmark sets of problems.

The first test set of the problem is SFJS and MFJS group containing 20 problems and developed by Fattahi et al. [12]. While 10 of the problems in this group consist of small jobs; the remaining 10 problems consist of major jobs. To solve these problems, Fattahi et al. [12] proposed a mathematical model and heuristic approach. For small jobs (SFJS1,…, SFJS10) their mathematical model achieved optimal solution but for major jobs did not find any optimal solutions in a reasonable time by mathematical model since FJSS problem is NP-hard. Then, Özgüven et al. [33] developed a mixed-integer linear programming model for solving FJSS problems. They used the same benchmark problems with Fattahi et al. [12]. They obtained optimal solutions for small jobs (SFJS1,…, SFJS10) like Fattahi et al. [12]. They also found for only five of major jobs problems optimal solutions in a reasonable time. Later, Birgin et al. [19] proposed a new a mixed-integer linear programming model for solving FJSS problems. They used the same benchmark problems with Fattahi et al. [12]. They obtained optimal solutions for small jobs (SFJS1,…, SFJS10) like Fattahi et al. [12] and Özgüven et al. [33]. They also found for only seven of major jobs problems optimal solutions in a reasonable time. For this reason, we proposed a new HSS method for solving the FJSS problems.

The results, in Table 8, acquired by running the algorithm are compared to the studies which are conducted for this set of problems in the literature. For all benchmark problems relative percentage deviation is calculated. The performance of the heuristic is calculated as the relative percentage deviation (RPD) with Equation 4.1 [12, 33, 16, 19].

$$\text{RPD} = \frac{\text{Heuristic Algorithm}(C_{max}) - \text{Lower Bound}}{\text{Lower Bound}} \text{x}100 \qquad \textbf{(4.1)}$$

In Equation 4.1, *Lower Bound (LB)* states the known lower bound of C_{max} obtained from the literature for benchmark problems. In Table 8, the proposed HSS method found the known best C_{max} value for eighteen problems. Only two benchmark problems, the proposed HSS couldn't find the best C_{max}.

The RPD and average relative percentage deviation (ARPD) of the proposed HSS method are compared with the Fattahi et al. [12]; Özgüven et al. [33]; Bagheri et al. [16] and Birgin et al. [19] from the literature. The ARPD is evaluated with Equation 4.2. The number of instances for each method is defined as I (L = 1,...,I) notation at Equation 4.2.

$$ARPD = \sum_{L=1}^{I} \frac{RPD}{I} \qquad \textbf{(4.2)}$$

In Table 8, the proposed HSS method found the minimum ARPD value for all twenty benchmark problems. When the obtained results are analyzed; the proposed HSS method is effective for the FJSS problems.

The proposed HSS method found the new job sequence for the MFJS9 benchmark problem The Gant schema of the MFJS9 benchmark problem is given in Fig. 10. It can be seen from Fig. 10, the best C_{max} is found 1060.

The second test instance is consisting of three different problems (K-A1, K-A2 and K-B1). The instances are developed by Kacem et al. [7,47]. These problems are also used by several studies in the literature for testing. The proposed new HSS's results are compared with the Kacem et al. [7,47], Xia and Wu [48]; Gao et al. [11] and Bagheri et al. [16] results from the literature.

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

The results are given in Table 9. It can be seen from Table 9 the proposed new HSS is found the best values of these three benchmark instances. The Gant schema of K-B1 in this problem group is presented in Fig. 11.

Table 9. The comparison of the C_{max} for the second benchmark instances.

	Problem	K-A1[a]	K-A2[b]	K-B1[c]
	*i*k*	8*8	10*10	15*10
	j	27	30	56
C_{max}	Kacem et al. [7,47]	14	7	11
	Xia and Wu [48]	15	7	12
	Gao et al. [11]	14	7	11
	Bagheri et al. [16]	14	7	11
	Proposed HSS	14	7	11

a Benchmark Problems 8*8 [7]

b Benchmark Problems 10*10 [7]

c Benchmark Problems 15*10 [47]

The proposed HSS is compared with the Kacem and SFJS-MFJS benchmark instances from the literature. The results are given in Tables 10 and 11.

Table 10. The performance of the proposed new HSS compared with Kacem benchmark problems.

Algorithm	Proposed new HSS (C_{max})		
	Equal	Better	Total
Kacem et al. [7,47]	3		3
Xia and Wu [48]	1	2	3
Gao et al. [11]	3		3
Bagheri et al. [16]	3		3

In Table 10, the proposed new HSS method is given a better makespan (C_{max}) for two problems according to the Xia and Wu [48] method. Also, the proposed new HSS method is given equal makespan (Cmax) for eleven problems according to Kacem et al [7,47], Xia and Wu [48], Gao et al. [11], and Bagheri et al. [16] methods from the literature.

It can be seen in Table 11, the proposed new HSS method is given the better makespan (C_{max}) for 77 instances according to the Fattahi et al. [12]; Özgüven et al. [33]; Bagheri et al. [16], and Birgin et al. [19]'s methods. Also, the proposed new HSS method is given equal makespan (C_{max}) for 103 instances according to the Fattahi et al. [12]; Özgüven et al. [33]; Bagheri et al. [16], and Birgin et al. [19]'s methods from the literature.

Table 11. The performance of the proposed new HSS compared with SFJS-MFJS benchmark problems.

Algorithm		Proposed new HSS (C_{max})		
		Equal	Better	Total
[12]	HSA/SA[a]	10	10	20
	HSA/TS[b]	10	10	20
	HTS/TS[c]	10	10	20
	HTS/SA[d]	9	11	20
	ISA[e]	9	11	20
	ITS[f]	9	11	20
[33]	MILP[g]	14	6	20
[16]	AIA[h]	14	6	20
[19]	AMILP[i]	18	2	20

a Hierarchical approach and SA heuristic for assignment problem and SA heuristic for sequencing problem.

b Hierarchical approach and SA heuristic for assignment problem and TS heuristic for sequencing problem.

c Hierarchical approach and TS heuristic for assignment problem and TS heuristic for sequencing problem.

d Hierarchical approach and TS heuristic for assignment problem and SA heuristic for sequencing problem.

e Integrated approach with simulated annealing heuristic

f Integrated approach with tabu search heuristic

g Mixed-integer linear programming

h Artificial immune algorithm

i A mixed-integer linear programming

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

Table 8. The comparison results of the first twenty benchmark instances from the literature.

Problem	i,j,k	LB	Fattahi et al. (2007)																Özgüven et al. (2010)			Bagheri et al. (2010)			Birgin et al. (2013)			Proposed HSS		
			HSA/SA[a]		RPD	HSA/TS[b]		RPD	HTS/TS[c]		RPD	HTS/SA[d]		RPD	ISA[e]		RPD	ITS[f]		RPD	MILP[g]		RPD	AIA[h]		RPD	AMILP[i]		RPD	HSS		RPD
			C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU		C_{max}	CPU	
SFJS1	**2*2*2**	66	66	12	-	66	1	-	66	1	-	66	2	-	66	25	-	66	1	-	66	0.02	-	66	0.03	-	66	0	-	66	1	-
SFJS2	**2*2*2**	107	107	13	-	107	1	-	107	1	-	107	3	-	107	35	-	107	1	-	107	0	-	107	0.03	-	107	0.01	-	107	1	-
SFJS3	**3*2*2**	221	221	14	-	221	1	-	221	1	-	221	5	-	221	40	-	221	1	-	221	0.02	-	221	0.04	-	221	0.05	-	221	1	-
SFJS4	**3*2*2**	355	355	14	-	355	1	-	355	1	-	355	7	-	355	45	-	355	1	-	355	0	-	355	0.04	-	355	0.02	-	355	1	-
SFJS5	**3*2*2**	119	119	14	-	119	2	-	119	1	-	119	9	-	119	50	-	119	1	-	119	0.06	-	119	0.04	-	119	0.04	-	119	1	-
SFJS6	**3*3*2**	320	320	18	-	320	3	-	320	1	-	320	7	-	320	50	-	320	2	-	320	0.03	-	320	0.04	-	320	0.01	-	320	1	-
SFJS7	**3*3*5**	397	398	19	-	398	4	-	398	1	-	398	9	-	398	55	-	398	2	-	398	0.02	-	398	0.04	-	398	0	-	398	1	-
SFJS8	**3*3*4**	253	253	17	-	253	5	-	253	2	-	256	10	**1.2**	253	35	-	253	2	-	253	0.02	-	253	0.05	-	253	0.04	-	253	1	-
SFJS9	**4*3*5**	210	210	19	-	210	6	-	210	2	-	210	11	-	215	55	**2.3**	215	3	**2.3**	210	0.03	-	210	0.05	-	210	0.01	-	210	1	-
SFJS10	**5*3*5**	516	516	21	-	516	7	-	516	4	-	516	10	-	516	55	-	516	3	-	516	0.02	-	516	0.05	-	516	0.02	-	516	1	-
MFJS1	5*3*6	396	479	22	**21.0**	491	55	**24.0**	469	15	**18.4**	469	30	**18.4**	488	60	**23.2**	548	9	**38.4**	468	0.44	**18.2**	468	9.23	**18.2**	468	0.26	**18.2**	468	7	**18.2**
MFJS2	5*3*7	396	495	62	**25.0**	482	55	**21.7**	482	12	**21.7**	468	30	**18.2**	478	60	**20.7**	457	8	**15.4**	446	6.49	**12.6**	448	9.35	**13.1**	446	0.87	**12.6**	446	24	**12.6**
MFJS3	6*3*7	396	553	82	**39.6**	538	75	**35.9**	533	20	**34.6**	538	50	**35.9**	599	107	**51.3**	606	8	**53.0**	466	4.14	**17.7**	468	10.06	**18.2**	466	1.66	**17.7**	466	32	**17.7**
MFJS4	7*3*7	496	656	102	**32.3**	650	85	**31.0**	634	27	**27.8**	618	80	**24.6**	703	195	**41.7**	870	9	**75.4**	564	1779.03	**13.7**	554	10.54	**11.7**	554	27.43	**11.7**	554	253	**11.7**
MFJS5	7*3*7	414	650	105	**57.0**	662	110	**59.9**	625	40	**51.0**	625	64	**51.0**	674	240	**62.8**	729	10	**76.1**	514	50.98	**24.2**	527	10.61	**27.3**	514	4.55	**24.2**	514	219	**24.2**
MFJS6	8*3*7	469	762	125	**62.5**	785	130	**67.4**	717	96	**52.9**	730	102	**55.7**	856	330	**82.5**	816	50	**74.0**	635	3600	**35.4**	635	22.18	**35.4**	634	52.48	**35.2**	634	180	**35.2**
MFJS7	8*4*7	619	1020	197	**64.8**	1081	290	**74.6**	964	129	**55.7**	947	190	**53.0**	1066	480	**72.2**	1048	240	**69.3**	935	3600	**51.1**	879	24.82	**42.0**	879	1890	**42.0**	879	6000	**42.0**
MFJS8	9*4*7	619	1030	230	**66.4**	1122	325	**81.3**	970	405	**56.7**	922	182	**48.9**	1328	610	**114.5**	1220	370	**97.1**	905	3600	**46.2**	884	26.94	**42.8**	884	3600	**42.8**	884	7200	**42.8**
MFJS9	11*4*8	764	1180	330	**54.5**	1243	660	**62.7**	1105	660	**44.6**	1105	330	**44.6**	1148	840	**50.3**	1124	680	**47.1**	1192	3600	**56.0**	1088	30.76	**42.4**	1137	3600	**48.8**	1060	7200	**38.7**
MFJS10	12*4*8	944	1538	425	**62.9**	1615	600	**71.1**	1404	960	**48.7**	1384	430	**46.6**	1546	850	**63.8**	1737	763	**84.0**	1276	3600	**35.2**	1267	30.94	**34.2**	1251	3600	**32.5**	1208	7200	**28.0**
ARPD					**48.59**			**52.96**			**41.22**			**39.69**			**58.30**			**62.98**			**31.02**			**28.53**			**28.57**			**27.10**

[a] Hierarchical approach and simulated annealing (SA) heuristic for assignment problem and SA heuristic for sequencing problem.

[b] Hierarchical approach and SA heuristic for assignment problem and tabu search (TS) heuristic for sequencing problem.

[c] Hierarchical approach and TS heuristic for assignment problem and TS heuristic for sequencing problem.

[d] Hierarchical approach and TS heuristic for assignment problem and SA heuristic for sequencing problem.

[e] Integrated approach with SA heuristic.

[f] Integrated approach with TS heuristic.

[g] Mixed-integer linear programming

[h] Artificial immune algorithm

[i] A mixed-integer linear programming

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 347-359
Doi: 10.18466/cbayarfbe.926756

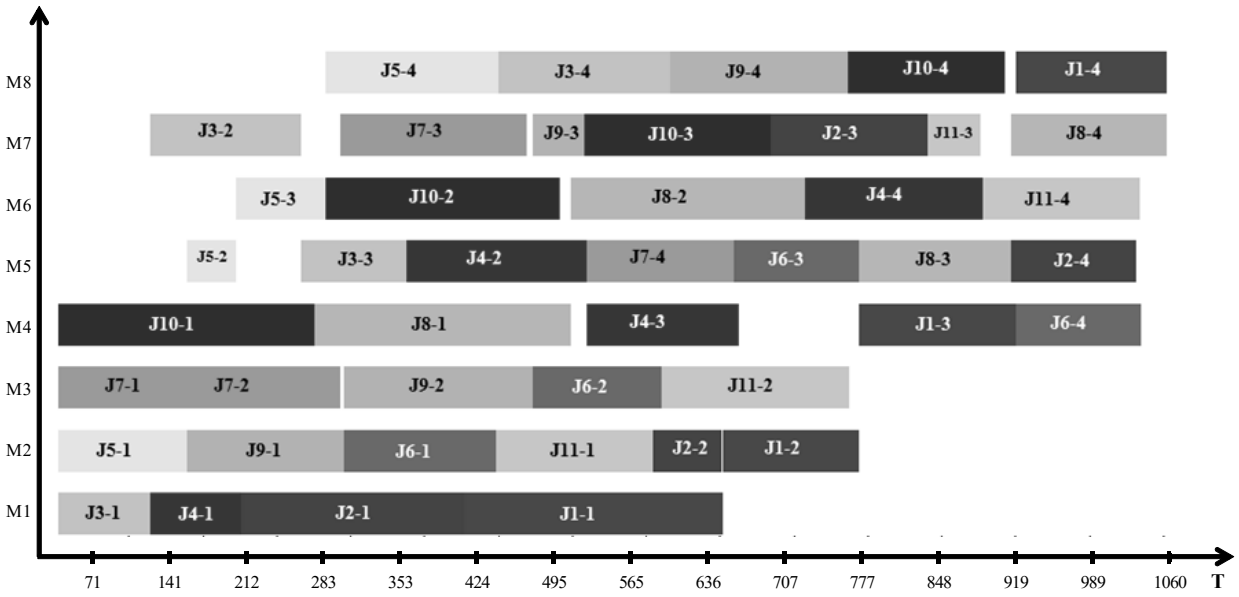

Figure 10. Gantt Schema of the MFJS9 instance.

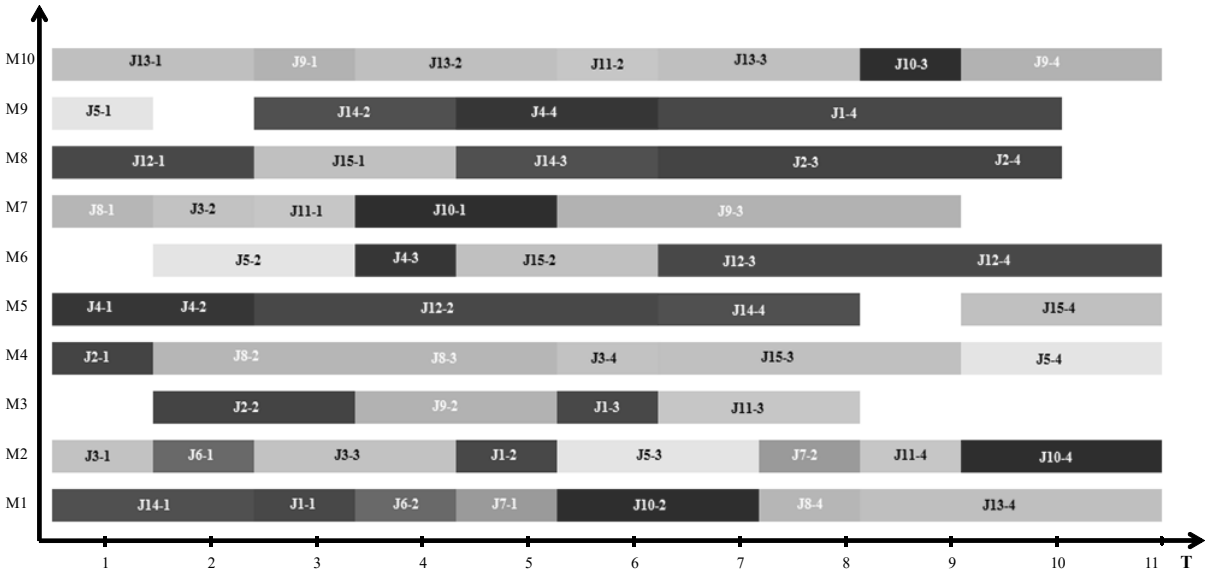

Figure 11. The Gant schema of K-B1.

5. Conclusion and future research

The flexible job shop is extra routing sub-problems of the job shop scheduling problems. The FJSS is known as an NP-hard problem from the literature. In this study, a new HSS method is proposed to solve the FJSS problems. The proposed HSS method is iterated the local and global search method for the initial population. The HSS method is consisting of the initial population, reference set, subset, reference set updating, population sub updating, reproduction, crossover, mutation operators, and their ratio. To determine the best parameter set of the proposed HSS for solving the FJSS problems a full factorial experimental design is made. The performance of the proposed HSS method is examined on the benchmark problems. There exist several problems with various sizes in the literature. First, the examinations on the problem groups consisting of 20 examples and developed by Fattahi et al. [12]. The proposed HSS method is found more efficient results for these problems. The second test problem is developed by Kacem et al. consisting of 3 instances [7,47]. When analyzed the results of the tests. The developed new HSS method is seen to produce efficient results on these benchmark instances. According to the computational results, the proposed new HSS method is influential in terms of reduced makespan for the FJSS problems. The proposed new

HSS is an efficient problem-solving technique for FJSS problems.

For future research, the proposed HSS heuristics may be used for multi-objective FJSS problems.

Author's Contributions

Safa KÜLAHLI: Methodology, Data curation, Writing-original draft, Visualization, Investigation
Orhan ENGİN: Supervision, Methodology, Validation, Writing-original draft, Investigation, Conceptualization.
İsmail KOÇ: Methodology, Software, Writing-original draft, Visualization.

Ethics

There are no ethical issues after the publication of this manuscript.

References

1. Zhang, G, Gao, L, Shi, Y. 2011. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert System with Application*; 38: 3563-3573.

2. Yazdani, M, Amiri, M, Zandieh, M. 2010. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert System with Application*; 37: 678-687.

3. Rossi, R, Tarim, SA, Hnich, B, Prestwic, S, Karacaer, S. 2010. Scheduling internal audit activities: a stochastic combinatorial optimization problem. *Journal of combinatorial optimization;* 19: 325- 346.

4. Karimi, H, Rahmati, SHA, Zandieh, M. 2012. An efficient knowledge-based algorithm for the flexible job shop scheduling problem. *Knowledge-Based System*; 36: 236-244.

5. Hwang, S, Cheng, ST. 2001. Combinatorial optimization in real-time scheduling: Theory and Algorithms. *Journal of combinatorial optimization;* 5: 345- 375.

6. Brucker, P, Schlie, R. 1990. Job-shop scheduling with multi-purpose machines. *Computing*; 45: 369-375.

7. Kacem, I, Hammadi S, Borne P. 2002. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics*; 32: 1-13.

8. Tay, J, Wibowo, D. 2004. An Effective Chromosome Representation for Evolving Flexible Job Shop Schedules. In: Genetic and Evolutionary Computation GECCO - Eds: Deb K: Springer Berlin Heidelberg, 210-221.

9. Ong, Z, Tay, J, Kwoh, C. 2005. Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules, Artificial Immune Systems. Eds: Jacob C, Pilat M, Bentley P, Timmis J: Springer Berlin Heidelberg, 442-455.

10. Ho, N, Tay, J, Lai, E. 2007. An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*; 179: 316-333.

11. Gao, J, Sun, L, Gen, M. 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computer Operation Research*; 35: 2892-2907.

12. Fattahi, P, Mehrabad, MS, Jolai, F. 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*; 18: 331-342.

13. Gholami, M, Zandieh, M. 2008. Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. *Journal of Intelligent Manufacturing*; 20: 481-498.

14. Xing, L, Chen, YW, Zhao, Q, Xiong, J. 2009. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing;* 10: 888-896.

15. Zhang, G, Shao, X, Li, P, Gao, L. 2009. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*; 56: 1309-1318.

16. Bagheri, A, Zandieh, M, Mahdavi, I, Yazdani, M. 2010. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Generation Computer Systems*; 26: 533-541.

17. Guohui, Z, Liang, G, Yang, S. 2010. Genetic algorithm and tabu search for multi objective flexible job shop scheduling problems. *International Conference on Computing, Control, and Industrial Engineering (CCIE)*; 251-254.

18. Wang, S, Yu, J. 2010. An effective heuristic for flexible job-shop scheduling problem with maintenance activities. *Computers & Industrial Engineering;* 59: 436-447.

19. Birgin, EG, Feofiloff, P, Fernandes, CG, EL. de Melo, Oshiro MTI, Ronconi DP. 2013. A MILP model for an extended version of the Flexible Job Shop Problem. *Optimization Letters*; 8: 1417-1431.

20. Demir, Y, İşleyen, SK. 2013. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*; 37: 977-988.

21. Yuan, Y, Xu, H. 2013. Flexible job shop scheduling using hybrid differential evolution algorithms. *Computers & Industrial Engineering*; 65: 246-260.

22. Demir, Y, İşleyen SK. 2014. An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations. *International Journal of Production Research*; 52: 3905-3921.

23. Abdelmaguid, TF. 2015. A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times. *Applied Mathematical Computing;* 260: 188-203.

24. Gao, KZ, Suganthan, PN, Chua, TJ, Chong, CS, Cai, TX, Pan, QK. 2015. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert System Applied*; 42: 7652-7663.

25. González, MA, Vela, CR, Varela, R. 2015. Scatter search with path relinking for the flexible job shop scheduling problem. *European Journal Operation Research;* 245: 35-45.

26. Ishikawa, S, Kubota, R, Horio, K. 2015. Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem. *Expert System with Application;* 42: 9434-9440.

27. Singh, MR, Mahapatra, SS. 2016. A quantum behaved particle swarm optimization for flexible job shop scheduling. *Computers & Industrial Engineering;* 93: 36-44.

28. Zabihzadeh, SS, Rezaeian, J. 2016. Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time. *Applied Soft Computing*; 40: 319-330.

29. Li, X, Peng, Z, Du, B, Guo, J, Xu, W, Zhuang. 2017. Hybrid artificial bee colony algorithm with a rescheduling strategy for solving

flexible job shop scheduling problems. *Computers and Industrial Engineering;* 113: 10- 26.

30. Shen, L, Dauzère-Pérès, S, Neufeld, JS. 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*; 265: 503-516.

31. Min, D, Dunbing, T, Adriana, G, Salido, MA. 2019. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics-Integrated Manufacturing*; 59: 143-157.

32. Li, JQ, Deng, JW, Li, CY, Han, YY, Tian, J, Zhang, B, Wang, CG. 2020. An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowledge-Based Systems;* 200: 106032.

33. Özgüven, C, Özbakır, L, Yavuz, Y. 2010. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*; 34: 1539-1548.

34. Engin, O, Yılmaz, MK, Baysal, ME, Sarucan, A, 2013. Solving fuzzy job shop scheduling problems with availability constraints using a scatter search method. *Journal of Multi-Valued Logic and Soft Computing*; 21: 317-334.

35. Engin, O, Kahraman, C, Yılmaz, MK, 2009. A Scatter Search Method for Multi Objective Fuzzy Permutation Flow Shop Scheduling Problem: A Real-World Application, 169-189, Computational Intelligence in Flow Shop and Job Shop Scheduling, Springer, Uday K. Chakraborty (Ed.), ISBN:978-3-642-02836-6.

36. Marti, R. 2003. Principles of Scatter Search, Leeds School of Business, University of Colorado, Campus Box 419, Boulder, CO.

37. Naderi, B, Ruiz, R. 2014. A scatter search algorithm for the distributed permutation flow shop scheduling problem. *European Journal Operation Research*; 239: 323-334.

38. Cano, DB, Santana, JB, Rodriguez, CC, DelAmo IJG, Torres MG, Garcia, FJM, Batista, BM, Perez, JAM, Vega, JMM, Martin, RR. 2004. Nature-inspired components of the Scatter Search, Technical Report.

39. Oktay, S, Engin, O. 2006. Scatter search method for solving industrial problems: literature survey. *Journal of Engineering and Natural Sciences*; 3: 144- 155.

40. Glover, F, 1998. A template for scatter search and path relinking, *Artificial Evolution*; 1363: 3-51.

41. Glover, F, Laguna, M, Marti, R. 2000. Fundamentals of scatter search and path relinking. *Control Cybernetics;* 29: 653-684.

42. Marti, R. 2006. Scatter search - Wellsprings and challenges. *European Journal Operation Research;* 169: 351-358.

43. Marti, R, Laguna, M, z, F. 2006. Principles of scatter search. *European Journal Operation Research;* 169: 359-372.

44. Engin, O, Ceran, G, Yılmaz, MK. 2011. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing;* 11(3): 3056-3065.

45. Kahraman, C, Engin, O, Kaya, I, Yılmaz, MK. 2008. An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *International Journal of Computational Intelligence Systems;* 1(2): 134- 147.

46. Fattahi, P, Jolai, F, Arkat, J. 2009. Flexible job shop scheduling with overlapping in operations. *Applied Mathematical Modelling*; 33: 3076-3087.

47. Kacem, I., Hammadi, S., Borne, P. 2002. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation*, 60(3-5): 245-276.

48. Xia, W, Wu, Z, 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*; 48(2): 409-425.