Computer Science • 22(4) 2021 https://doi.org/10.7494/csci.2021.22.4.4130

Amin Rezaeipanah
Fariba Sarhangnia
Mohammad Javad Abdollahi

META-HEURISTIC APPROACH BASED ON GENETIC AND GREEDY ALGORITHMS TO SOLVE FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

Abstract *Job-shop scheduling systems are one of the applications of group technology in industry, the purpose of which is to take advantage of the physical or operational similarities of products in their various aspects of construction and design. Additionally, these systems are identified as cellular manufacturing systems (CMS). In this paper, a meta-heuristic method that is based on combining genetic and greedy algorithms has been used in order to optimize and evaluate the performance criteria of the flexible job-shop scheduling problem. In order to improve the efficiency of the genetic algorithm, the initial population is generated by the greedy algorithm, and several elitist operators are used to improve the solutions. The greedy algorithm that is used to improve the generation of the initial population prioritizes the cells and the job in each cell and, thus, offers quality solutions. The proposed algorithm is tested over the P-FJSP dataset and compared with the state-of-the-art techniques of this literature. To evaluate the performance of the diversity, spacing, quality, and run-time criteria were used in a multi-objective function. The results of the simulation indicate the better performance of the proposed method as compared to the NRGA and NSGA-II methods.*

Keywords job-shop scheduling, meta-heuristic method, genetic algorithm, greedy algorithm, jobs priority

Citation Computer Science 22(4) 2021: 463–488

1. Introduction

A cellular manufacturing system (CMS) is an effective system for the economical manufacturing of pieces in industrial units. The design of this system is considered to be an application of group technology [27]. The design involves the process of making a set of similar pieces, which is done by a group of machines that are dedicated to the cell. Cell formation is the first and most important step in implementing the CMS. Categorizing the machinery and pieces and forming manufacturing cells and family pieces are accomplished in the stage of cell formation. The problem of cell formation and manufacturing scheduling are two important steps in the implementation of these systems [22]. CMSs are in fact the application of group technology in the field of manufacturing, the purpose of which is to classify the pieces and machines in such a way that their physical or operational similarities in different aspects of manufacturing and design are used [17]. On the other hand, cellular manufacturing is a process of manufacturing that is a subsection of just-in-time manufacturing and lean manufacturing that encompass group technology.

The scheduling problem for a finite number of jobs has been widely considered in a great deal of research. However, jobs are sometimes repeated infinitely over time in the real world. In this situation, jobs arrive successively to a shop with some time intervals. This condition is generally considered in two major categories: real--time, and cyclic scheduling problems [29]. The real-time models usually consider the scheduling of a set of simple tasks (single-operation jobs) with a regular or irregular repeating pattern. In this category, a due date is determined for each task, which must be satisfied in the scheduling model [22, 27], whereas the considered tasks in real-time models are single-operation; this is not often suitable for job-shop environments that include jobs with multiple operations. On the other hand, cyclic job-shop scheduling models are able to schedule high-variety repeatable jobs in a job-shop environment.

The problem with job-shop manufacturing scheduling is finding an optimal sequence for performing different operations and finding optimal sequences that are related to each machine in each cell (as well as determining the optimal sequence of the cells themselves). These problems are usually addressed with the aim of minimizing the length of the schedule; the timing of implementing operations is fixed and predetermined in them. The problem of cell manufacturing scheduling is one of the NP-hard problems [15]. Due to the intrinsic complexity of compositional optimization problems (and especially the problem of cell manufacturing timing), the use of innovative and meta-functional methods to solve such problems makes effective improvements in the manufacturing of acceptable responses. As the dimensions of these problems extend, traditional methods of determining the optimal response will, in fact, lose their effectiveness due to taking too much time [5].

In order to meet the needs of different industries in today's competitive world, manufacturing systems with improved performance must be used. In this regard, the automation and flexibility of machines are insufficient; systems with appropriate timing must be created [5,15]. The problem of CMS scheduling is among the combined

optimization problems, which; extensive studies have been conducted to solve the problems of the CMS due to the applicability of this type of problem [5,30,32,33]. In many of the conducted studies, artificial intelligence-optimization algorithms such as genetics are used to solve this type of problem [2,28,30]. Generally, designing a CMS consists of four steps [30]. The first (and foremost) step is solving a cell-formation problem. At this stage, pieces that have similarities in their shapes and configurations and are produced by the same or similarly required machines are considered to be in the same family to be processed by a group of machines located in one cell. In the second step, facility layout is determined; this involves cell layouts at the level of a workshop as well as machines within each cell. The third step is about scheduling the operation on each machine in each cell. Finally, the fourth step is the allocation of resources in which such resources as tools, manpower, and materials are assigned to the machines and cells. Family formation for the parts and grouping machines have some advantages, such as reducing setup times, material handling costs, work--in-process inventories, throughput times, and production costs [2].

This paper describes a method for the layout design of a cellular manufacturing system (CMS) that would simultaneously allow for the grouping of machines that are unique to a part family into cells as well as those shared by several cells to be located together in functional sections. In this paper, the focus is on the use of a hybrid multi-functional algorithm in multi-objective mode to solve this problem. Since one of the most important components in decision-making is time in all cases of optimization, objective functions such as minimizing times to complete jobs, tardiness costs, earliness costs, etc., are going to be used in the case of a CMS.

In the remainder of this paper, we state a literature review in Section 2. In Section 3, the problem of job-shop manufacturing scheduling will be reviewed. Section 4 presents our proposed method. The comparison work and results of evaluating the proposed method are presented in Section 5. Finally, our conclusions and suggestions are presented in Section 6.

2. Literature review

A study on the problem of job-shop scheduling was first implemented by Johnson; it is believed that he was one of the founders of time scheduling theory. He proposed an optimal algorithm for a two-machine workflow job-shop problem and extended this algorithm to the job-shop work schedule problem. After this, Akers (1956) used a Boolean algebraic method to represent the processing sequence. The scientific principles of group technology were established by Epitz (1958) in Germany, and various definitions and interpretations have been made of these principles [22]. Lopez considered group technology to be a new manufacturing philosophy in 1998, eliminating the disadvantages of both the custom-made and mass-produced manufacturing philosophies [27].

Reddy et al. examined the timing and sequence of operations of a CMS and proposed an innovative method for scheduling operations [23]. Karsikian et al. examined the problem of group technology design in the scheduling of a CMS us-

ing meta-heuristic algorithms [13]. In this method, a variety of products are grouped by using a custom cluster in each cell, and then the optimization problem of the scheduling in the cells is developed. Another study that was conducted in the field of job-shop manufacturing planning was presented by Pezla and his colleagues. They introduced a genetic algorithm that combined several strategies to generate an initial population; this algorithm is selected for remanufacturing.

In [6], the comparison of CMS design methods was investigated. This study focused on designing a CMS and minimizing the number of cells. The results showed that meta-functional methods performed better with respect to the average flow time and inventory under construction for most designs. Yazi et al. examined the scheduling of cell manufacturing by considering the preparation times that were dependent on cell sequence; their goal was to minimize the average total of flow time for all of the tardiness [17]. The method of this study was based on a mathematical model, and a method of searching for a harmonic hybrid was proposed. In another study, Kia et al. presented a mathematical model for evaluating the effects of feature segmentation on a dynamic CMS [14]. In this research, a two-objective model and Pareto front were used. Jalilvand and Fattahi proposed combining a mathematical model and a genetic algorithm to solve the problem of flexible job-shop manufacturing scheduling [12]. In this method, a linear mixed programming model is used to schedule jobs; its aim was to minimize the tardiness costs, setups, and maintenance costs for small-scale problems. To solve large-scale problems, the mathematical model was combined with a genetic algorithm. In [18], a case study was developed on the ability of genetic algorithms to solve the problem of the time-latency of job-shop manufacturing. The results of this study showed that the use of the genetic algorithm prolonged the processing time but ultimately resulted in lowest labor costs. Mahmoudian et al. proposed a new intelligent particle optimization algorithm to solve the problem of cell manufacturing that was based on neural computations and applications [19]. Using this approach to reconstruct the configuration of a real company in the agricultural manufacturing sector showed reasonable results.

The NRGA algorithm was developed by Jadan et al.; it is a genetic algorithm of the defeated class that is meant to solve the problem of job-shop manufacturing [11]. This algorithm uses a parametric penalty method to search for a set of the best Pareto solutions. The NSGA-II algorithm was proposed by Ahmadi et al., which was an expanded version of the NRGA algorithm [1]. The main difference between NSGA-II and NRGA is in the selection process: in NSGA-II, it uses the competition selection algorithm, while NRGA applies the roulette wheel algorithm. Also in NSGA-II, the simultaneous improvement of the two goals of makespan (completion time the jobs) and stability through the Pareto front was considered.

3. Problem of job-shop manufacturing scheduling

The problem of job-shop manufacturing scheduling includes two stages of timing. In the first stage, the sequence of cells is determined; in the second, the sequence of jobs

is determined. The main goal in this problem was to minimize the maximum time to complete the work. In this case, there is a piece (work) (P) that must be applied to the machines (M). The machines are located in a separate cell (C), which creates time for the pieces to move between the cells. In addition, t_{ij} is the processing time of the i-th piece on the j-th machine in this case. a_{ij} shows binary if the i-th piece needs a j-th machine to process ($a_{ij} = 1$); otherwise, $a_{ij} = 0$.

In the mathematical model of the problem of job-shop manufacturing scheduling, inputs must ensure that each piece is allocated to only one cell. In Equation (1), this constraint is guaranteed:

$$\sum_{c \in C} x_{ic} = 1, \qquad \forall\, i \in P \tag{1}$$

where P indicates the number of pieces, and x_{ic} is a binary variable for the decision-making. If piece i is assigned to cell c, it is $x_{ic} = 1$; otherwise, $x_{ic} = 0$.

In a feasible solution, the pieces that are assigned to a cell must be present only in the sequence of the same cell. In Equation (2), this constraint is guaranteed:

$$\sum_{k} z_{ikc} = x_{ic}, \qquad \forall\, i \in P,\ \forall\, c \in C \tag{2}$$

where z_{ikc} is a binary variable for the decision-making. If piece i is present in the k-th sequence of cell c, $z_{ikc} = 1$; otherwise, $z_{ikc} = 0$.
The completion time of each piece can be calculated as Equation (3):

$$\begin{aligned} c_i = \max \quad & \Big(\forall\, k \in K,\ j \in M,\ c \in C \& b \in K_c : c(k,j,c,b) \cdot z_{ikc} \cdot y_{cb} \\ & + \textstyle\sum_{c=1}^{c} \big(\sum_{j=1}^{M} a_{ij} \cdot |a_{ij} - a_{jc}| \big) \cdot x_{ic} \cdot T_{ic} \Big), \qquad \forall\, i \in P \end{aligned} \tag{3}$$

where K_c is the number of sequences, C_i is the completion time of piece i, and T_{ic} specifies the time of the intercellular movement for piece i in cell c. Also, y_{cb} is a binary variable for the decision-making. If cell c is assigned to sequence b, $y_{cb} = 1$; otherwise, $y_{cb} = 0$. In addition, $C_{\max}$ (maximum makespan) is calculated according to Equation (4):

$$C_{\max} = \max \left(C_{kjcb} + \sum_{i=1}^{P} \sum_{c=1}^{C} \Big(\sum_{j=1}^{M} a_{ij} \cdot |a_{ij} - a_{jc}| \Big) \cdot x_{ic} \cdot T_{ic} \right) \tag{4}$$

In the following, the problem of manufacturing scheduling is explained through an example. Consider a company that manufactures four types of car pieces. The company has two manufacturing cells and a total of five machines. The pieces show the jobs that need to be manufactured by different machines. The paths for the pieces show the machines that must be applied to those parts. Details of this example are shown in Table 1.

In this example, Piece 1 requires Machines 1, 2, and 3 to complete the manufacturing process. Piece 2 requires Machines 1 and 2, Piece 3 requires Machines 3, 4,

and 5, and finally Piece 4 requires Machines 3 and 4. Manufacturing a piece by any machine has a time process.

Table 1
Machines required for processing pieces

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Piece 1	1	1	1	0	0
Piece 2	1	1	0	0	0
Piece 3	0	0	1	1	1
Piece 4	0	0	1	1	0

Table 2 shows the time that is required to process each component on the machines. Here, the time required to complete Piece 1 on machines 1, 2 and 3 is 4, 5 and 12, respectively. For the other components, the processing times are specific and fixed. According to the placement of the machines in two cells, the dependence of the machines that are available on each of the cells is shown in Table 3.

Table 2
Time needed to process pieces on machines

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Piece 1	4	5	12	0	0
Piece 2	6	4	0	0	0
Piece 3	0	0	7	6	14
Piece 4	0	0	5	3	0

Table 3
Machine dependence on cells

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Cell 1	1	1	0	0	0
Cell 2	0	0	1	1	1

Regarding the dimensions of the problem in Figures 1 and 2 (two pieces, two cells, and five machines), the problem is solved with two different modes of the cell sequence.

The solutions for the two different modes show the pieces with total construction times of 27 and 48. Considering the objective function of "minimizing the total time of manufacturing the pieces", the best sequence obtained is that the pieces are first applied to Cell 2 and then to Cell 1. With this sequence, the total construction time of the pieces is 27 (which is less than the second method). In the sequence of cells, each piece must be applied to the cell in which it is located. Therefore, the sequence of jobs (pieces) is also taken into account in addition to the sequence of cells in this problem.

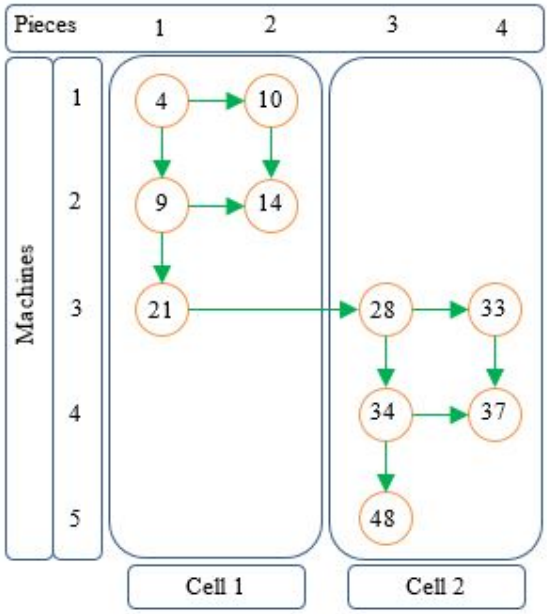

Figure 1. Sequence in form of Cell 1 and then Cell 2 – State 1

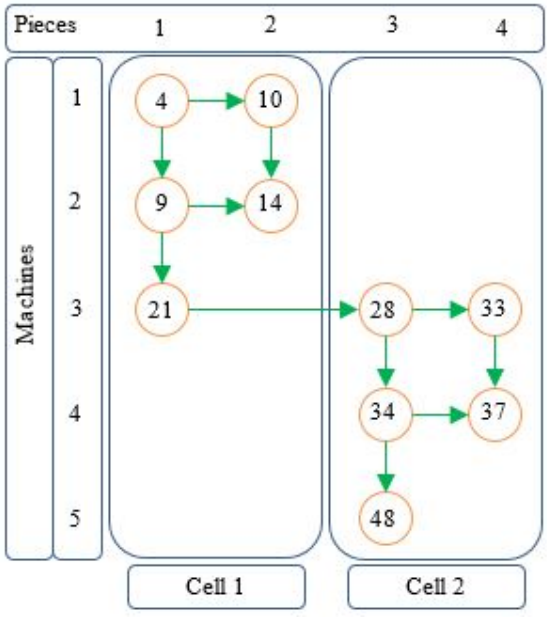

Figure 2. Sequence in form of Cell 1 and then Cell 2 – State 2

4. Proposed method

In this study, a combination of genetic and greedy algorithms has been used to provide a meta-functional method for solving the problem of job-shop manufacturing scheduling. Combining these two algorithms to achieve quality initial responses accelerates the finding of the range of responses; thus, the speed of the problem-solving and the appropriate quality of the final response are guaranteed. The structure of chromosome representation plays a key role in optimizing the genetic algorithm [9]. This paper uses a simple linear structure for the problem of job-shop manufacturing scheduling. In the genetic algorithm optimization process, the initial population is generated by a greedy algorithm [25]. The solutions are then improved by using several elitism operators. In the proposed greedy algorithm, the cells and jobs in each cell are prioritized. Based on the priority of each cell, one cell is selected as the next cell; this is done by the tournament selection strategy for cell manufacturing scheduling. Then, the jobs in the cells are prioritized; and similar to a roulette wheel, one job is selected as the next job.

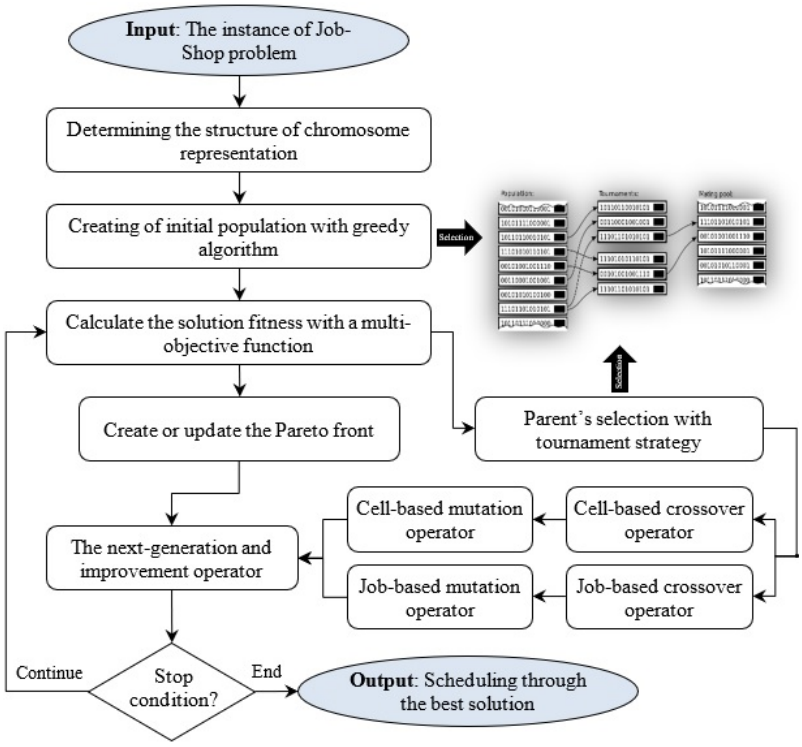

Figure 3. Flowchart of proposed method

In this paper, the fitness of the solutions is calculated based on four different objective functions. These objective functions include "reducing the total time of the

piece manufacturing", "reducing the intercellular movement", "reducing the cost of the pieces", and "reducing the tardiness cost". Figure 3 of the flowchart shows the proposed method.

The research assumptions and details of the proposed meta-heuristic algorithm are described below.

4.1. Research assumptions

In the job-shop scheduling problem, there are several situations and limitations in which the assumptions for presenting the proposed method are expressed.

- In this problem, there are M machines and P pieces where the machines are settled in the cells.
- The processing time is a fixed amount and is independent of the sequence of jobs.
- Each piece has a specific process to complete.
- There is a fixed time to complete each piece on each machine.
- Scheduling is the processing of each piece according to the cell in which the machine is located.
- There is no failure time for machines, and each machine performs definite operations that are related to the piece.
- The number of pieces, machines, and cells in the problem is specific and fixed.
- Each machine can only be applied to one piece at a time; therefore, other pieces must wait if they need to use the machine.
- The cells are located at varying distances from each other, and displacing pieces between cells bears costs.
- The goal of solving this problem is to find the best sequence of jobs in each cell as well as the sequence of cells, where the intercellular movement, earliness cost of the pieces, and tardiness cost of the pieces are minimized.
- All machines are available at the beginning of the scheduling; their time is zero at the start.
- Some pieces in the process of completing need machines that may be in several cells. In this case, the completion time of the piece is equal to the sum of the completion time in the previous cell sequence and the processing time in the current sequence.
- The completion time of the piece in the j-th machine for a sequence is equal to the sum of the completion time on the $(j-1)$-th machine and the processing time of the piece on the j-th machine.
- The completion time of a piece for a sequence on the j-th machine in a cell is equal to the sum of the maximum completion time on the $(j-1)$-th machine and the processing time of the piece on the j-th machine. Meanwhile, if the completion time of the last sequence in the previous cell is greater than the maximum completion time on the $(j-1)$-th machine, the completion time is equal to the sum of the completion time of the last sequence, and the processing time is in the current cell sequence on the j-th machine.

4.2. Creating initial population based on greedy algorithm

In this study, a vector with a total number of jobs (P) is used for chromosome representation in the genetic algorithm. In this structure, the sequence of jobs shows the timing of its performance in the CMS. Due to the simultaneous optimization of the sequence of cells and functions, the timing of the cells is automatically determined according to the location of the jobs. This simple structure increases the design capabilities of various operators and reduces the complexity of the proposed method. Also, it is easily possible to switch jobs between cells by considering only the number of jobs in the proposed structure. The structure of the proposed chromosome is shown in Figure 4.

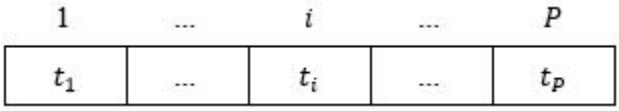

Figure 4. Structure of proposed chromosome representation

,

where t_i represents the i-th piece, and the length of each solution is equal to the number of pieces. In addition, if the t_i-th piece belongs to the machine in cell c_i, this piece is scheduled in the current sequence of cell c_i.

In general, the quality of the initial population plays an important role in solving the problem by the GA. To reduce the size of the problem space and the number of generations as well as increase the quality of the population, a greedy algorithm is used to generate an initial population. The greedy algorithm proposed at each stage makes the best choice based on the current state of the system and the prioritization of the cells and jobs. Table 4 shows the stages of initial population manufacturing.

In this algorithm, the minimum number of intercellular displacements is calculated. For example, suppose a piece needs machines $\{m_1, m_2, m_3\}$ in order to be completed so that the m_1 and m_2 machines are in cell c_1 and machine m_2 is in cell c_2 to the components that are assigned to the machines within that cell. Under this circumstance, the minimum number of displacements for this piece is two. After applying the m_1 machine in cell c_1, it must move to machine m_2 in cell c_2; so, a case of displacement can occur. In the next step, the piece must move from cell c_2 to cell c_1 with the completion of the m_2 machine process in order to use machine m_3. This step also imposes an intercellular displacement for the piece. To consider the intercellular displacement for each specific cell, displacement is regarded for the cell from which the transfer took place. In this case, the number of displacements for cells c_1 and c_2 is one.

Table 4
Generate initial population with greedy algorithm

Generation of initial population with greedy algorithm	
1	Start
2	The number of displacement is calculated for each cell.
3	The probability of selecting each cell should be calculated based on a smaller number of displacements.
4	The next cell in the chromosome should be selected based on the tournament strategy.
5	The pieces associated with the selected cell are identified, where they should not be present on the chromosome.
6	The probability of selecting each piece should be calculated based on the criterion of being less "the result of multiplying the displacement of the piece by the processing time of the piece".
7	The next piece in the chromosome should be selected based on the tournament strategy.
8	If all of the current cell components are not selected, go to Step 5.
9	If all of the cells are not selected, go to Step 4.
10	End

In general, the reason for using a tournament strategy is to increase the population diversity and not to create duplicate chromosomes. This strategy selects k random individuals from the population and then selects the two best individuals among these k individuals to be parents. In addition, the pieces that are associated with a cell are determined by considering the first machine that is needed to complete that piece; for example, when machines $\{m_1, m_2, m_3\}$ are needed to complete a piece and these machines are in cells $\{c_1, c_2, c_1\}$, respectively. Then, based on the machine $m_1$present in cell c_1, the desired piece belongs to cell c_1. The underlying reason is that the first machine determines the start of the manufacturing process of the piece.

This algorithm will largely maintain the distribution of jobs between machines and cells based on their priorities; it will also reduce the number of displacements between cells. In cases where there are the same priorities for selecting jobs, the jobs are randomly selected.

4.3. Calculating fit and creating Pareto front

In this study, four objective functions of "reducing the total time of the piece manufacturing", "reducing the intercellular movement", "reducing the cost of the pieces", and "reducing the cost of delaying the pieces" are used to calculate the suitability of the solutions. The aim of the proposed algorithm is to minimize these functions in order to improve the quality of the manufacturing system schedule. The target functions are shown in Equations (5) through (8).

$$F_1 = C_{\max} \tag{5}$$

$$F_2 = C_{Move} \tag{6}$$

$$F_3 = \sum_{i=1}^{P} E_i . \max\{0, d_i - C_i\} \tag{7}$$

$$F_4 = \sum_{i=1}^{P} L_i . \max\{0, C_i - d_i\} \tag{8}$$

The value that is attributed to F_1 is the time for making the last piece ($C_{\max}$). The F_2 function is used to indicate the number of displacements that are made between cells in the construction of the pieces. If a piece needs several machines to be made and these machines are located in different cells, the piece must be moved between the cells; this displacement (which is proportional to the distance between two cells) imposes costs (increased timing) in the system. Therefore, the sequence of cells and the components must be created in such a way as to move the least number of C_{Move} in the solution. In F_3, E_i is the earliness cost for the i-th piece, d_i is the delivery time of the i-th piece, and C_i is the completion time of the i-th piece. The earliness cost of the pieces is equal to the total earliness cost for all of the P pieces. The positive difference between the delivery time of a piece and its time of completion shows the amount of time that the piece is prepared ahead of time and creates maintenance cost (E). Finally, F_4 calculates the total tardiness cost for all of the P pieces. In this case, L_i is the tardiness cost for the i-th piece. The positive difference between the completion time of a piece and its delivery time shows the amount of time that the piece is prepared later than the due time and leads to tardiness cost (L).

Various studies have shown that meta-functional methods are much more effective at solving multi-objective optimization problems than traditional tools. There is no single answer in solving multi-objective problems due to contradictions between goals; in other words, there is no single answer in which all goals are optimal. Finally, a set of dominant answers will be presented as optimal answers. These answers are called the Pareto solution front [8]. The main advantage of this method is its capability of providing a set of non-dominated answers for the decision-maker. In this study, the size of the Pareto front is fixed, and this front is updated at each repetition.

In general, it is absolutely uncommon to find a single solution that presents the minimum (or maximum) values for all of the objectives at the same time. Vector $F(x)$ is said to dominate another vector $F(y)$ (x and $y \in C$, which is denoted by $F(x) \prec F(y)$) if and only if $f_i(x) \leq f_i(y)$ for all $i \in \{1, 2, \cdots, o\}$ (where o is the number of criteria) and $f_i(x) < f_i(y)$ for at least one $j \in \{1, 2, \cdots, o\}$. A point $x \in C$ is said to be globally Pareto optimal or a globally efficient point if and only if there does not exist $y \in C$ that satisfies $F(y) \prec F(x)$. $F(x)$ is then called globally non--dominated. In this case, we do not have a single optimal solution but a variety of Pareto-optimal solutions.

4.4. Genetic operators

Selection operator: both parents that are used for remanufacturing are selected by elitism from the Pareto list. The selection policy in both methods is based on the roulette wheel.

Crossover operator: in this study, two crossover operators have been used – one to improve the sequence of the cells, and the other to improve the timing of the jobs. This operator is applied with a probability of CR. In the cell-based crossover operator, the cells are first identified in the parents; then, a cell is randomly selected from each parent and copied in the child's chromosome (with the restriction of non-repetition). If one or more jobs are already available among the existing jobs in the selected cell that is intended for copying in the child's chromosome, the jobs will be removed from the cell; then, the copying will be done. Figure 5 shows an example of a cell-based crossover operator. In a job-based crossover operator, a work is first selected from each parent at random and then copied to the child's chromosome (with the restriction of non-repetition). The sequence of job selection indicates the sequence of performing jobs in the timing system. In order to increase the efficiency of this operator, the selected jobs are chosen from one cell at each stage. In fact, when a c-cell is selected from the first parent, a c-cell is selected from the second parent (if one exists). This strategy prevents successive displacements in the child's chromosome. Figure 6 shows an example of a job-based crossover operator.

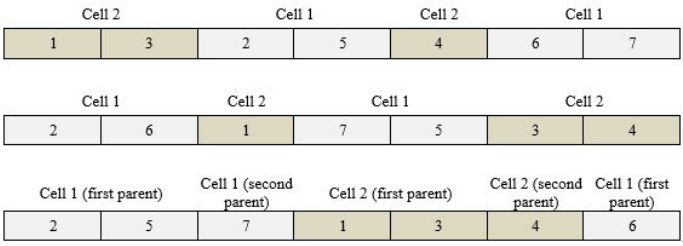

Figure 5. Example of cell-based crossover operator

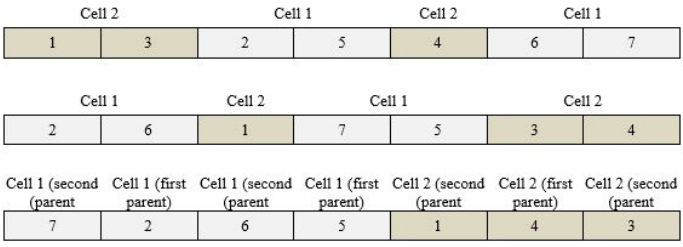

Figure 6. Example of job-based crossover operator

Mutation Operator: in this study, two mutation operators were used – one to improve the cell sequence, and the other to improve the work timing. In cell-based mutation, all non-adjacent cells on the chromosome are first identified and then randomly selected for each cell; then, the displacement occurs. In the job-based mutation operator, the non-adjacent cells in the chromosome are first identified; then, another operation is performed for each job in the same cell by randomly. Through the probability of MR, the displacement occurs.

4.5. Adaptive control of crossover and mutation operators' rate

The performance probability of the crossover and mutating operators depends on CR and MR, where their optimal settings affect the quality of the solutions and accelerates the convergence of the genetic algorithm. Numerous studies have shown that using adaptive control rather than considering the constant probability for CR and MR makes tremendous improvements in the ability to search for genetic algorithms [16, 26]. In this paper, the adaptive probability of CR and MR is determined according to Equations (9) and (10):

$$CR = \begin{cases} k_1 \frac{(iter)}{(Iter_{Max})} \times CR & \acute{f} \geq \bar{f} \\ CR, & \acute{f} < \bar{f} \end{cases} \tag{9}$$

$$MR = \begin{cases} k_2 \frac{(iter)}{(Iter_{Max})} \times MR & \acute{f} \geq \bar{f} \\ MR, & \acute{f} < \bar{f} \end{cases} \tag{10}$$

where k_1 and k_2 are two constant values that are smaller than the one that determines the rate of the reductions of CR and MR. $\bar{f}$ and $\acute{f}$ are the average fitness solutions of the old and current generation, respectively, which are determined based on $s(i)$ in the Pareto front. The $s(i)$ for the i-th solution equals the number of solutions that have been defeated by the this solution. Finally, $iter$ and $Iter_{Max}$ are the iteration current and maximum number of iterations, respectively.

4.6. Next-generation and improvement operator

The interesting point of the genetic algorithm is the possibility of generating highly suitable chromosomes in intermediate generations (yielding solid results at the fitness function). These chromosomes might be destroyed as a result of the crossover and mutation operators and not be generated anymore. The elitism technique identifies such cases and uses them in subsequent generations.

In this paper, the current-generation solutions, old-generation solutions, and Pareto front solutions are used to generate the population of the next-generation. In each generation, a number of Pareto front solutions are transferred directly to the next-generation. Therefore, if the size of the Pareto front list is N_{Pareto} and

the population size is N_{Pop}, the composition of the next-generation solutions will be determined according to Equation (11):

$$Pop_{new} = [Pareto(1 : \alpha.N_{Pareto}), Pop(1 : N_{Pop} - \alpha.N_{Pareto})] \tag{11}$$

where α represents the percentage of the Pareto front that is selected as the next-generation population. *Pop* is the sum of the two current- and old-generation populations; this is used to complete the next-generation population solutions. The solutions in both the *Pareto* and *Pop* parts are selected based on $s(i)$. In addition, this paper uses an improvement operator to improve the efficiency of the solutions in the next-generation population. The improvement operator is a local search for finding the best neighbors of the solutions. The operator tries to place pieces together by moving the contents of non-adjacent cells so that their machines are located in the same cells. This process will eventually reduce the number of displacements among the cells in a solution. Table 5 shows the steps of the improvement operator for an input solution.

Table 5
Improvement operator steps

Improvement operator steps	
1	Start
2	Set the number of non-adjacent cells in the solution to the variable
3	For $i = 1, 2, \cdots, n_c$ do
4	Select a non-adjacent cell from cell i at random.
5	Replace all the pieces from the selected cell to all the pieces from cell i.
6	Changes will be apply, if new solution versus a solution of Pareto front prevails.
7	End for
8	End

5. Results and experiments

In this section, the performance of the proposed algorithm is evaluated to improve flexible job-shop manufacturing planning systems using the P-FJSP dataset from the UCI machine-learning repository. The simulation is done with MATLAB software (Version 2016a), and the NSGA-II and NRGA algorithms have been used for the comparison work. In all of the experiments, an average of 15 runs for each algorithm are reported to ensure the results. The efficiency and effectiveness of the hybrid and meta-heuristic algorithms greatly depend on the appropriate adjustment of the parameters. Here, the parameters are determined by using the Taguchi method to achieve the best solution [24]. The results that were obtained through various parameters of the three levels are estimated based on standard table of orthogonal arrays L_{27} [24]. The aim of the Taguchi method is to maximize the S/N ratio (signal-to-noise), which is calculated by Equation (12) in the paper.

$$S/N_{ij} = -10\log_{10}\left(\frac{1}{m}\sum_{i=1}^{m} F_{obj}(i,j)^2\right), \qquad \forall\, j \in level, \tag{12}$$

where $F_{obj}(i,j)$ is the objective function value that uses parameter i on the j-th level, and m is the number of times that the j-th level of parameter i is repeated over the runs of all of the trials. According to Taguchi, the values of the parameters that were used in the simulation of the proposed algorithm are as follows: $N_{Pop} = 50$, $CR = 0.85$, $MR = 0.15$, $N_{Pareto} = 10$, $Iter_{Max} = 500$, $k_1 = 0.05$, $k_2 = 0.01$, and $\alpha = 0.4$.

5.1. Evaluation criteria

To evaluate the performance quality of the proposed algorithm with the four-objective function, the diversity, spacing, quality, and run-time criteria are used.

The diversity criterion shows the amount of change that exists among the data of a distribution. This criterion specifies the Euclidean distance between the first and last solutions in a Pareto front. Higher diversity values imply the higher quality of the results [1]. This criterion is defined as Equation (13):

$$D = \sqrt{\sum_{m=1}^{2}\left(\max_i f_m^i - \min_i f_m^i\right)^2}, \quad \forall\, i = 1, 2, \cdots, N \tag{13}$$

where N is the number of non-dominated solutions in a Pareto front, and f_m^i represent the i-th value of the non-dominated solution (which are makespan form $m = 1$ and stability form $m = 2$).

The spacing criterion is used to show the compatibility of the distance between the solutions in the Pareto front [10]. Lower values of the distance criterion indicate that the distance stability between the solutions is higher. This criterion is defined as Equation (14):

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d_i - \bar{d})^2} \tag{14}$$

where $\bar{d}$ is the average of the N solutions in the Pareto front, and d_i represents the Euclidean distance between the two non-dominated solutions in the Pareto front (which is defined according to Equation (15)):

$$d_i = \min_{k \in N, k \neq i} \sum_{m=1}^{2} |f_m^i - f_m^k| \tag{15}$$

where f_m^i and f_m^k represent the i-th and k-th values, respectively, of the non-dominated solution.

The quality criterion tests the distribution of the Pareto front solutions that are obtained at the solution boundary. This criterion is defined as Equation (16):

$$Q = \frac{\sum_{i=1}^{N-1} |\bar{x} - x_i|}{(N-1) \cdot \bar{x}} \tag{16}$$

where x_i represents the Euclidean distance between the two non-dominated solutions that are adjacent to the Pareto front, $\bar{x}$ is the average x_i, and N indicates the number of members in the Pareto front list.

The last criterion used for comparison is run-time. Since all of the parameters in the algorithms are the same, this criterion is a suitable comparison criterion for evaluating the manufacturing scheduling system. This criterion is reported in the comparisons in seconds.

5.2. The P-FJSP dataset

In this paper, the P-FJSP dataset (BRdata) was used to evaluate the proposed algorithm and performance the comparisons. This dataset was produced by BrandMiart (1993) and includes ten instances [3,7]. The parameters of each of the datasets in this dataset are generated randomly by using a uniform distribution between two ranges. The number of jobs is defined as 10 to 20, the number of machines – 4 to 15, the number of operations for each job – 5 to 15, and the number of operations for all jobs – 55 to 240. In addition, it is specified which machines are required for each job. Table 6 shows the details of the P-FJSP dataset.

Table 6
Details of P-FJSP dataset

Instances	No. of jobs	No. of machines	No. of cells	No. of operations	No. of processing time	Size instance	Total operation time
MK01	10	6	3	5–7	1–7	10×6	56
MK02	10	6	6	5–7	1–7	10×6	58
MK03	15	8	5	10–10	1–20	15×8	150
MK04	15	8	3	3–10	1–10	15×8	90
MK05	15	4	2	5–10	5–10	15×4	106
MK06	10	15	5	15–15	1–10	10×15	150
MK07	20	5	5	5–5	1–20	20×5	100
MK08	20	10	2	5–10	5–10	20×10	225
MK09	20	10	5	10–15	5–10	20×10	250
MK10	20	15	5	10–15	5–20	20×15	240

In the first line of each P-FJSP instance, there are at least two numbers. The first is the number of jobs, and the second is the number of machines (the third is not necessary – it is the average number of machines per operation). Then, every

line represents one job: the first number is the number of operations of that job, and the second (let us say $k >= 1$) is the number of machines that can process the first operation. Then, according to k, there are k pairs of numbers (machine, processing time) that specify which are the machines and which are the processing times; the data for the second operation follows, and so on.

5.3. Results and discussion

For comparison, the NRGA and NSGA-II algorithms are used according to the diversity, spacing, quality, and run-time (seconds) criteria. A comparison of the diversity criterion in the different algorithms on ten experimental instances is shown in Figure 7.

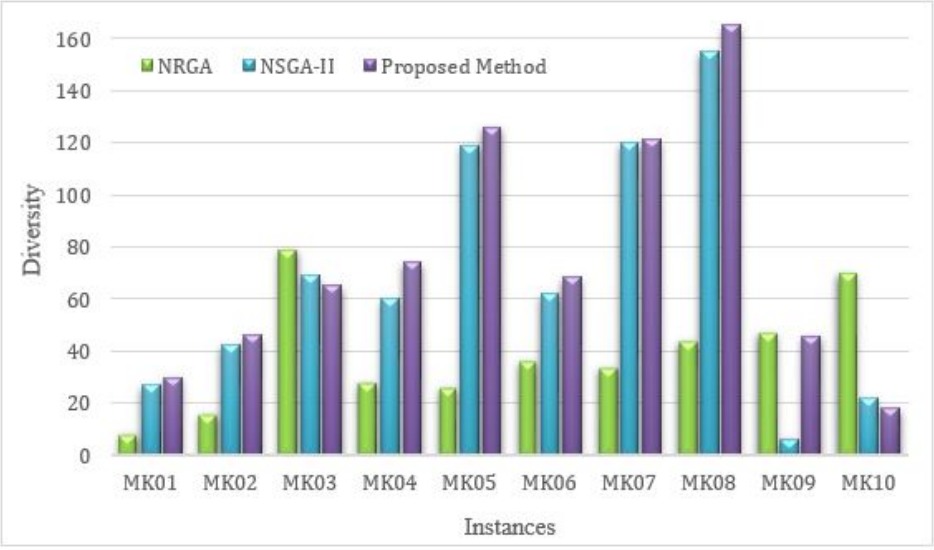

Figure 7. Comparison of different methods in diversity criterion

The results show that the NSGA-II algorithm has better quality than NRGA. On the other hand, the proposed algorithm performs better than both methods (with a diversity criterion of 74.85). Figure 8 shows the comparison results in the spacing criterion. Since lower values of distance mean higher quality, the NRGA method performed better than NSGA-II in the MK09 and MK10 models. However, the spacing criterion in the NRGA and NSGA-II algorithms averaged 19.19 and 12.71, respectively; this indicates that higher-quality solutions are produced by the NSGA-II algorithm. Despite the superiority of the NSGA-II algorithm over the NSGA, the proposed algorithm performed better than both methods (with an average spacing criterion of 12.4).

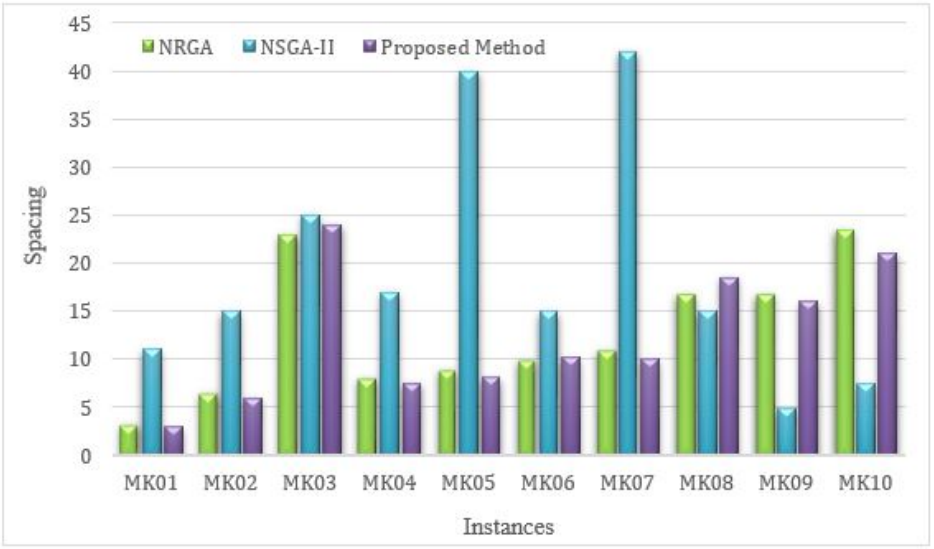

Figure 8. Comparison of different methods in spacing criterion

Based on the quality criterion, the NSGA-II method is superior to the NRGA method in all instances except for MK08 of NRGA. In this criterion, the superiority of the proposed method existed in all instances when compared to the NSGA-II method; as a result, it showed a better quality of performance. Figure 9 shows the results of comparing the quality criterion of each instance on average. Regarding the run-time criterion, there is no significant statistical difference between the results of the three algorithms being compared. In Figure 10, the results of the run-times of these algorithms are reported in the ten tested instances (with slight differences in the results).

In general, the proposed method in the diversity criterion has a better performance than the NRGA and NSGA-II methods in all instances except for MK03, MK09, and MK10. Furthermore, the proposed method increased the average diversity criterion to 19.36 and 96.8 units, respectively; so, the diversity criterion witnessed an increase in this method when compared to the two other methods. The average reduction in the spacing criterion in the ten tested instances decreased by 7.05 and 0.57, respectively, when compared to NRGA and NSGA-II.

The results of the proposed method in the quality criterion also have an appropriate performance; on average, this criterion increased by 0.01 and 0.11, respectively, as compared to the two comparison methods. Since all three comparison methods were applied to the genetic algorithm, the results fluctuated within almost the same range as the run-time criterion. The results of this criterion for the proposed method and the NRGA and NSGA-II methods averaged 2965, 3015, and 2931 seconds, respectively; this is considered to be a relative superiority for the proposed method.

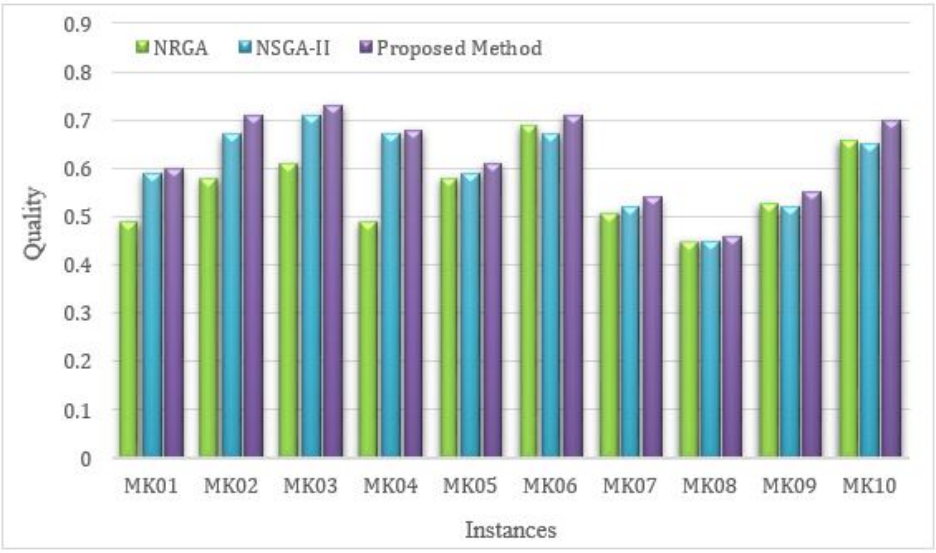

Figure 9. Comparison of different methods in quality criterion

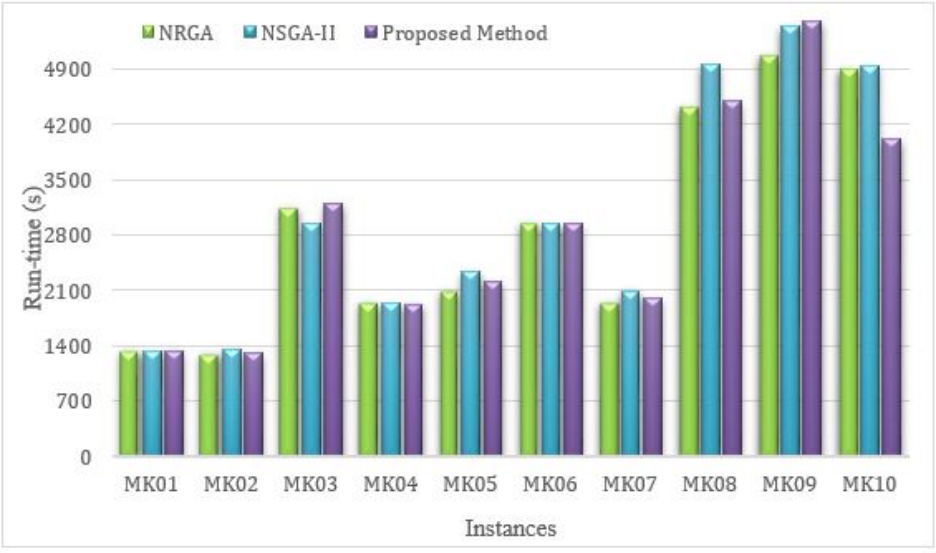

Figure 10. Comparison of different methods in run-time criterion

Finally, the numerical results that compare the proposed method with the NRGA and NSGA-II algorithms are shown in Table 7. Each row represents an instance of a problem, and the columns show the diversity, spacing, quality, and run-time criteria, respectively. In addition, the last row for each algorithm represents the number of instances with better results than other algorithms.

Table 7
Comparison of proposed method with NRGA and NSGA-II algorithms in different criteria

Instances	Proposed method				NRGA				NSGA-II			
	Diversity	Spacing	Quality	Time [s]	Diversity	Spacing	Quality	Time [s]	Diversity	Spacing	Quality	Time [s]
MK01	28.33	2.60	0.77	1267	7.75	11.18	0.49	1417	27.29	2.27	0.72	3194
MK02	44.89	5.95	0.61	1014	15.7	14.32	0.58	1395	42.52	6.13	0.64	2017
MK03	66.87	23.56	0.63	2678	77.32	25.57	0.58	2958	69.64	23.25	0.62	2182
MK04	73.26	7.18	0.59	1609	26.89	17.56	0.50	1963	60.76	8.21	0.63	3042
MK05	125.82	8.09	0.55	2267	25.55	40.13	0.54	2370	113.32	9.44	0.57	2027
MK06	66.91	10.13	0.65	3018	35.76	15.21	0.45	2970	60.95	10.20	0.66	4445
MK07	119.01	9.50	0.54	2011	32.12	42.13	0.53	2178	105.95	10.62	0.54	5194
MK08	161.91	18.10	0.49	5828	43.38	16.66	0.61	4589	153.86	15.21	0.45	4877
MK09	43.44	15.14	0.55	5528	47.13	3.92	0.33	5412	5.26	16.84	0.54	965
MK10	18.14	21.18	0.68	4371	75.08	6.57	0.38	4906	19.41	23.10	0.63	1361
# Wins	7	7	9	4	3	2	1	2	0	1	0	4

In general, the proposed method in the diversity criterion performed better than the NRGA and NSGA-II algorithms in all instances except for MK03, MK09, and MK10. The superiority of the proposed method in the spacing criterion was 7.05% and 0.57% against NRGA and NSGA-II, respectively. The results of the proposed method in the quality criterion also had a satisfactory performance, where the superiority of the proposed method was 0.01% and 0.11% against NRGA and NSGA-II, respectively. Due to the use of all three comparative methods of the genetic algorithm, the results fluctuated approximately the same as with the run-time criterion. However, the average results for the proposed method and the NRGA and NSGA-II algorithms were 2965, 3015, and 2931 seconds, respectively.

In order to more accurately evaluate the proposed method, we provide further comparisons in Table 8. This comparison includes the genetic algorithm (GA) [21], the neighborhood-based genetic algorithm (NGA) [4], the hybrid evolutionary algorithm (HEA) [31], the genetic algorithm with a tabu search in a holonic multi-agent model (GATS+HM) [20], and the improved genetic algorithm with adaptive variable neighborhood search (IGA-AVNS) [10]. The data reported in this table was collected from the corresponding literature. This comparison is based on the best heuristic solution that was discovered after ten operations; i.e., the minimum value of the completion time obtained.

Table 8
Comparison of proposed method with other algorithms

Instances	GA	NGA	HEA	GATS+HM	IGA+AVNS	Proposed method
MK01	40	37	40	41	40	40
MK02	26	26	27	28	26	26
MK03	204	204	204	204	204	204
MK04	60	60	60	65	60	60
MK05	173	173	173	175	173	173
MK06	63	67	59	67	60	58
MK07	149	148	144	144	144	145
MK08	523	523	523	523	523	523
MK09	311	307	307	312	307	305
MK10	212	212	209	225	208	208
# Wins	0	1	0	0	0	2

The results show the superiority of NGA for MK01 over the other algorithms; however, all of the algorithms offer the same solution for MK03, MK05, and MK08. The proposed method performs better than GA for almost all instances. The proposed method provides a better solution than the GATS+HM algorithm for MK02, MK04, MK06, MK09, and MK10. Also, compared to the HEA algorithm, a better solution was found for MK02 and MK10. For MK07, the IGA-AVNS algorithm finds a better solution, but for MK06 and MK09, the proposed method performs better.

In summary, the proposed method is an efficient method for solving the job-shop manufacturing scheduling problem.

6. Conclusion and future work

In the present study, the multi-objective scheduling of a flexible job-shop manufacturing system using a meta-heuristic algorithm was investigated. This algorithm is a combination of genetic and greedy algorithms to find the optimal sequence of jobs as well as the sequence of cells. In order to fulfill the suitability of the solutions, four goals were used: reducing the total time of manufacturing the pieces, reducing the intercellular movement, reducing the cost of the pieces quickly, and reducing the tardiness cost of the pieces. Since random solutions bring about a reduction in the quality of the initial population and an increase in the duration of the convergence, a greedy algorithm was used in this study to create an initial population to reduce the size of the problem space and the number of generations. This algorithm makes the most efficient choice at each stage based on the current state of the system and the prioritization of the cells and jobs. Therefore, the minimum number of intercellular displacements is calculated according to the pieces that are assigned to the machines within that cell. For future studies, the times that are related to machine preparation as well as the problem scrutiny in a dynamic environment can be scrutinized.

References

[1] Ahmadi E., Zandieh M., Farrokh M., Emami S.M.: A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms, *Computers & Operations Research*, vol. 73, pp. 56–66, 2016.

[2] Beasley D., Bull D.R., Martin R.R.: An Overview of Genetic Algorithms: Part 2, Research Topics, *University Computing*, vol. 15(4), pp. 170–181, 1993.

[3] Brandimarte P.: Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, vol. 41(3), pp. 157–183, 1993.

[4] Driss I., Mouss K.N., Laggoun A.: A new genetic algorithm for flexible job-shop scheduling problems, *Journal of Mechanical Science and Technology*, vol. 29(3), pp. 1273–1281, 2015.

[5] Engin O., Döyen A.: A new approach to solve hybrid flow shop scheduling problems by artificial immune system, *Future Generation Computer Systems*, vol. 20(6), pp. 1083–1095, 2004.

[6] Erenay B., Suer G.A., Huang J., Maddisetty S.: Comparison of layered cellular manufacturing system design approaches, *Computers & Industrial Engineering*, vol. 85, pp. 346–358, 2015.

[7] Flexible Job Shop Problem. http://people.idsia.ch/~monaldo/fjsp.html (last visited on 7 May. 2021).

[8] Gao K.Z., Suganthan P.N., Pan Q.K., Chua T.J., Cai T.X., Chong C S.: Pareto--based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling, *Information Sciences*, vol. 289, pp. 76–90, 2014.

[9] Ghalehgolabi M., Rezaeipanah A.: Intrusion Detection System Using Genetic Algorithm and Data Mining Techniques Based on the Reduction, *International Journal of Computer Applications Technology and Research*, vol. 6(11), pp. 461–466, 2017.

[10] Gu X., Huang M., Liang X.: An Improved Genetic Algorithm with Adaptive Variable Neighborhood Search for FJSP, *Algorithms*, vol. 12(11), pp. 243–259, 2019.

[11] Jadaan Al O., Rajamani L., Rao Raghavendra C.: Non-dominated ranked genetic algorithm for solving constrained multi-objective optimization problems: NRGA, *Journal of Theoretical & Applied Information Technology*, vol. 5(5), pp. 60–67, 2009.

[12] Jalilvand-Nejad A., Fattahi P.: A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem, *Journal of Intelligent Manufacturing*, vol. 26(6), pp. 1085–1098, 2015.

[13] Karthikeyan S., Saravanan M., Ganesh K.: GT Machine Cell Formation Problem in Scheduling for Cellular Manufacturing System Using Meta-Heuristic Method, *Procedia Engineering*, vol. 38, pp. 2537–2547, 2012.

[14] Kazemi M., Gol S.S., Tavakkoli-Moghaddam R., Kia R. Khorrami J.: A mathematical model for assessing the effects of a lot splitting feature on a dynamic cellular manufacturing system, *Production Engineering. Research and Development*, vol. 11(4–5), pp. 557–573, 2017.

[15] Knosala R., Wal T.: A production scheduling problem using genetic algorithm, *Journal of Materials Processing Technology*, vol. 109(1–2), pp. 90–95, 2001.

[16] Land W.H., Schaffer J.D.: Background on Genetic Algorithms. In: *The Art and Science of Machine Intelligence*, pp. 1–44, Springer, Cham, 2020.

[17] Li Y., Li X., Gupta J.N.: Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search, *Expert Systems with Applications*, vol. 42(3), pp. 1409–1417, 2015.

[18] Maghfiroh M.F., Darmawan A., Vincent F.Y.: Genetic Algorithm for Job Shop Scheduling Problem: A Case Study, *International Journal of Innovation, Management and Technology*, vol. 4(1), pp. 137–140, 2013.

[19] Mahmoodian V., Jabbarzadeh A., Rezazadeh H., Barzinpour F.: A novel intelligent particle swarm optimization algorithm for solving cell formation problem, *Neural Computing and Applications*, vol. 31(2), pp. 801–815, 2019.

[20] Nouri H.E., Driss O.B., Ghédira K.: Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model, *Journal of Industrial Engineering International*, vol. 14(1), pp. 1–14, 2018.

[21] Pezzella F., Morganti G., Ciaschetti G.: A genetic algorithm for the Flexible Job-shop Scheduling Problem, *Computers and Operations Research*, vol. 35(10), pp. 3202–3212, 2008.

[22] Rauch Ł., Górecki G., Pietrzyk M.: Hybrid computer system for the identification of metallic material models on the basis of laboratory experiments, *Computer Science*, vol. 17, pp. 123–143, 2016.

[23] Reddy V., Narendran T.T.: Heuristics and sequence-dependent set-up jobs in flow line cells, *International Journal of Manufacturing Research*, vol. 41(1), pp. 193–206, 2003.

[24] Rezaeipanah A., Ahmadi G., Hajiani M., Darzi M.R.: An Improved Hybrid Cuckoo Search Algorithm for Vehicle Routing Problem with Time Windows, *Journal of Quality Engineering and Production Optimization*, vol. 4(2), pp. 189–208, 2019.

[25] Rezaeipanah A., Matoori S.S., Ahmadi G.: A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search, *Applied Intelligence*, vol. 51(1), pp. 467–492, 2021.

[26] Shojaedini E., Majd M., Safabakhsh R.: Novel Adaptive Genetic Algorithm Sample Consensus, *Applied Soft Computing*, vol. 77, pp. 635–642, 2019.

[27] Solimanpur M., Vrat P., Shankar R.: A heuristic to minimize makespan of cell scheduling problem, *International Journal of Manufacturing Economics*, vol. 88(3), pp. 231–241, 2004.

[28] Srinivas M., Patnaik L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24(4), pp. 656–667, 1994.

[29] Tenschert A., Kubert R.: SLA-based job submission and scheduling with the Globus Toolkit 4, *Computer Science*, vol. 13, pp. 183–204, 2012.

[30] Varela M.L., Trojanowska J., Carmo-Silva S., Costa N.M., Machado J.: Comparative Simulation Study of Production Scheduling in the Hybrid and the Parallel Flow, *Management and Manufacturing Engineering Review*, vol. 8(2), pp. 69–80, 2017.

[31] Wang C., Tian N., Ji Z.C., Wang Y.: A hybrid evolutionary algorithm for flexible job shop scheduling problems. In: *2016 35th Chinese Control Conference (CCC)*, pp. 2690–2696, IEEE, 2016.

[32] Yang Y., Chen Y., Long C.: Flexible robotic manufacturing cell scheduling problem with multiple robots, *International Journal of Manufacturing Research*, vol. 54(22), pp. 6768–6781, 2016.

[33] Zeng C., Tang J., Yan C.: Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles, *Journal of Intelligent Manufacturing*, vol. 26(5), pp. 845–859, 2015.

Affiliations

Amin Rezaeipanah
University of Rahjuyan Danesh Borazjan, Department of Computer Engineering, Bushehr, Iran, amin.rezaeipanah@gmail.com

Fariba Sarhangnia
Islamic Azad University, Department of Computer Engineering and Information Technology, Bushehr Branch, Bushehr, Iran, sarhangnia.fariba@gmail.com

Mohammad Javad Abdollahi
University of Lian Bushehr, Department of Computer Engineering, Bushehr, Iran, it1393@yahoo.com

Received: 20.02.2021
Revised: 23.05.2021
Accepted: 07.06.2021