Operational Research in Engineering Sciences: Theory and Applications
Vol. 3, Issue 3, 2020, pp. 13-28
ISSN: 2620-1607
eISSN: 2620-1747
DOI: https://doi.org/10.31181/oresta20303013s

AN APPLICATION OF METAHEURISTIC OPTIMIZATION ALGORITHMS FOR SOLVING THE FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

Aleksandar Stanković*, Goran Petrović, Žarko Ćojbašić, Danijel Marković

Faculty of Mechanical Engineering, University of Niš, Serbia

Received: 11 June 2020
Accepted: 10 August 2020
First online: 24 September 2020

Original scientific paper

Abstract. *The Flexible Job Shop Planning (FJSP) problem is another planning and scheduling problem. It is a continuation of the classic problem of scheduling jobs, where each operation can be performed on different machines, while the processing time depends on the machine being used. FJSP is a difficult NP problem that consists of two sub-problems, scheduling problems and scheduling operations. The paper presents a model for solving FJSP based on meta-heuristic algorithms: Genetic algorithm (GA), Tabu search (TS) and Ant colony optimization (ACO). The efficiency of the approach in solving the aforementioned problem is reflected in the flexible search of space and the choice of dominant solutions. The results of the computation are graphically represented on the Gantt chart.*

Keywords: *Scheduling, Flexible job-shop, Genetic algorithm, Tabu Search, Ant Colony Optimization, Local search.*

1. Introduction

The planning of production and production processes has a very important role in the successful functioning of production. Planning and scheduling problems occur in almost every field of economics, engineering, up to industrial production. One of the most important production issues is the planning and scheduling of operations. A key reason for scheduling and planning operations is to increase production productivity. Scheduling and planning operations can be very easy, but it can also be one of the most difficult scheduling problems, depending on the type of problems and planning conditions. A problem where there is more than one machine available for each operation, where there is flexibility in selecting a machine from a set of alternative machines is called the Flexible Job Shop Problem (FJSP). According to the

* Corresponding author.
aleksandar.stankovic@masfak.ni.ac.rs (A. Stanković), goran.petrovic@masfak.ni.ac.rs (G. Petrović), zcojba@ni.ac.rs (Ž. Ćojbašić), danijel.markovic@masfak.ni.ac.rs (D. Marković)

JSP action routine, each job is processed on a machine with a defined processing time, and each machine can only process one operation. In practice, a machine may have the flexible ability to perform more than one type of operation, leading to the modification of the JSP in the FJSP. As Pinedo (2008) stated his book, the definition of FJSP can be expressed as a generalization of the workplace and the parallel environment of machines. Instead of *m* machines in a row, there are *c* centers for working with each work center in parallel with the same number of identical machines. Each job has its own route to follow throughout the shop; job *j* requires processing in each work center on only one machine and each machine can run. If a business on its way through the store can visit the work center more than once, then the *b*-field contains the *rcrc* entry for recirculation.

The aim of this paper is to test and compare tree meta-heuristic optimization methods in order to minimize the amount of time spent planning and scheduling operations on the available set of machines. The results obtained using different approaches should help managers to identify an appropriate method for this class of problem.

There are different approaches to solving FJSP available in the literature, and that will be reviewed in the next section. In the earlier years of research into planning and scheduling problems, exact methods were used in the allocation of resources. Today, methods such as constraint programming and simulation methods are being used more and more in the planning world. The objective of this paper is reflected in the application of several meta-heuristic methods, namely Tabu Search (TS) algorithm, Ant Colony Optimization (ACO) and Genetic Algorithm (GA) and their comparison in the speed of the convergence and accuracy of the solutions. The basic idea is to assess which of the applied algorithms is most applicable for solving FJSP.

2. Literature review

Resource planning and scheduling, as well as the methods used to solve scheduling problems, are gaining ground in many areas of logistics, planning, search, and routing, where this methodology is very significant and applicable. Speaking of scheduling (Brucker & Schile, 1990), they are one of the first scientists to develop a graphical algorithm for planning and scheduling. The algorithms most commonly used today to solve scheduling problems within the FJSP are meta-heuristic algorithms: Genetic Algorithm (Fraser, 1957; Bremermann, 1958; Holland, 1975), Ant Colony Optimization (Dorigo et al., 2006), Simulated Annealing (Kirkpatrick et al., 1983), Tabu Search (Glover & Laguna, 1997), Particle Swarm Optimization (Kennedy & Eberhart, 1995) and others. In the continuation of the work in Figure 1, methods are presented on the basis of which it is possible to solve the problems of planning and scheduling resources.

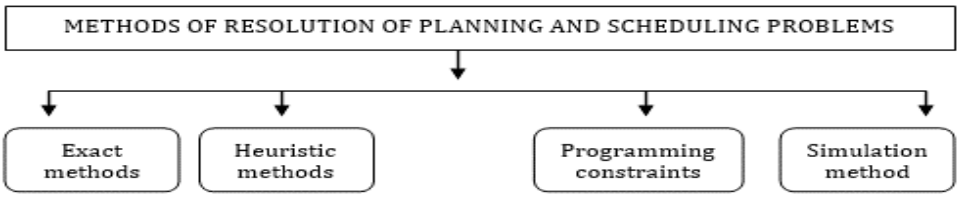

Figure 1. Methods for solving scheduling problems

Exact methods: The basic feature of exact methods is the accuracy in defining mathematical model as well as finding optimal solutions depending on the size of the data tested. One of the major drawbacks of exacting methods is solving robust models. This group of methods includes some of the basic techniques used to solve scheduling and planning problems: nonlinear, linear, dynamic, integer, and disjointed programming techniques. There are many exact algorithms in the research literature that are used to solve such research problems, such as Branch and Bound methods (Lomnicki, 1965), and they can be defined through various techniques for determining the lower and upper bounds.

Klein & Scholl (1996), as well as Blazewicz, et al., (1996) present, in their works, the Branch and Baund method in the example of resource planning, and the main aim of their work is to assign tasks to a certain number of machines in order to achieve maximum productivity. Liu, et al. (1996) and Thomalla (2001) present, in their papers, the problem of scheduling by the Lagrangian relaxation method, where they prove that the scheduling and planning problem can be successfully solved by this method. Robson (1986) worked to refine an algorithm he had already developed to improve temporal complexity. Ostergard (2002) proposed a Branch and Baund algorithm that defines each node with a different color to distinguish the nodes from each other, which at that point is a new tagging methodology. Vandaele (2000), Hasan & Arefin (2017), and Aslan et al. (2017) show, in their papers, the problem of planning and scheduling and the success in solving these problems by an integrated method of planning.

Heuristic methods: Alan Turing is probably the first to use heuristic algorithms when he broke the German Enigma code during World War II. The next significant step in the evolution of evolutionary algorithms was (Holland, 1975) and his associates at the University of Michigan in the 1960s and 1970s. Such search methods do not guarantee finding the optimal solution, but effectively finding a good enough solution. Heuristics are divided into: heuristics that give only one solution within the search and heuristics that give results during the search through a series of iterative solutions. In the works, (Sentlelro, 1993), (Lagodimos & Leopoulos, 2000), (Spyropoulos, 2000), (Xing & Zhang, 2000), we can see a heuristic approach to solving planning and scheduling problems, and also based on the obtained results, it can be seen that this approach gives optimal results. Kung & Chern (2009) show another way to solve planning problems. In this paper, we can see a scheduling heuristic approach that focuses on solving factory planning and scheduling operations for different job (product) structures. For this planning problem, a heuristic algorithm is proposed, and is referred to in the paper as the factory planning heuristic algorithm, abbreviated as HFPA. Xing & Zhang (2000) use a method of heuristic approach to solve the problem of planning and scheduling on the problem of M parallel machines with minimal total cost. Sobeyko & Mönch (2016) present a heuristic approach to solving scheduling problems in large-scale flexible operations. Based on the results obtained in this paper, we can conclude that the proposed heuristics arrive at satisfactory solutions quickly.

Programming constraints: This problem-solving approach belongs to the group of NP-hard problems, and the basic feature is the programming of constraints on problem solving in industrial planning and scheduling systems. There are a number

of different software applications used in this troubleshooting category that can be used to program scheduling constraints. This approach originated in the field of artificial intelligence. The programming languages most commonly used today to solve artificial intelligence planning and scheduling problems are: MatLab, Python, C ++, C #, Java, and more. When it comes to programming time, constraints planning, and scheduling problems are meta-heuristic methods largely presented.

One example of solving scheduling problems can be seen in (Stanković, et al., 2019). It should be noted that two approaches are presented in the literature: the deterministic approach and the stochastic approach (Pinedo, 2008). The time of completion of the scheduling of operations (products) in the scientific literature is indicated by $C_{\max}$ which represents the total criterion value of the function. Examples of solving problems of planning and scheduling can be seen in (Jamili, 2018); (Fan, et al., 2019). Solving FJSPs based on meta-heuristic algorithms with programming constraints in the form of time constraints, periods of unavailability can be seen in the papers (Zhang, et al., 2011); (Stanković, et al., 2019).

Tamssaouet, et al. (2018) compares several meta-heuristic algorithms with their associates with periods of machine unavailability in the form of preventive and corrective machine maintenance. Liaw, (2000) presents the application of a hybrid algorithm with a basic genetic algorithm. Further research includes a local tab-based search enhancement process. The results obtained show optimality compared to other search algorithms.

Simulation methods: Simulation modeling has a great ability to present complex systems in a multitude of details, which is its main advantage over other methods. Simulation-based planning is used for many operations and system controls, and as a final output, a detailed work plan is obtained. Simulation-based planning models need to be more detailed than other typical simulation models. Many models in practice with this type of problem can only be solved by the simulation-based optimization method, an approach in which the simulation model is integrated with meta-heuristic search methods such as GA and TS (Laguna, et al., 2003).

The Kanban method is used to increase the productivity of product flow through a single production system and eliminate potential errors at the end of a cycle. Kanban is a system that controls the flow of material (resources) through a number of multiple optimization processes. Kanban system was developed by Toyota engineers - Taiichi Ohno (Industrial Engineer and Businessman) to optimize their manufacturing process. The implementation and success of solving the problems of planning and scheduling operations in small and medium-sized enterprises can be seen in many professional papers. Schaefers et al., (2000) is one example of solving product planning and flow problems as well as cost optimization. Problems were identified, analyzed and optimized based on the Kanban method. Japanese industry management technique has been applied in many western countries.

Graver & Price (1987) present the Kanban method in their work and use it to solve JSP planning and scheduling problems in the form of simulation, and the final results of the simulation show a significant improvement of the system over previous planning practice. Kumar & Panneerselvam (2007) present the Kanban system and 100 state-of-the-art research papers as well as suggestions for further research.

The Work Load Control (WLC) method involves three models: planning, control, and scheduling. The basic task of this method is to solve the problem of production load. In many cases, when planning and selecting a job, the rules of job priority are used, depending on the delivery time of a certain type of product, which is one of the most important factors during the planning process in small and medium-sized enterprises. Such methods are of great use in the form of simulations and in solving planning and scheduling problems, which can be seen in papers (Thürer et al., 2012); (Thürer et al., 2017).

3. Methodology

Accurate and heuristic methods are used to solve planning and scheduling problems. The application of exact methods is limited to simple problems, while more complex meta-heuristic methods are used for complex real systems. The term meta-heuristics was first proposed by Fred Glover in 1986, while the same author defined meta-heuristics many years later as a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms (Sörensen & Glover, 2013). Meta-heuristics are designed to solve complex optimization problems when other optimization methods fail to effectively solve the optimization problem.

These methods are nowadays recognized as one of the most important practical approaches to solving many complex problems, and this is especially important for solving many real combinatorial optimization problems, hence the application for the FJSP problem. In general, meta-heuristics can be said to be higher level heuristics. Below, we present three meta-heuristic methods that have been applied in solving the problems of allocation and scheduling of FJSP operations.

3.1. Tabu Search

The Tabu Search (TS) algorithm was first mentioned by a famous scientist Glover, (1986). The TS algorithm is a meta-heuristic search method that uses local search methods. Search implementation uses structural models that describe average places, that is, possible solutions, or use sets that the user defines as the initial parameters of the problem under consideration. This means that if a potential solution was previously visited at some point in the search or if the set search rules were exceeded, then it will be marked in the tab list. So, the TS algorithm does not take the same solution multiple times as possible solutions during search.

Tabu searches during previous research have proven to be the optimal search method in a wide range of classic and practical planning problems, and even in the field of neural networks, as can be seen in papers (Nowicki & Smutnicki, 2005); (Zhang et al. 2007). To avoid problems during the search, the size of the taboo list during the search needs to be modified (Talbi, 2009). Tabu list size is crucial during this type of search. For a taboo list that is too small, a search will tend to cycle through the same possible solutions multiple times, whereas if the taboo list is too

large, the lack of available moves can lead to possible errors during the search. The TS algorithm is represented by a pseudocode in Table 1 (Glover, 1989).

Table 1. Pseudocode of Tabu Search

```
Pseudocode of Tabu Search
sBest ← s0
bestCandidate ← s0
tabuList ← []
tabuList.push(s0)
While (not stoppingCondition())
    sNeighborhood ← getNeighbors(bestCandidate)
   for (sCandidate in sNeighborhood)
 if ((not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)))
     bestCandidate ← sCandidate
   end for
   end if
   if (fitness(bestCandidate) > fitness(sBest))
    sBest ← bestCandidate
   end if
    tabuList.push(bestCandidate)
   if (tabuList.size > maxTabuSize)
                tabuList.removeFirst()
   end if
end While
```

3.2. Ant colony optimization

The Ant Colony Optimization (ACO) method was first proposed by Dorigo (1992). Ant colony optimization is a population-based meta-heuristic that can be used to find approximate optimal solutions for different test cases. The algorithm is inspired by the behavior of ants in nature. The basic characteristic of the collective behavior of ants is that all members of the colony exchange information about their environment indirectly or directly, i.e. the phenomenon of collective intelligence. It has been discovered in nature that each ant leaves a trail behind, releasing a certain amount of a chemical called a pheromone. The more ants go in one path, the more pheromones, and that is, for each subsequent ant, positive information about the correctness of that path. In this way, the ants indirectly communicate with each other via pheromones. All ants start with a value of 0, which means that no operations are scheduled before the search begins. All nodes have an initial pheromone 1. The pheromone will decrease after each round of search. Local search depends on the number of pheromones and the search time, and the total time is calculated based on the extra time required to activate the next O_{ij} operation on the available M_n machine. A random value is generated for comparison with r_0. If the rand value is less than r_0, the local search will select the planning route with the maximum amount of pheromone. Only the best ants can deposit the pheromone on the path from point A to point B. The total amount of pheromone deposited is calculated based on the

expression $\Delta\tau = 1 / (best\ cost)$. The pseudocode of the ACO algorithm is presented in Table 2 (Yang, 2010).

Table 2. Pseudocode of Ant Colony Optimization

Pseudocode of Ant Colony Optimization

Objective function f (x), x = (x1,…xn)T
[or f (xij) for routing problem where (i, j) € {1, 2, ….. n}]
Define pheromone evaporation rate $^{\gamma}$
while (criterion)
for loop over all n dimensions (or nodes)
Generate new solutions
Evaluate the new solutions
Mark better locations/routes with pheromone $^{\delta\varphi}$ij
Update pheromone: $^{\varphi}$ij – (1-$^{\gamma}$) $^{\varphi}$ij + $^{\delta\varphi}$ij
end for
Daemon actions such as finding the current best
end while
Output the best results and pheromone distributions

3.3. Genetic algorithm

Genetic Algorithm (GA) is an optimization technique used to solve nonlinear or non-differential optimization. The GA was developed by Holland in the 1970s of the last century (Holland, 1975). The genetic algorithm is characterized by several stages in solving a defined problem, in this case of planning and scheduling.

The algorithm mimics the natural selection process, while changes in the genetic structure are possible by mutation of genetic material, the essence of which is to expand the search area and overcome local extremes. Crossing in solving GA is a process of combining several units to get a new unit selected, and this type is compared to a natural process like parents and their offspring. New individuals inherit their parents' genes. When it comes to solving planning and scheduling problems, the most common examples are based on a genetic algorithm. One of the most common solutions is based on a coded job scheduling matrix used for scheduling problems on more than one machine, an example of such a matrix can be seen in Figure 2.

$$\begin{aligned} &M1: O_{11}; O_{13}; O_{12} \longrightarrow M1: J_1; J_3; J_2 \\ &M2: O_{22}; O_{23}; O_{21} \longrightarrow M2: J_2; J_3; J_1 \\ &M3: O_{31}; O_{32}; O_{33} \longrightarrow M3: J_1; J_2; J_3 \end{aligned}$$

Figure 2. Job scheduling matrix

The mutation involves a random change in the genes of the individuals. Mutation achieves uncontrolled alteration of genetic material. By changing the genetic structure, completely new solutions are achieved. The basic goal is to get an individual that cannot be obtained in other stages.

This random value initiates a search across the entire allowed domain. The mutation rate should be small from about 0.001% to 0.01% in order to avoid a random, stochastic and uncontrolled procedure. The pseudocode of GA is presented in Table 3 (Yang, 2010).

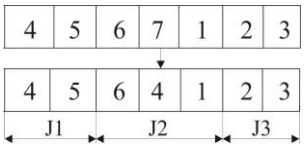

Figure 3. Mutation in the case of scheduling machine

Table 3. Pseudocode of GA

Pseudocode of GA
Objective function $f(_x)$, $x= (x_1,....,x_n)^T$
Encode the solution into chromosomes (binary strings)
Define fitness F (eg, $F \infty f(x)$ for maximization)
Generate the initial population
Initial probabilities of crossover (p_c) and mutation (p_m)
While (t < Max number of generations)
Generate new solution by crossover and mutation
If p_c > rand, Crossover;
end if
If p_m > rand, Mutate;
end if
Accept the new solutions if their fitness increase
Select the current best for new generation
end while
Decode the results and visualization

4. Case study

This section presents a method for solving FJSPs based on three proposed meta-heuristic algorithms. The first part of the case studies presents a model for testing algorithms on a classic data set. The efficiency of the algorithms is reflected in the speed of the convergence solution through a series of iterations. The input parameters consist of 25 jobs and 10 machines with a defined processing time of each operation on an individual machine. The mathematical model of solution optimization with objective function and time constraint is represented by the following notation:

- J – number of jobs,
- M – number of identical machines,
- $p_{i,j,k}$ – the processing time of each operation.

Constraints:

$$J > M > 1,\ J, M \in Z^+ \tag{1}$$

$$\forall\ p \in T,\ \ p \in Z^+ \& 1 \le p \le M \tag{2}$$

Cost function:

$$f(x)=max_i f(i), \quad i=1,2,\ldots, M \tag{3}$$

The input parameters as well as the results of the optimization problem for all three meta-heuristic algorithms are presented in Table 4.

Table 4. Input parameters and results: TS, ACO and GA

TS parameters		
MaxIter	J	M
1000	25	10
Time for TS algorithm for which an optimal solution is found [s]		
Local best found for 1000 iterations	187	
ACO algorithm parameters		
MaxIter	J	M
1000	25	10
Time for ACO algorithm for which an optimal solution is found [s]		
Local best found for 1000 iterations	182	
GA parameters		
MaxIter	J	M
1000	25	10
Time for GA for which an optimal solution is found [s]		
Local best found for 1000 iterations	176	

Based on the results presented in Table 4, it can be concluded that TS, ACO and GA give satisfactory results, but that GA gives the most favorable results. On the basis of the obtained results, GA was used in the FJSP case study of planning and scheduling operations in an industrial environment.

The case study covers the planning and scheduling of production cycles related to the FJSP solution. The basic structure of the problem being solved relates to a set of jobs that need to be done on the machines, and each job is allocated to a list of activities that are processed in the order. The essence of the problem and the defined activity is to keep the total completion time as low as possible in accordance with the planned operations with defined time of each operation individually. The set of operations performed to make one job complete and therefore a finished product. The mathematical model of FJSP can be represented as follows (Özgüven et al., 2010).

It is necessary to schedule n products $J = \{J_1, J_2, ..., J_n\}$, with each job J_j ($_j$ = 1,2, ... n) having a predetermined order of operations nj (O_1, j, O_2, j ,. ... They, j), should be realized in the order given in m machines $M = \{M_1, M_2,..., M_m\}$. For a completely flexible problem, each machine can perform only one operation at a time, and the processing time of each operation depends on the machines available and is represented by p, i, j, k (processing time of the operation O_i, j on the machine M_k). The goal of the problem observed is to assign each operation to a corresponding machine and then determine the arrangement of all the machines assigned to the operations

to reduce the target function, in this case minimizing the manufacturing process based on GA. An example of the problem examined is presented in Table 5.

Table 5. Input matrix of partial FJSP

Jobs	Operations	M_1	M_2	M_3	M_4	M_5	M_6
	O_{11}	7	-	1	5	-	-
	O_{21}	3	-	-	5	-	8
J_1	O_{31}	-	6	-	7	-	10
	O_{41}	7	-	5	7	-	-
	O_{51}	-	5	-	-	6	3
	O_{61}	7	-	-	-	6	3
	O_{12}	-	8	-	8	-	9
	O_{22}	5	-	5	-	-	4
J_2	O_{32}	10	-	11	-	10	-
	O_{42}	8	-	7	-	-	10
	O_{52}	10	-	-	10	-	12
	O_{62}	-	7	15	4	-	-
	O_{13}	8	-	5	-	7	-
	O_{23}	-	4	-	4	6	-
J_3	O_{33}	-	-	-	-	-	8
	O_{43}	9	-	8	-	7	-
	O_{53}	-	3	-	5	-	3
	O_{63}	-	5	6	-	7	-
	O_{14}	-	5	-	6	-	9
	O_{24}	5	-	4	-	8	-
J_4	O_{34}	9	-	5	-	-	10
	O_{44}	5	11	-	3	-	-
	O_{54}	15	-	9	-	8	-
	O_{64}	-	10	-	11	-	9
	O_{15}	9	-	9	-	9	-
	O_{25}	-	3	6	-	8	-
J_5	O_{35}	-	6	7	-	5	-
	O_{45}	-	6	-	5	-	4
	O_{55}	3	-	4	-	4	-
	O_{65}	-	8	-	6	-	9
	O_{16}	-	3	-	5	-	4
	O_{26}	8	-	-	3	6	-
J_6	O_{36}	7	-	8	-	-	9
	O_{46}	10	-	8	9	-	-
	O_{56}	-	9	-	10	4	-
	O_{66}	7	-	6	-	8	-

It should be noted that not all machines need to be able to perform all the operations as can be seen in the attached Table 5. This troubleshooting approach is called partial troubleshooting or partial flexibility where some operations can only be performed on specific machines. Such examples are much more common in real-world cases during optimization of production processes. Table 5 of the problem described can show 6 jobs and 6 machines, job 1 has 6 operations, the first operation can only be processed by one machine and that machine is M_3 with a processing time of 1. We can see the graphical results of the first investigated case of partial research flexibility in Figure 4 in the Gantt chart.

One of the most common means of presenting a result in the case of production planning is Gantt charts and as they are called in the research literature, Gantt charts. Gantt chart is a diagram in a coordinate system in which the horizontal axis is time, and the vertical axis shows the planning tasks on which to determine: the beginning, total duration and end of the cycle, which is the main problem in determining the planning and scheduling in this case of machine and work. Another examined case of planning and scheduling is called total FJSP, with each operation being deployable on any of the available *m* machines, as all machines are capable of performing each operation at a specified time during a specified processing time of each operation. An example of the problem examined is presented in Table 6.

Table 6. Input matrix of total FJSP

Jobs	Operations	M_1	M_2	M_3	M_4	M_5	M_6
J_1	O_{11}	7	11	9	7	8	9
	O_{21}	3	6	12	10	5	7
	O_{31}	5	6	6	11	7	10
	O_{41}	5	5	7	7	8	7
	O_{51}	9	5	10	9	6	3
	O_{61}	10	3	11	8	6	7
J_2	O_{12}	7	8	7	5	7	7
	O_{22}	4	7	5	10	9	8
	O_{32}	5	7	9	10	10	9
	O_{42}	3	5	11	10	9	10
	O_{52}	10	7	9	9	6	6
	O_{62}	3	9	5	4	11	10
J_3	O_{13}	5	6	5	7	8	9
	O_{23}	2	5	9	4	8	9
	O_{33}	8	5	9	10	10	8
	O_{43}	9	5	9	10	11	8
	O_{53}	7		7	10	7	9
	O_{63}	7	9	9	10	7	11
J_4	O_{14}	7	5	7	8	9	10
	O_{24}	5	5	7	9	9	9
	O_{34}	5	9	5	10	6	10
	O_{44}	6	7	8	3	9	3
	O_{54}	2	5	9	9	8	10
	O_{64}	7	9	9	10	11	9
J_5	O_{15}	9	5	9	8	10	11
	O_{25}	6	5	5	8	10	6
	O_{35}	9	5	9	10	5	9
	O_{45}	5	9	10	11	12	4
	O_{55}	3	5	5	6	9	11
	O_{65}	9	8	9	8	10	7
J_6	O_{16}	2	3	9	8	10	7
	O_{26}	2	5	9	3	9	7
	O_{36}	2	5	9	9	10	9
	O_{46}	10	9	10	9	9	10
	O_{56}	9	5	10	6	4	9
	O_{66}	10	5	9	4	9	8

Table 6 shows the input parameters of the test problem where we have 6 jobs (products) where each of these 6 jobs has 6 operations, each of which defined operations with machine processing time in minutes can be performed on each machine. As we can see in the previous part of the paper, we call this case complete flexibility. Graphical results of the survey can be seen in Figure 5. As for the implementation of GA, the experimental results were tested in python and the numerical values were derived on standard MS Windows based PC platform.

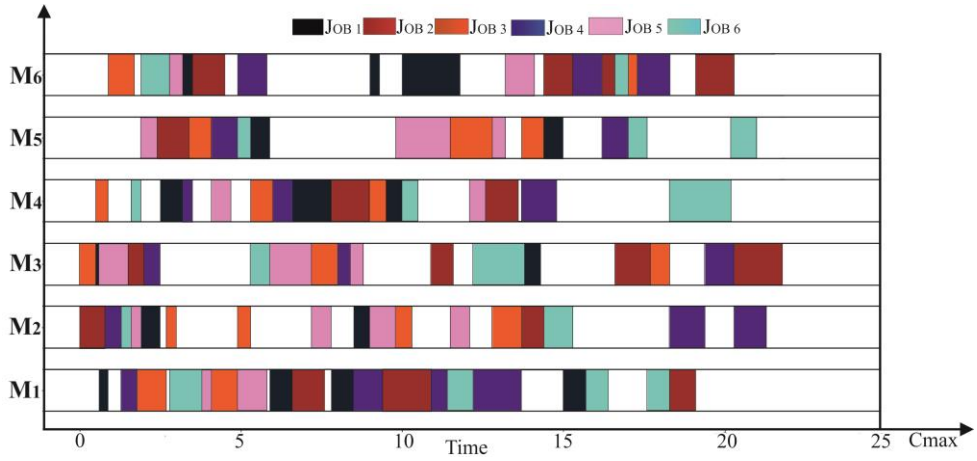

Figure 4. Graphical representation of the results for the first case in Table 5

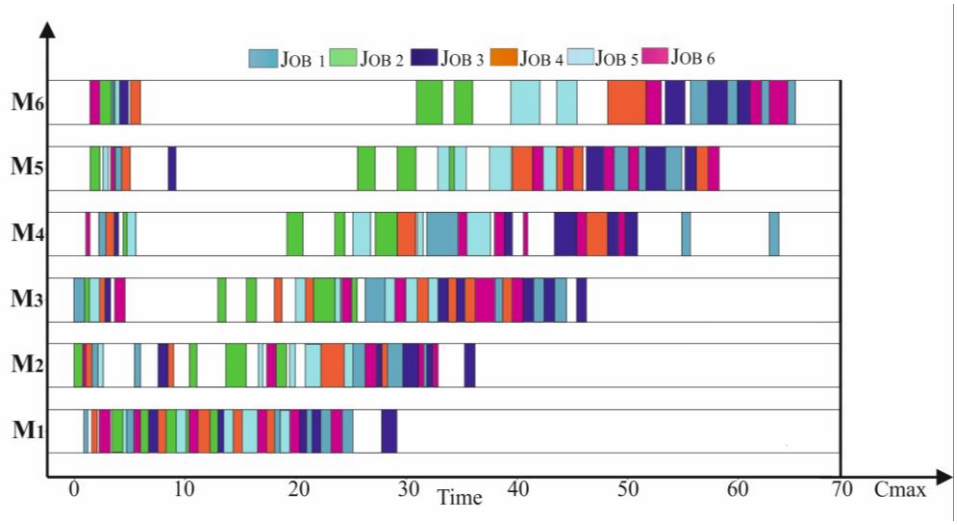

Figure 5. Graphical representation of the results for the second case in Table 6

5. Results and discussion

After introductory discussions and clarification of the FJSP, as well as a review of the research literature on the topic of planning, a specific set-up of the problems and

results of the planning was made. It is important to note that the tested test data is randomly given. As can be seen in the previous part of the paper, two cases of planning and scheduling were tested, which was at the same time the aim of this problem. Table 5 shows that not every machine can perform every operation during the planning and scheduling of operations. Partial flexibility is one of the most common causes that can be encountered in manufacturing, it is often the case that not every machine can perform every operation to get the job (product) done. The results of the input data in Table 5 are graphically presented in the chart in Figure 4 for the first case of partial flexibility. In Figure 4, we can see the layout of operations on individual machines depending on the initial constraints. J defines each job in a different color to differentiate jobs (products). On the horizontal axis, time is presented with the criterion function value $C_{\max}$ = 218 units in minutes. Regarding testing, it was noted that the search was performed by Genetic algorithm with MaxIter = 500. GA based search in a python programming language, that is, total planning and scheduling time is 41.63 seconds for the first case tested. On the vertical axis, we can see all 6 machines that are linearly aligned. With regard to the second case examined, it is presented in Table 6 and is clearly different from the first case examined. In Table 6, based on the input data presented, it is possible to see complete flexibility where each machine can perform each operation with defined time.

Also, in this case, we have J = 6 jobs (products) and each of the jobs has O_{ij} = 6 operations that can be performed on each machine depending on the schedule of operations with a defined processing time of each operation individually. On the horizontal axis, we can see the total time represented by the value of the criterion function $C_{\max}$ = 652 units in minutes. The value of the criterion function representing the total time of completion of all operations and the completion of the planning process is represented by the $C_{\max}$ mark, which could be seen in the previous part of the paper and is presented in minutes. The Gantt chart presents a detailed schedule of operations that will be performed on machines. The machines are represented linearly on the vertical axis. The total simulation time based on GA in a python programming language is 90.89 in seconds. The objective of the examined problem of planning and scheduling operations was successfully realized, as can be seen from the attached results. The results obtained confirm the success of the genetic algorithm in solving FJSP.

Regarding the proposal for further research, the FJS problem can be further expanded by observing available workers who participate in the realization of production activities. In this problem, workers and machines are limited, and the problem is called the Dual Resource Constrained Flexible Job Shop.

Acknowledgment

This research was financially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

References

Aslan, E., Şimşek, T., & KarkacieR, A. (2017). A binary integer programming model for exam scheduling problem with sSeveral departments, 13th International Conference on Knowledge, Economoy and Management, Bilgi Ekonomisi ve Yönetimi Dergisi, 12 (2), 169-175.

Blazewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, 93, 1-33.

Bremermann, H. J. (1958). The evolution of intelligence. The nervous system as a model of its environment, Technical Report No. 1, Department of Mathematics, University of Washington, Seattle, WA.

Brucker, P., Schile, R. (1990). Job-shop scheduling with multi-purpose machines. Computing, 45(4), 369–375.

Dorigo, M. (1992). Optimization, learning and natural algorithms (in italian), Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant Colony Optimization. IEEE Computational Intelligence Magazine 1(4):28-39, DOI: 10.1109/MCI.2006.329691

Fan, K., Wang, M., Zhai, Y., & Li, X. (2019). Scatter search algorithm for the multiprocessor task job-shop scheduling problem. Computers & Industrial Engineering, 127, 677–686.

Fraser, A. S., (1957). Simulation of genetic systems by automatic digital computers. II: Effects of linkage on rates under selection, Austral. J. Biol. Sci. 10:492–499.

Glover, F. W. (1989). Tabu Search – Part 1. ORSA Journal on Computing. 1 (2): 190–206. doi:10.1287/ijoc.1.3.190.

Glover, F. W., & Laguna, M. (1997). Tabu Search. Springer US.

Graver, M., Price, W. L. (1987). Using the Kanban in a job shop environment. International Journal of Production Research, 26(6):1105-1118, DOI: 10.1080/00207548808947921

Hasan, M., & Arefin, R. (2017). Application of Linear Programming in Scheduling Problem. Dhaka University Journal of Science, 65(2): 145-150.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Jamili, A., (2018). Job shop scheduling with consideration of floating breaking times under uncertainty. Engineering Applications of Artificial Intelligence, Volume 78, 28-36.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science. 220 (4598): 671–680.

Kumar, C. S., Panneerselvam, R., (2007). Literature review of JIT-KANBAN system. International Journal of Advanced Manufacturing Technology 32(3):393-408, DOI: 10.1007/s00170-005-0340-2.

Kung, L. C., & Chern, C. C. (2009). Heuristic factory planning algorithm for advanced planning and scheduling. Computers&OperationsResearch, 36(9), 2513—2530.

Lagodimos, A. G., & Leopoulos, V. (2000). Greedy heuristic algorithms for manpower shift planning, International Journal of Production Economics, 68, 95-106.

Laguna, M., Marti, R., & Marti, R. C. (2003). Scatter Search: Methodology and Implementation in C. New York: Springer.

Liaw, C. F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research, 124(1):28-42.

Liu, G., Luh, P.B., & Resch, R. (1996). Scheduling permutation flow shop using the lagrangian relaxation technique. IAFC 13th Triennial World Congress. San Francisco. USA, la-Ol 6.

Lomnicki, Z. A. (1965). "Branch-and-Bound" Algorithm for the exact solution of the three-machine scheduling problem. J Oper Res Soc 16, 89–100. https://doi.org/10.1057/jors.1965.7

Nowicki, E., & Smutnicki, C. (2005). An advanced tabu search algorithm for the job shop problem. Journal of Scheduling, 8(2), 145–159.

Ostergard, P. R. J. (2002). A fast algorithm for the maximum clique problem, Discrete Applied Mathematics, 120(1-3):197-207.

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.

Robson, J.M., (1986). Algorithms for maximum independent sets. Journal of Algorithms, 7(3):425-440.

Schaefers, J., Briquet, M., & Colin, J. (2000). A generic KANBAN based push-pull schedule in a job shop. IFAC Proceedings Volumes, 33(17):585-590

Sentlelro, J. J. S., (1933). Planning and scheduling: non-classic heuristic, 12th Triennial World Congress. Sydney. Australia.

Sobeyko, O., & Mönch, L. (2017). Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics. International Journal of Production Research, 55:2, 392-409, DOI: 10.1080/00207543.2016.1182227

Sörensen, K., & Glover, F. (2013). Metaheuristics. Encyclopedia of Operations Research and Management Science. Springer, New York.

Spyropoulos, C. D., (2000). AI planning and scheduling in the medical hospital environment. Artificial Intelligence in Medicine, 20(2):101-111.

Stanković, A., Marković, D., Petrović, G., Ćojbašić, Ž. (2019). Simulated annealing and particle swarm optimization for the vehicle routing problem and communal waste collection in urban areas, 14th International Conference on Accomplishments in Mechanical and Industrial Engineering, DEMI 2019, 497-505, isbn: 978-99938-39-84-2.

Talbi, E. G. (2009). Metaheuristics: From design to implementation, John Wiley & Sons.

Tamssaouet, K., Dauzère-Pérès, S., & Yugma, C. (2018). Metaheuristics for the job-shop scheduling problem with machine availability constraints. Computers & Industrial Engineering, 125:1-8.

Thomalla, C. (2001). Job shop scheduling with alternative process plans. International Journal of Production Economics 74(1-3):125-134 DOI: 10.1016/S0925-5273(01)00119-0

Thürer, M., Huang, G., Stevenson, M., Silva, C., & Filho, M. (2012). The performance of due date setting rules in assembly and multi-stage job shops: an assessment by simulation. International Journal of Production Research, 50(20), 5949–5965.

Thürer, M., Land, M. J., Stevenson, M., & Fredendall, L. D. (2017). On the integration of due date setting and order release control. Production Planning & Control, 28(5), 420–430.

Vandaele, N. J. (2000). An integrated, hierarcmcal framework for planning, scheduling and controlling job shop. IFAC Proceedings Volumes, 30(14):121-126.

Xing, W., & Zhang, J. (2000). Parallel machine scheduling with splitting jobs. Discrete Applied Mathematics, 103(1-3):259-269.

Yang, X. S., (2010). Engineering Optimization: An introduction with metaheuristic applications. University of Cambridge, United Kingdom.

Zhang, C. Y., Li, P. G., Guan, Z. L., & Rao, Y. Q. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. Computers & Operations Research, 34(11):3229-3242

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem, Expert Systems with Applications, 38(4):3563–3573.