Universal Journal of Applied Mathematics 10(2): 7-19, 2022 http://www.hrpub.org
DOI: 10.13189/ujam.2022.100201

An Open-Source Simulation Model for Solving Scheduling Problems

Aydin Teymourifar[1,*], **Jie Li**[2], **Dan Li**[2], **Taicheng Zheng**[2]

[1]CEGE - Centro de Estudos em Gestão e Economia, Católica Porto Business School, Porto, Portugal
[2]Centre for Process Integration, Department of Chemical Engineering, School of Engineering, The University of Manchester, Manchester, United Kingdom

Received May 12, 2022; Revised September 5, 2022; Accepted September 20, 2022

Cite This Paper in the following Citation Styles
***(a)**: [1] Aydin Teymourifar, Jie Li, Dan Li, Taicheng Zheng, "An Open-Source Simulation Model for Solving Scheduling Problems," Universal Journal of Applied Mathematics, Vol.10, No.2, pp. 7-19, 2022. DOI: 10.13189/ujam.2022.100201*
***(b)**: Aydin Teymourifar, Jie Li, Dan Li, Taicheng Zheng (2022). An Open-Source Simulation Model for Solving Scheduling Problems. Universal Journal of Applied Mathematics, 10(2), 7-19. DOI: 10.13189/ujam.2022.100201*

Abstract In this study, an open-source simulation model is presented for solving scheduling problems. The model is capable of solving different benchmarks. The methods involved in the simulation are mainly based on generating dispatching rules or using them to solve problems, but there are other heuristics as well. Dispatching rules in an evolutionary process are generated using Gene Expression Programming. For this aim, a coding method, which has not been described in the literature before, is explained. Along with the explanation of the properties of the source code, information about deterministic, dynamic models, buffer states, machine breakdown states, and the methods used to deal with them is presented. Concepts are explained with visual examples. In addition, a subject that has not been investigated in the literature before is analyzed by using the simulation model. This topic is to examine the results of solving machine assignment and operation sequencing sub-problems in flexible job shop scheduling problems with different rules. Moreover, objective functions that the source code can handle are discussed. Unlike many studies in the literature, the codes are presented to the readers as open source. Also, it is open to development and can be easily modified by users to solve other types of problems. Finally, in the study, experimental results are presented on the basis of some benchmarks available in the literature, and the limits of the study and source code are explained.

Keywords Simulation, Flexible Job Shop Scheduling, Dispatching Rules, Gene Expression Programming

1 Introduction

Scheduling problems have been of interest to researchers for a long time because they have many applications in the industry [3]. Exact algorithms designed for these problems cannot yield results in polynomial time [1]. Since the run times of these algorithms are non-deterministic polynomial, scheduling problems are called NP-hard. Because of the high complexity, the usage of heuristics to solve especially real-life and large-scale scheduling problems has become popular [4-7]. One category of robust heuristics is related to dispatching rules (DRs), which transform the solution time into the polynomial. An example is provided in other sections to illustrate this point. In the literature, several classic DRs exist, like First In First Out (FIFO) and Shortest Processing Time (SPT), which are robust rules, meaning they can be used for any environment however, they may not always achieve good results. These rules provide decisions based on only one feature but in composite rules, the decision is made over multiple features. In recent years, there have been many works on the generation of composite DRs using evolutionary algorithms. The related problems are usually dynamic and/or stochastic and therefore simulation is used to solve them.

In general, there is no open-source code for the simulation model that can be modified by others. One of the main goals of this work is to provide a source code that is open to everyone, can be developed easily, and is able to solve different types of scheduling problems. Further, using this simulation model, an important subject in the flexible job-shop scheduling problem (FJSSP) is investigated. This problem is usually solved by splitting it into machine assignment and operation sequencing sub-problems. As in many combinatorial optimization problems, it is possible to manage the sub-problems of FJSSP simultaneously or hierarchically [8]. Solving them with different rules has not been investigated enough in the literature and commonly, they are solved with the same rules or a specific one. In this study, it is shown that the approach of solving them with different rules is meaningful.

Herewith, contributions of the study can be summarized as follows:

- Different from many studies, the benefit of solving machine assignment and operation sequencing sub-problems in FJSSP with different DRs is investigated.
- An open-source simulation model is introduced to solve FJSSP, which is also able to solve JSSP.
- Information about different types of FJSSP, related benchmarks, DRs, GEP, and also some heuristics is given.
- A new and easy method for coding and decoding of GEP is described.

Also, like many studies, there is a limitation, which can be summarized like this: theoretically, the contribution to the literature is not much.

The advantages of the source code are summarized as follows:

- It is able to solve FJSSP and also job-shop scheduling problem (JSSP).
- It solves the problem by dividing machine assignment and operation sequencing into sub-problems.
- It can apply the same or different DRs to these two sub-problems.
- It contains benchmarks.
- It can solve instances containing buffer states.
- It can handle machines breakdown situations with the right-shift heuristic.
- It can solve instances containing stochastic states.
- It can do both offline and online scheduling.
- It can generate schedules via orders of operations.
- It can apply the left-shift heuristic to possible gaps.
- It can generate DRs by GEP and apply them as well as integrate other DRs from literature.
- It includes classic rules such as FIFO, SPT, etc.
- It contains different objective functions.
- It can draw Gantt charts automatically.
- It is open to development.

Also, it has a disadvantage as follows: It may not have enough speed to solve large-scale problems.

There are models developed for FJSSP in Rockwell Arena software, with some given details. But there are deficiencies in machine assignment such that DRs or other heuristics cannot be used for this sub-problem. Also, the evolutionary process and hence the combination of features for DRs generations is not done in this model [9].

An open-source code written in Python that can provide multiple trees, in the cellular system, is also available. But it does not include situations such as machine breakdowns and buffer states [2]. Most of the works in this area don't provide open-source codes to be open to development.

The other parts of the work are organized as follows: a short description of the problem and explanations are given about the simulation model. Then the experimental results are presented. Conclusion and future works form the last part of the study.

2 Description of Problem and Simulation

2.1 Definition of the Problem

The problem is an FJSSP, in which there are some jobs, each of them consists of some operations to be processed on machines. The processing times of operations are deterministic and predefined but stochastic breakdowns of machines can occur, which lengthens the process of operations. It can be said that the problem consists of two sub-problems of machine assignment and operation sequencing. A detailed description of the FJSSP can be found in [10].

2.2 Simulation Model

As mentioned before, the problem is solved using DRs. The simulation model is used to employ previously defined DRs and as well as to derive new ones using GEP. The designed simulation is an event-based one. Events refer to occurrences that happen in the shop, as follows:

- Release of a job (an operation).
- Assigning an operation to a machine.
- Starting an operation on a machine $\sim$ changing the relevant machine state to busy.
- Completion of the process of an operation.
- Completion of the process on the related machine $\sim$ changing the relevant machine state to free.

The above-mentioned events form the basis of the simulation, but in addition to them, there are other events as follows, which also need to be checked during the simulation.

- Completion of a job $\sim$ completion of the process of the last operation of the job.
- A job enters to or exits from buffers states.

Based on these events, the fundamental stages of the simulation model are as follows:

- Jobs are released.
- The release time of a job is equal to the release time of the first operation of the job.
- Release time of other operations is equal to the finish time of previous operations of the same job.
- After each operation is released, it is assigned to the waiting list of a machine according to a DR or least waiting time (LWT) heuristic [2].
- If the machine is free, the process of the operation starts, otherwise it waits on the waiting list.
- When a machine is free, one of the operations waiting on the waiting list is selected using a DR and the machine starts to process.
- The finish of the process of machines, operations (and therefore jobs) is checked.

An example is given in Fig. 1 for a better understanding of the steps of the simulation, whose stages are summarized as follows:

At the start time, as shown in Fig. 1 (a), all the machines are free. Operations 1 and 7 are released, their machine assignment is made randomly and they start the process. It should be noted that the random selection is just for start however, any other heuristic can be used instead. Fig. 1(b) shows $t = 1$. The reason to advance to this time is that some events occur in this time, which are releases of operations 4, 9, and 12. It should be noted that in this example, each color represents a job with its sequenced operations. for instance, operations 1 and 2 belong to job 1, and operation 2 must start after operation 1 is completed. In this instance, LWT is used to assign operations to machines. According to this heuristic, when an operation is released, its total possible waiting time is calculated according to all assignable machines. For this aim, for all assignable machines, if the machine is busy at that moment, the remaining time to finish, if there are jobs waiting on the waiting list, the sum of their processing times and the time of the operation on each machine are added up. Thus, the total possible waiting time of a release operation is calculated according to all machines. The operation is assigned to the machine that has the minimum value of total possible waiting times. According to the LWT heuristic, operation 4 is assigned to machine 1, while operations 9 and 12 are assigned to machine 3. But since these machines both are busy at the assignment time, that is, at $t = 1$, the operations wait on the waiting lists of machines. At $t = 2$, operation 2 is released, and LWT is assigned to machine 2, which is shown in Fig. 1(c). At $t = 2$, the process of operation 1 ends, and hence machine 1 becomes free. Also, machine 2 is already free at this time. Consequently, as seen in Fig. 1(d), the processes of operations 2 and 4 start on machines 2 and 1, respectively, and the situation becomes as in Fig. 1 (e). The next event is related to the finish of the process of operation 7 on machine 3, which takes place at $t = 3$. So the simulation proceeds there, as in Fig. 1 (f). At this time, machine 3 becomes free and since operations 9 and 12 are waiting on its waiting list, it is needed to prioritize one of them to start its process before the other. This is the meaning of operation sequencing. Assuming that the SPT rule is applied for operation sequencing, operation 12 is selected and starts on machine 3 at $t = 3$, since it has a shorter processing time. This process continues until all processes are completed.

As indicated before, the simulation model is capable of doing both offline and online scheduling. The differences between these two approaches are as follows:

- In the offline scheduling, at first, the sequence of operations is created, then the schedule and Gantt chart are created over it.
- In the online scheduling, step-by-step schedule and Gantt charts are built by advancing in time. In this way, the order of the operations is also determined.

The relationship between dynamic and stochastic scheduling are as follows:

- In dynamic scheduling, some events happen in the workshop over time, and the situation changes. For example, during the time new work enters the workshop.
- The problem is deterministic dynamic if the events occurring are known beforehand, but stochastic if only the distribution of events is known. That is, the dynamic problems include deterministic and stochastic cases.

In general, in real scheduling problems, there is no unlimited space in the workshop, especially between machines. Because of this, there may be delays in the start of some operations. Buffer states can be summarized as, but are not limited to:

- If an operation of a job has finished on the current machine and the machine assigned for the next operation of that job is busy, one of the following situations occurs:
  - If the current machine has buffer space, the job waits there.
  - If the current machine does not have buffer space, the job waits on that machine until the next machine is finished.

In the implemented simulation, this breakdown state occurs during processing time and as seen in Fig. 2 with the right-shift heuristic, repair time is added to processing time.

Similarities and differences between GA, GP, and GEP can be explained as follows:

- GP and GEP have similar structures [11]. But implementation and coding of the GEP may be easier.
- Also, as seen in Fig. 3 GA and applied GEP in this study are similar. They utilize alike crossover and mutation operators.

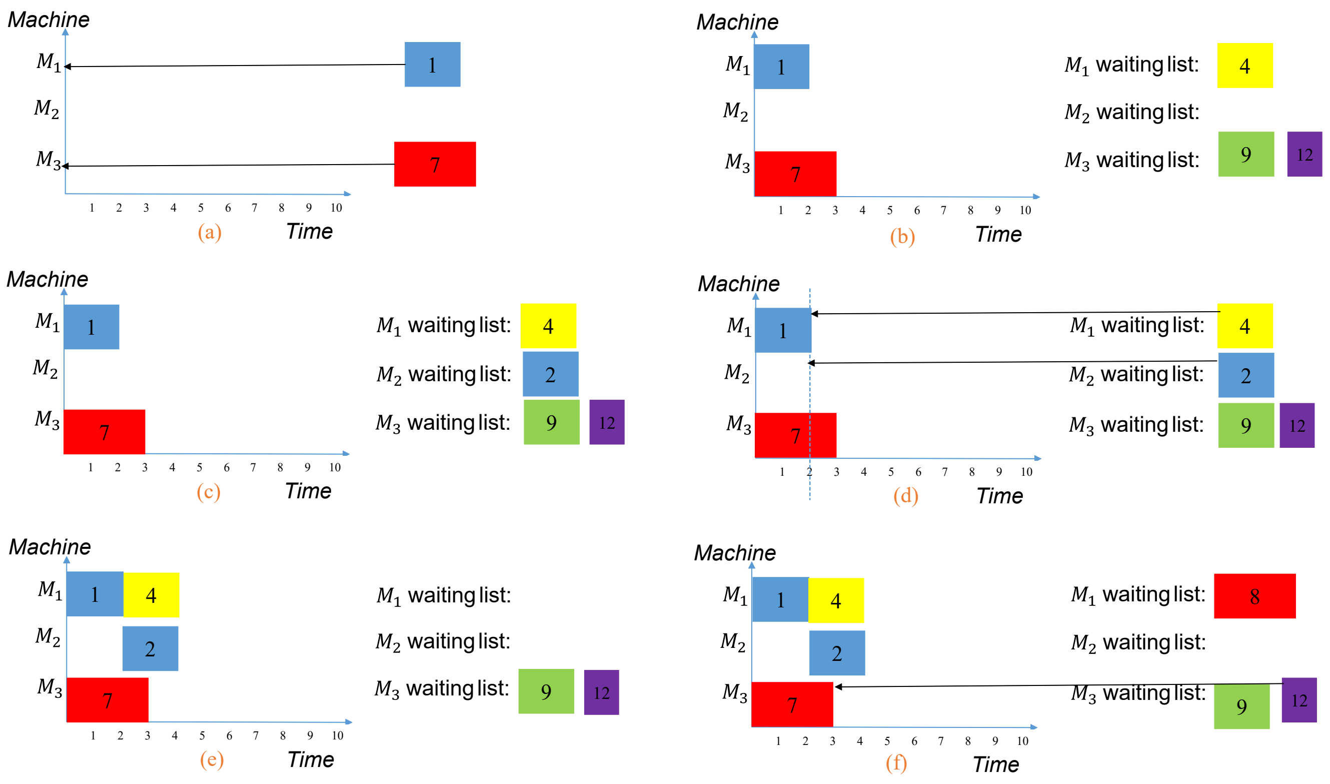


Figure 1. An example showing the stages of the simulation.

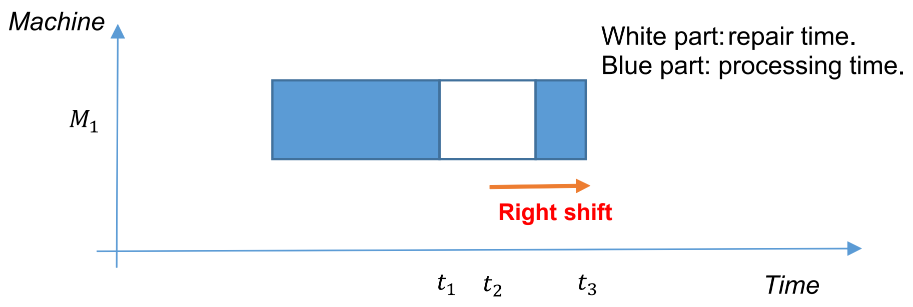


Figure 2. Right shift heuristic to include repair time into total processing time of an operation, in case of breakdown.

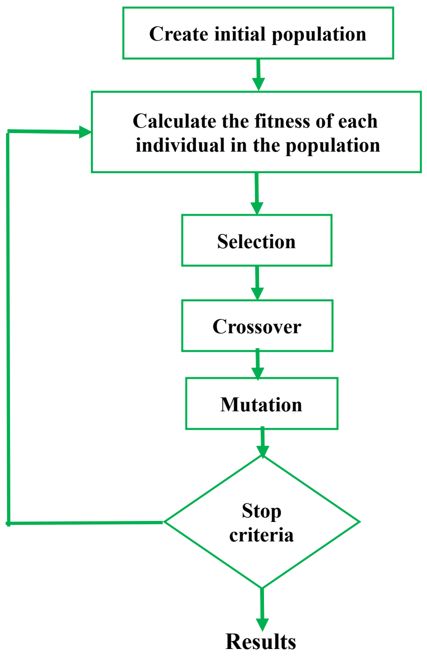

Create initial population
Calculate the fitness of each individual in the population
Selection
Crossover
Mutation
Stop criteria
Results

Figure 3. Flowchart of the evolutionary process in GEP.

- The structure of the implanted GEP is flexible, and the number of genes that generate the tree can be changed.

As depicted in Fig. 4, GEP is a population-based algorithm. In the current implementation, the initial population is built randomly.

Each individual in the population encloses a chromosome, which is the genotype of a DR. Related details are explained in Fig. 5.

By decoding each genotype, the corresponding phenotype, that is, the mathematical expression of the corresponding DR is obtained. For each of them, objective functions are calculated by running the simulation once, if the problem is deterministic, or multiple times if it contains stochastic occurrences. The population is sorted over this, and then a new population is generated using the crossover, and mutation operators. Thus, iterations continue until the stop condition, that is, the maximum number of iterations is met.

Element set, i.e. features that during the evolutionary process, GEP generates DRs combining them, are as follows:

- **Rt**: Release time.
- **Pr**: Processing time.
- **Nr**: Number of operations.
- **Rnr**: Number of remaining operations.
- **Rw**: Remaining workload.

As seen in Fig. 6. in the decoding method, at first, the location of the last gene is found and the expression making of DR starts from the last gene to the top. The reason for this is that although in each chromosome the total number of genes is constant, only some of them involved in the expression of DR. So, without finding the last gene, mathematical expression and also the corresponding tree cannot be formed.

Fig. 7 shows the tree resulting from the chromosome displayed in Fig. 5. As seen in Fig. 8 only the marked piece of the chromosome shown in Fig. 5 generates the tree. So the number of effective generals in the chromosome is seven.

There are already existing DRs in the simulation model, which are as follows [3]:

- For operation sequencing:
  - **FIFO**: first in first out.
  - **SPT**: shortest processing time.
  - **LnOp**: least operation number.
  - **LRnOp**: least remaining operation number.
  - **LTWK**: least total work content.
  - **LRWK**: least remaining work content.
- For machine assignment:
  - **LWK**: least waiting time.
  - **LPK**: least processing time.

Actually, LPT and SPT are mathematically the same but the first is used for machine assignment and the second for operation sequencing. The description of the LWP has been given earlier.

The included objective functions in the model are as follows:

- C_{max}: Maximum completion time of machines (makespan).
- W_{max}: Maximum machine workload.
- W_t: Total workload of machines.
- Fl: Average flow time of operations.
- F: Average of C_{max}, W_{max}, W_t and Fl.

It should be noted that DRs construct non-delay schedules. It means that when a machine is free, if there are jobs on the waiting list, one of them is selected with a rule and starts the process on the machine. Though the schedule obtained in this way may not be optimal, which can be found in the set of active schedules. If a schedule cannot be obtained by filling any left shifts, then it is an active schedule. Left shift is a heuristic to fill gaps by posterior operations. These relationships are shown in Fig. 9.

Although left shift is an important heuristic, it is not utilized in this study. Because if this heuristics is applied, there cannot be a fair comparison between the results of DRs. In Fig. 10, an example is provided to clarify the matter. Fig. 10 (a) shows a schedule with several gaps and in Fig. 10 (b) gaps are explored, in which operation 2 can be placed with a left shift. It should be pointed out that when scheduling with a DR is applied for scheduling, situations like Fig. 10 (a) do not arise, because it produces a non-delay schedule. This situation may occur when offline scheduling is done according to a predefined order of operations.

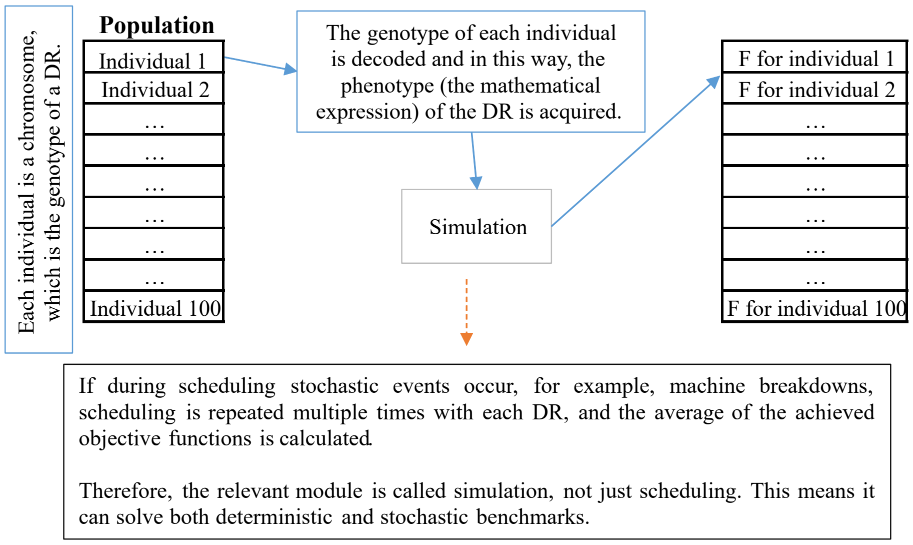

Each individual is a chromosome, which is the genotype of a DR.
Population
Individual 1
Individual 2
…
…
…
…
…
…
Individual 100
The genotype of each individual is decoded and in this way, the phenotype (the mathematical expression) of the DR is acquired.
Simulation
F for individual 1
F for individual 2
…
…
…
…
…
…
F for individual 100
If during scheduling stochastic events occur, for example, machine breakdowns, scheduling is repeated multiple times with each DR, and the average of the achieved objective functions is calculated.
Therefore, the relevant module is called simulation, not just scheduling. This means it can solve both deterministic and stochastic benchmarks.

Figure 4. Formation of population and usage of simulation to calculate objective functions at each iteration.

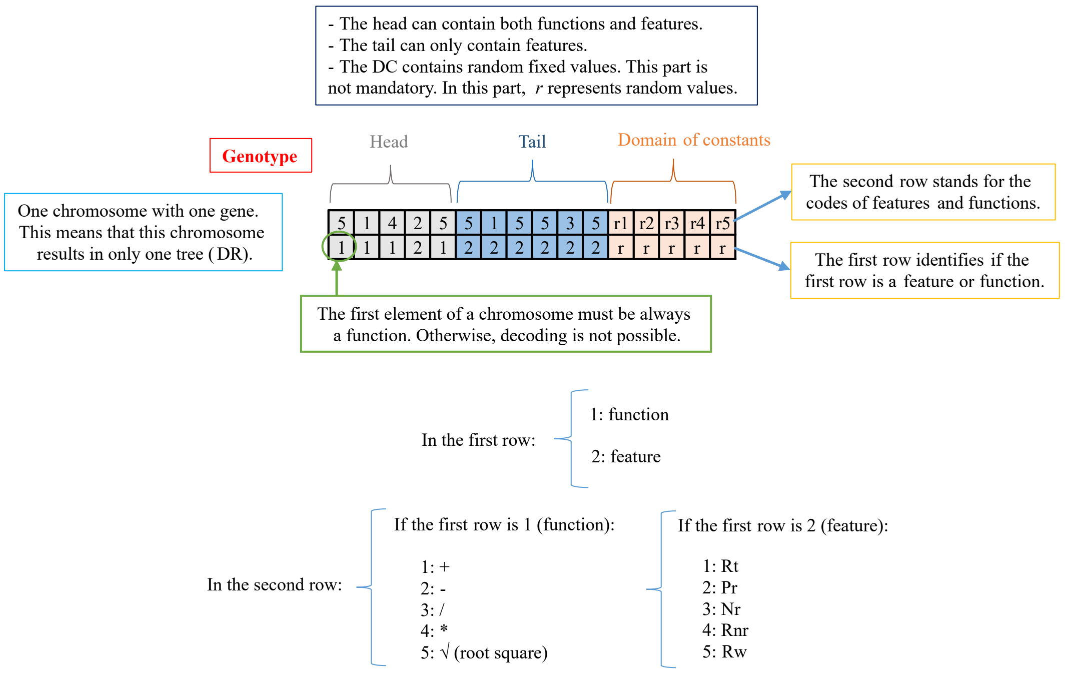
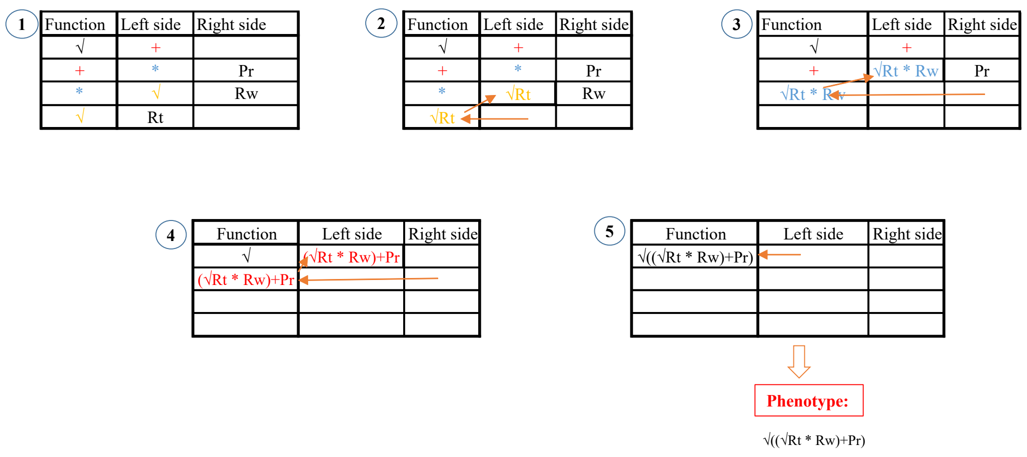

- The head can contain both functions and features.
- The tail can only contain features.
- The DC contains random fixed values. This part is not mandatory. In this part, r represents random values.
Genotype
Head
Tail
Domain of constants
One chromosome with one gene. This means that this chromosome results in only one tree (DR).
5 1 4 2 5 5 1 5 5 3 5 r1 r2 r3 r4 r5
1 1 1 2 1 2 2 2 2 2 2 r r r r r
The second row stands for the codes of features and functions.
The first row identifies if the first row is a feature or function.
The first element of a chromosome must be always a function. Otherwise, decoding is not possible.
In the first row:
1: function
2: feature
In the second row:
If the first row is 1 (function):
1: +
2: -
3: /
4: *
5: √ (root square)
If the first row is 2 (feature):
1: Rt
2: Pr
3: Nr
4: Rnr
5: Rw

Figure 5. Structure of a chromosome in GEP to evolve a DR.



1

Function	Left side	Right side
√	+	
+	*	Pr
*	√	Rw
√	Rt	

2

Function	Left side	Right side
√	+	
+	*	Pr
*	√Rt	Rw
√Rt		

3

Function	Left side	Right side
√	+	
+	√Rt * Rw	Pr
√Rt * Rw		

4

Function	Left side	Right side
√	(√Rt * Rw)+Pr	
(√Rt * Rw)+Pr		

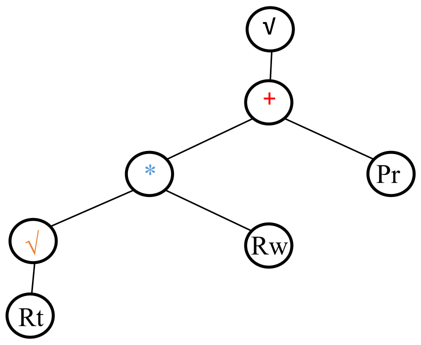
5

Function	Left side	Right side
√((√Rt * Rw)+Pr)		

Phenotype:

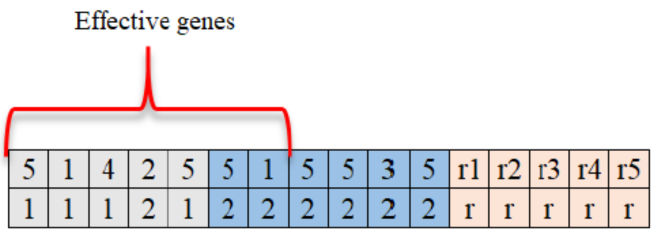
√((√Rt * Rw)+Pr)

Figure 6. An example for the decoding process.



This chromosomes results in a single tree, but in the literature there other types of chromosomes that contain multiple sub -trees.

Figure 7. The acquired tree, as the result of the decoding in Fig. 6.


Effective genes

5	1	4	2	5	5	1	5	5	3	5	r1	r2	r3	r4	r5
1	1	1	2	1	2	2	2	2	2	2	r	r	r	r	r

Figure 8. Involved genes in the formation of the tree.

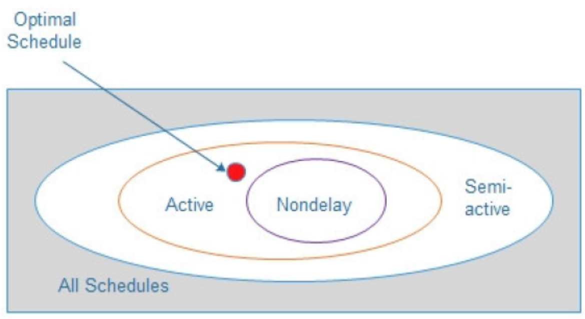
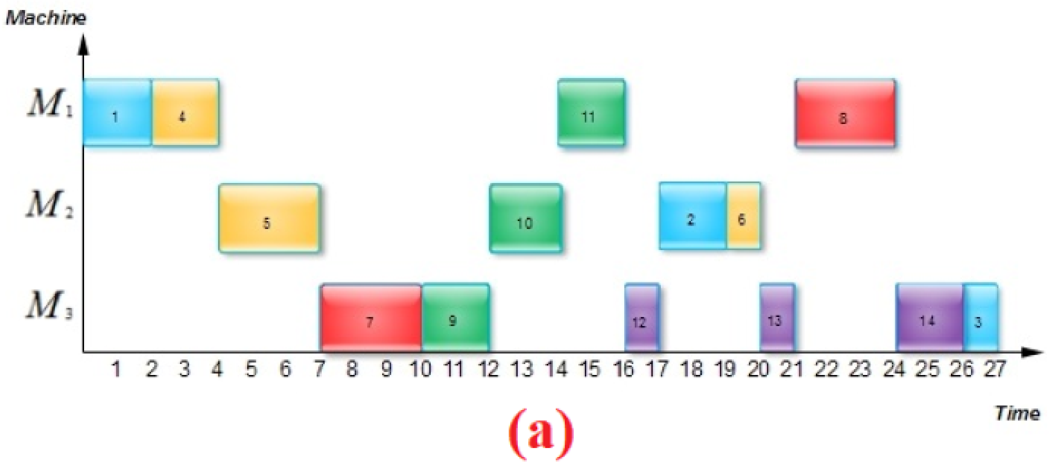


Figure 9. Classification of schedules.



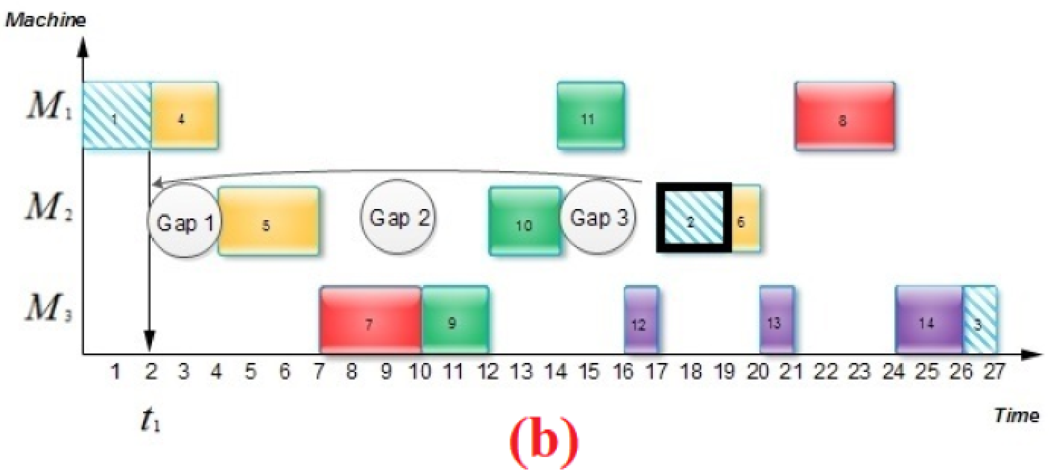
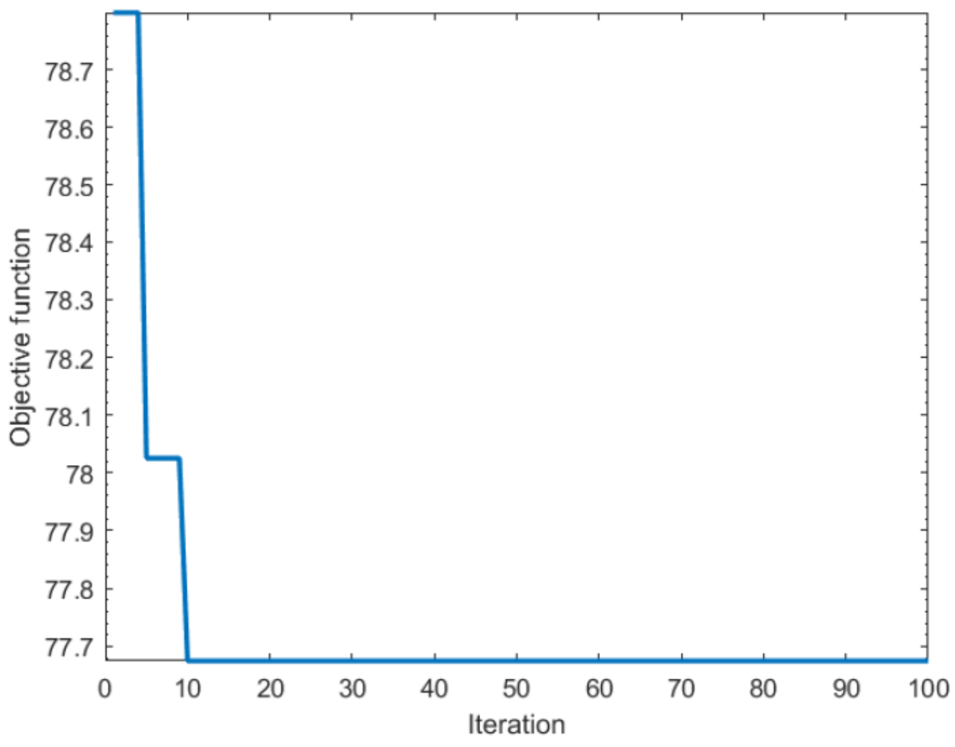
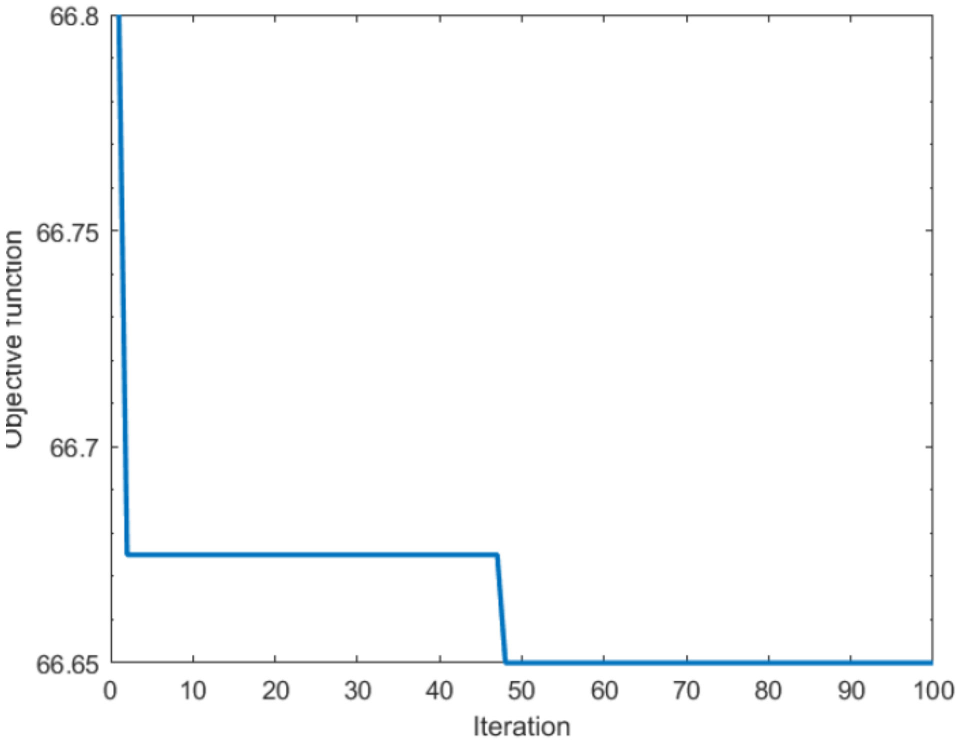


Figure 10. (a) A schedule with several gaps, (b) looking for appropriate gaps where O_2 can be placed with the left shift.



(a)



(b)

Figure 11. Convergence of evolutionary process of (a) GEP+LWT, and (b) GEP+LPT for MK2.

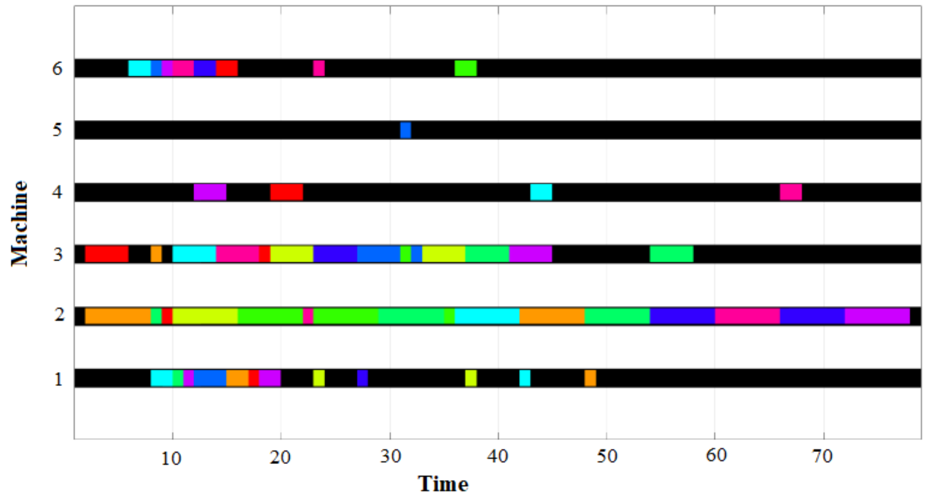


Figure 12. Gantt chart of schedule obtained by GEP+LWT for MK1.

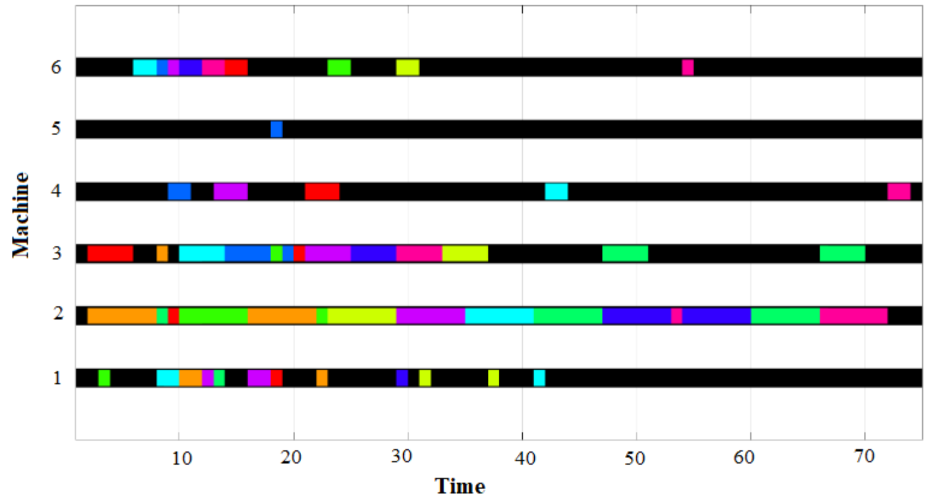


Figure 13. Gantt chart of schedule acquired by GEP+LPT for MK1.

Table 1. Acquired results.

Benchmark	**Rule**	**Objective Function**					T
		C_{max}	W_t	W_{max}	Fl	F	
MK-1	FIFO+LWT	90	163	88	59.8	100.2	≈ 0
	FIFO+LPT	74	153	70	48.1	86.27	≈ 0
	SPT+LWT	92	170	88	53.6	100.9	≈ 0
	SPT+LPT	74	153	70	42.2	84.8	≈ 0
	LnOp+LWT	105	170	100	60.1	108.77	≈ 0
	LnOp+LPT	75	153	70	43.4	85.35	≈ 0
	LRnOp+LWT	117	174	106	61.1	114.52	≈ 0
	LRnOp+LPT	81	153	70	43.4	86.85	≈ 0
	LTWK+LWT	105	195	94	52.6	104.15	≈ 0
	LTWK+LPT	81	153	70	40.2	86.05	≈ 0
	LRWK+LWT	91	168	88	54.2	100.3	≈ 0
	LRWK+LPT	80	153	70	41.5	86.12	≈ 0
	GEP+LWT	78	161	76	50.1	91.27	297.81
	GEP+LPT	**74**	**153**	**70**	**41.3**	**84.57**	175.58
MK-2	FIFO+LWT	74	152	72	56.2	88.62	≈ 0
	FIFO+LPT	50	**140**	**45**	38.5	68.37	≈ 0
	SPT+LWT	78	150	67	42.2	84.3	≈ 0
	SPT+LPT	58	**140**	**45**	32.8	68.95	≈ 0
	LnOp+LWT	77	153	73	52	88.75	≈ 0
	LnOp+LPT	51	**140**	**45**	36.4	68.1	≈ 0
	LRnOp+LWT	77	149	69	45.1	85.02	≈ 0
	LRnOp+LPT	58	**140**	**45**	34.6	69.4	≈ 0
	LTWK+LWT	71	151	67	41.2	82.55	≈ 0
	LTWK+LPT	**50**	**140**	**45**	33.5	67.12	≈ 0
	LRWK+LWT	77	152	71	43.6	85.9	≈ 0
	LRWK+LPT	58	**140**	**45**	32.4	68.85	≈ 0
	GEP+LWT	67	149	56	38.7	77.67	290.64
	GEP+LPT	51	**140**	**45**	**30.6**	**66.65**	302.74
MK-3	FIFO+LWT	446	884	432	354.13	529.03	≈ 0
	FIFO+LPT	343	**812**	**330**	252.27	434.32	≈ 0
	SPT+LWT	436	880	423	255.86	498.72	≈ 0
	SPT+LPT	343	**812**	**330**	**194.07**	419.77	≈ 0
	LnOp+LWT	496	896	459	258.87	527.47	≈ 0
	LnOp+LPT	343	**812**	**330**	200.2	421.3	≈ 0
	LRnOp+LWT	496	872	432	249.27	505.57	≈ 0
	LRnOp+LPT	362	**812**	**330**	201.4	426.35	≈ 0
	LTWK+LWT	435	875	406	239.13	488.78	≈ 0
	LTWK+LPT	367	**812**	**330**	200.13	427.28	≈ 0
	LRWK+LWT	430	869	404	229.4	483.1	≈ 0
	LRWK+LPT	359	**812**	**330**	195.73	424.18	≈ 0
	GEP+LWT	372	864	367	245.27	462.06	1299.56
	GEP+LPT	**335**	**812**	**330**	195.26	**418.07**	714.25
MK-4	FIFO+LWT	191	328	**188**	113.07	205.02	≈ 0
	FIFO+LPT	**189**	**324**	**188**	112.53	203.38	≈ 0
	SPT+LWT	192	328	**188**	98.87	201.72	≈ 0
	SPT+LPT	192	**324**	**188**	98.6	200.65	≈ 0
	LnOp+LWT	197	332	**188**	86.87	200.97	≈ 0
	LnOp+LPT	197	**324**	**188**	86.47	198.87	≈ 0
	LRnOp+LWT	197	328	**188**	89.4	200.6	≈ 0
	LRnOp+LPT	197	**324**	**188**	88.93	199.48	≈ 0
	LTWK+LWT	197	328	**188**	82.87	198.97	≈ 0
	LTWK+LPT	197	**324**	**188**	**82.53**	197.88	≈ 0
	LRWK+LWT	197	332	**188**	89.07	201.52	≈ 0
	LRWK+LPT	197	**324**	**188**	87.47	199.12	≈ 0
	GEP+LWT	191	328	**188**	85.66	198.17	355.93
	GEP+LPT	191	**324**	**188**	85.46	**197.12**	482.69
MK-5	FIFO+LWT	306	690	304	243.6	385.9	≈ 0
	FIFO+LPT	241	**672**	239	191.07	335.77	≈ 0
	SPT+LWT	273	690	254	169.93	346.73	≈ 0
	SPT+LPT	261	**672**	239	151.4	330.85	≈ 0
	LnOp+LWT	296	693	276	160.73	356.43	≈ 0
	LnOp+LPT	268	**672**	239	142.13	330.28	≈ 0
	LRnOp+LWT	297	698	295	162.6	363.15	≈ 0
	LRnOp+LPT	271	**672**	239	138.47	330.12	≈ 0
	LTWK+LWT	302	698	262	154.87	354.22	≈ 0
	LTWK+LPT	268	**672**	239	141.07	330.02	≈ 0
	LRWK+LWT	339	695	285	170.87	372.47	≈ 0
	LRWK+LPT	271	**672**	239	**138.2**	330.05	≈ 0
	GEP+LWT	**220**	697	**219**	175.47	327.87	642.10
	GEP+LPT	244	**672**	239	142.8	**324.45**	509.25

3 Experimental Results

The algorithms described in Section 2. are implemented in MATLAB. A system with an Intel Core i5 processor, 2.4 GHz with 12 GB of RAM is utilized. MK set is employed as the benchmark. In GEP, the population number is chosen as 100, the maximum number of iterations is 100, the crossover rate is 1, the mutation rate is 0.4, and the head size in the chromosome is 5. As seen in Fig. 5, the number of functions and the number of features in element size are both equal to 5. The domain of constants (DC) part appears in the same figure, which, although applicable in the utilized code, is not used because the application of fixed values in DR does not make sense.

The gained results are given in Table 1. where the best outcomes are bold. The last column of the same table shows processing times in seconds. As seen, due to GEP's evolutionary process, the duration is longer than others for classic DRs. Convergences of GEP+LWT and GEP+LPT for MK2 are shown in Fig. 11.

As seen, the best values are got by GEP+LPT, which also signifies that applying different rules for sub-problems of machine assignment and operation sequencing is influential. Gantt charts of schedules acquired by GEP+LWT and GEP+LPT for MK1 are shown in Fig. 12 and Fig. 13.

As said before, the simulation model can also handle buffer states and machine breakdowns, and even can solve JSSP. Details, benchmarks, and results on all these are available from the corresponding author's email address. Nevertheless, it should be noted that even when these situations are considered, similar results will arise and there are studies on them in the literature [3].

4 Conclusion and Future Works

This study has two principal goals, which are (i) providing an open-source scheduling model to solve various types of scheduling problems, and (ii) analyzing the effect of solving machine assignment and operation sequencing sub-problems in FJSSP with different rules. The presented open-source simulation model is capable to solve different types of FJSSP and JSSP. Furthermore, it is easy to develop and modify it for other types of scheduling problems. Using the simulation model, it is shown that solving two sub-problems of FJSSP with different rules is meaningful. Generally, machine assignment and operation sequencing sub-problems of FJSSP are solved with the same rule or a specific one. The results of this study, which are gained by applying different rules to the two sub-problems of FJSSP show that it is possible to improve outcomes by involving a proper rule for machine assignment part as well as operation sequencing. One of the important parts of the simulation model is to evolve DRs using GEP for the operation sequencing sub-problem of FJSSP. As already mentioned, the model is able to apply different rules to the sub-problems, however, in the current implementation, there are only two rules applicable to the machine assignment part. This can be considered as a limit for this study, but in future studies, it is planned to extend GEP and evolve DRs to the machine assignment sub-problem.

Acknowledgments

Financial support from Fundação para a Ciência e Tecnologia (through project UIDB/00731/2020) is gratefully acknowledged.

Jie Li appreciates financial support from Engineering and Physical Sciences Research Council (EP/T03145X/1).

The authors would like to thank the editors and the anonymous referees for their valuable comments which helped to significantly improve the manuscript.

Supporting Information

All codes are accessible to readers via the following public link on GitHub:

https://github.com/aydinteymurifar/Scheduling-Source-Codes

REFERENCES

[1] A. Teymourifar, G. Ozturk, and O. Bahadir. A Comparison between Two Modified NSGA-II Algorithms for Solving the Multi-objective Flexible Job Shop Scheduling Problem, Universal Journal of Applied Mathematics, 6(3), pp. 79-93, 2018.

[2] G. Ozturk, O. Bahadir, and A. Teymourifar. Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming, International Journal of Production Research, 57(10), pp. 3121-3137, 2019.

[3] A. Teymourifar, G. Ozturk, Z. K. Ozturk, and O. Bahadir. Extracting new dispatching rules for multi-objective dynamic flexible job shop scheduling with limited buffer spaces, Cognitive Computation, 12(1), 195-205, 2020.

[4] A. Teymourifar, G. Ozturk. A neural network-based hybrid method to generate feasible neighbors for flexible job shop scheduling problem, Universal Journal of Applied Mathematics, 6(1), 1-16, 2018.

[5] A. Teymourifar, O. Bahadir, and G. Ozturk. Dynamic Priority Rule Selection for Solving Multi-objective Job Shop Scheduling Problems, Universal Journal of Industrial and Business Management, 6(1), 11-22, 2018.

[6] A. Teymourifar. Dinamik atölye çizelgeleme problemleri için taslim zamanı belirleme ve yeni sevk etme kuralları, Master dissertation, Anadolu University, Turkey, 2015.

[7] A. Teymourifar, G. Ozturk, New dispatching rules and due date assignment models for dynamic job shop scheduling problems, International Journal of Manufacturing Research, 13(4), 302-329, 2018.

[8] A. Teymourifar, A. M. Rodrigues, and J. S. Ferreira. A comparison between simultaneous and hierarchical approaches to solve a multi-objective location-routing problem, In Graphs and Combinatorial Optimization: from Theory to Applications, Springer, Cham, 251-263, 2021.

[9] M. Thenarasu, K. Rameshkumar, J. Rousseau, and S. P. Anbuudayasankar, Development and analysis of priority decision rules using MCDM approach for a flexible job shop scheduling: A simulation study, Simulation Modelling Practice and Theory, 114, 102416, 2022.

[10] N. Rakovitis, D. Li, N. Zhang, J. Li, L. Zhang, and X. Xiao, Novel approach to energy-efficient flexible job-shop scheduling problems, Energy, 238, 121773, 2022.

[11] M. Oltean, C. Grosan, A comparison of several linear genetic programming techniques, Complex Systems, 14(4), 285-314, 2003.

Appendix

DC: Domain of Constant Values
DR: Dispatching Rule
F: Objective Function
FIFO: First In First Out
Fl: Average Flow Time of Operations
GA: Genetic Algorithm
GEP: Gene Expression Programming
GP: Genetic Programming
LnOp: Least Operation Number
LPT: Least Processing Time
LRnOp: Least Remaining Operation Number
LRWK: Least Remaining Work Content
LTWK: Least Total Work Content
LWT: Least Waiting Time
SPT: Shortest Processing Time