International Journal of Hybrid Information Technology
Vol.8, No.12 (2015), pp. 191-204
http://dx.doi.org/10.14257/ijhit.2015.8.12.13

Job Shop scheduling by Using the Combinational Viruses Evolutionary and Artificial Immune Algorithms in Dynamic Environment

Zohreh Davarzani, Soheila Staji and Fahimeh Dabaghi-Zarandi

Payame Noor University, POB 19395–3697 Tehran, Iran
Technical and vocational university of Sabzevar academy, Iran
Faculty of Computer Engineering Department, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran
Davarzani.z@Pnurazavi.ac.ir, soheila_staji@yahoo.com, f.dabaghi@vru.ac.ir

Abstract

This article deals with solving the flexible job shop scheduling problem in dynamic environment (DFJSSP). In this problem, environment may face with many real time events such as random arrival of jobs or breakdown machine efficiency. Jobs and their operations are processing the machines according to static scheduling in which environment might face with such events. Regarding being NP – hard of the problem , a hybrid of artificial immune and virus evolutionary algorithm are offered to solve it which use the technique of stable action – reaction scheduling . In these algorithms two objective functions are minimized: Efficiency and stability. Efficiency is the objectives value in static scheduling, whereas stability is presented in dynamic scheduling because its purpose is to improve static scheduling, reduce the deviation from the first scheduling, and increase system stability.

Keyword*: Dynamic job shop scheduling problem, Virus evolutionary algorithm, Artificial immune algorithm*

1. Introduction

System scheduling is considered one of the significant problems in manufacturing systems. Namely, like processors and available resources, the system scheduling is one of the essential problems in today's world and their effective scheduling causes rise in efficiency, proper use of resources, cut in time of jobs completion and generally systems profitability.

Job shop scheduling problem (JSSP) is one of the commonest problems of manufacturing systems that is used in most manufacturing environments. In such problems a number of machines are available in the shop that each job depending on its type of operations needs to use one or more machines in a particular sequence. One of the limitations of this type is that in each moment each job can be done only with one machine and each machine is just able to process one operation at each moment.

In general, these problems can be studied from two aspects: Static and dynamic scheduling. In static scheduling, all jobs, machines and prior information on the behavior and time of their process are available and no uncertainty for future exists in such a way that the determined scheduling can process without any change on machines. In DJSS problem, one or more conditions of the problem like number of jobs, random arrival of job or machine breakdown.

Are changed by any new event. In other words, there is uncertainty in it. Therefore the solution before the event is not good or even feasible longer. So in addition to scheduling

ISSN: 1738-9968 IJHIT

problem, it is needed to deal with dynamic events. These events are divided into two groups:

Resources – based events: these kinds of events are in relation with machines which are diminishing machine efficiency, adding a machine, and uncertainty about breaking down the tools.

Job based events: These are related to jobs and include: work deletion, changing the job deadline, sooner or later arrival, change in priority level and / or change in its process time.

The first study in dynamic job shop scheduling was published by Holloway and Nelson [1]. They implemented a multi-pass procedure by generating schedules periodically. They concluded that a periodic policy (scheduling/rescheduling periodically) is effective in dynamic job shop environments. Fang and Xi [2] provided a meta-heuristic method for the dynamic job shop scheduling problem. They presented a hybrid method based on the genetic algorithm and dispatching rules for solving job shop scheduling problems with sequence-dependent setup times and due date constraints. Chryssolouris and Subramaniam [3] presented a genetic algorithm for DJSP. They considered two performance measures, named mean job tardiness and mean job cost, to demonstrate multiple criteria scheduling. They indicated that the genetic algorithms scheduling approach produced better scheduling performance in comparison with several common dispatching rules. Adibi, *et al.,* [4] presented a variable neighborhood search method for a multi-objective DJSSP. They considered a JSP with random job arrivals and machine breakdowns. Their multiobjective performance measure consisted of two efficiency criteria (Makespan and Tardiness). Rangsaritratsamee, *et al.,* [5] indicated that the strategies applied in previous researches on DJSSP can improve classic measures of efficiency. They presented a methodology to address DJSSP based on a bi-criteria objective function that simultaneously considers efficiency and stability.

The remainder of this paper is organized as follows: problem definition is given in Section 2. Section 3 introduces main concepts of virus evolutionary algorithm. Section 4 is consisted of two parts: in part 4-1, random job arrival is considered and in part 4-2, machine break down is studied. The experimental results are given in Section 6.

2. Dynamic Job Shop Scheduling Problem

As said previously, manufacturing system are by nature with dynamism and might face with several real time events. Thus, their scheduling is more complex with regard to static condition. The proposed algorithm is shown in Figure 1. This algorithm is implemented in two steps:

- Scheduling of job by static algorithm proposed by Davarzani [6].
- Improve the static scheduling in dynamic environment by new events.

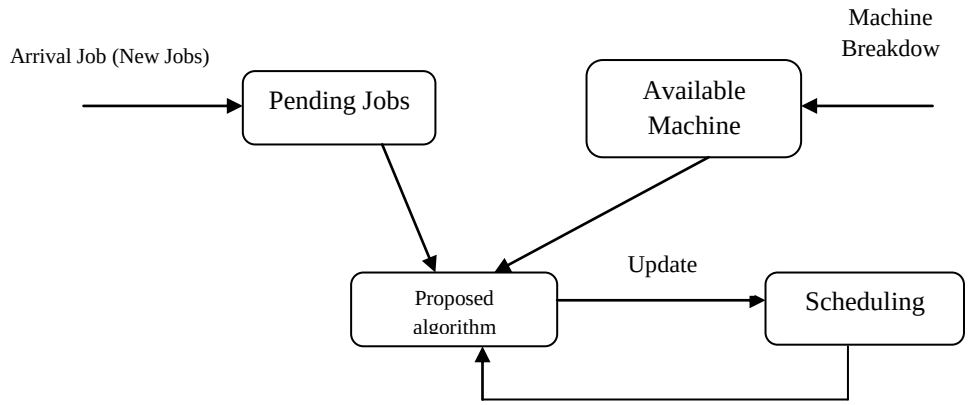

Figure 1. Flowchart of Dynamic Job Shop Scheduling

In this structure the static scheduling proposed by Davarzani [6] is created. The new jobs which arrive in system are added to list of the jobs waiting to be scheduled and changes in the set of machines likes machine breakdown, are added to the machines information. Then the static scheduling improves with these new changes. In this system 3 kinds of jobs are available in the environment. They are:

1-The jobs that are scheduled and processed.
2-The jobs that are scheduled but not processed (waiting jobs).
3-The new jobs which have arrived and not yet scheduled (new jobs).

The first type is the jobs which are processed and not present in the new scheduling. The second type is the jobs that are scheduled by static scheduling and are ready for process. The third type is the jobs which have just arrived the environment and not yet scheduled.

There are two criterions in dynamic job shop for optimizing. These purposes are:
1-Efficiency: the efficiency criterions are the criteria exist in static scheduling and it consists of:

- Minimizing the completion time of the jobs which is defined as equation 1:

$$F_1 = min(C_{max}) \quad (1)$$

- Minimizing delay of job with maximum delay that is defined as following:

$$F_2 = max(0, C_i - d_i) \quad (2)$$

- Minimizing the percent of delayed jobs as following:

$$F_3 = \frac{n_t}{n} \times 100 \quad (3)$$

2- Stability: This criterion is defined in dynamic scheduling that causes CPU used energy to be saved and it is described as the difference in the starting time of jobs in the old and new scheduling. This purpose is indicated in equation 4.

$$F_4 = \sum_i \sum_j |t'_{ij} - t_{ij}| \quad (4)$$

3. Virus Evolutionary Algorithm (VEA)

To solve the problem in dynamic environment, the combination of artificial immune and Virus Evolutionary Algorithms called Virus Immune Algorithm is used. VEA uses the theory of viruses evolution in human body. Viruses that attack human body are special type of genes which can be useful and useless for human body. Figure 2 shows this algorithm. As shown this Figure, this algorithm has two host and virus populations. The

main operator of VEA is infection. During each generation, viruses transfer to the host and according to their life time they stay alive and transfer to the next generation [7].

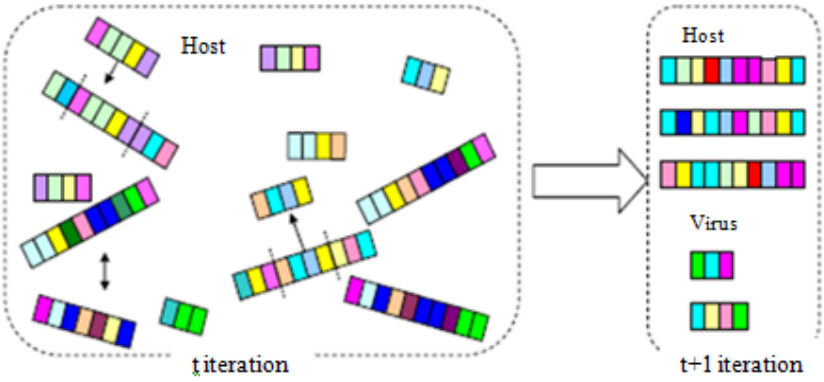

Figure 2. Virus Evolutionary Algorithm

4. Proposed Algorithm Structure

As mentioned, two algorithms have been presented to face with real time events to solve the problem. Algorithm 1 (Figure 3) examines random arrival of jobs in the environment and in algorithm 2, machine breakdown are studied.

```
1-  Procedure: Proposed Virus Immune Algorithm (VIA)
2-  Input: Max_Gen, number_clone, rate_hypermutation, population_host, population_virus
3-  Output: Best schedule
4-  Begin
5-     Number _Generation ← 0,
6-     Initialize host and virus population from pending and new jobs;
7-        While (Number_Generation ≤ max_Gen) do
8-           Virus infection procedure;
9-           new_population← Clonal selection and hypermutation on host population;
10-          Merge new_population and current population;
11-          Create next population by roulette_wheel_selection;
12-          Number_Generation=Number_Generation+1;
13-       End while;
14-    Output: Best schedule
15- End;
```

Figure 3. Proposed Algorithm

4.1. Part 1: Algorithms 1

The general structure of algorithm1 is demonstrated in Figure 3. In this algorithm, at first the waiting jobs are specified by static scheduling and their scheduling map for producing the host population is kept. Consequently, the host population in this algorithm contains the scheduling of the waiting jobs which have been made of static scheduling. The virus population includes the population of new jobs which have just arrived the environment. In other words, the new jobs are considered as the virus population which attacks the host population.

4.1.1. Initializing the Host and Virus Population

To initialize the virus and host population, it is assumed at time t, a number of jobs have been arrived the environment to be scheduling. At this time, the processing of some operations has finished and some of them have also scheduled and wait to be processed.

As mention, the host population is created by the waiting jobs for processing (static scheduling) and virus population by the new arrived jobs in the environment.

4.1.2. Calculating Fitness Function

Generally speaking, the number of virus transfers to host depends on virus infection (e). The infection is measured by using the amount of a virus fitness function which is represented by ***fitvirusi***. If the amount of virus fitness function is low, the infection is raised and the virus chance for transferring into host is more and in case the amount of virus fitness function is high the infection is reduced and its chance to transfer decreases. The amount of virus fitness function is obtained by equation 5.

$$fitvirus_i = \sum_{j \in S} (FithostPre_{i,j} - Fithostaft_{i,j}) \quad (5)$$

In this equation, *and* respectively show *jth* host fitness function prior to ith virus transfer and after it, and **s** includes the set of host antibodies to which ith virus has been transferred. Since antibodies only contain the data on the sequence of operations, to assign the operations to machines, the method said in [12] is utilized. Also because this problem is multipurpose, the adaptive weighting method has been employed. Therefore, the amount of virus fitness function is estimated by the equation 6:

$$Fithost = w_1 \times F_1 + w_2 \times F_2 + w_3 \times F_3 + w_4 \times F_4 \quad (6)$$

In this equation, s are the weights of fitness function and each of F_i s is the amount of the objective functions that are expressed in section 3. Following the calculation of each virus fitness function amount through using the equation 5, the ***life time*** amount of each virus is counted by the equation 7:

$$LF_{i,t+1} = e \times LF_{i,t} + 1/fitvirus_i \quad (7)$$

4.1.3. The Operators of Virus Evolutionary Algorithm

One of the main operators of virus evolutionary algorithm is transfer operator that its general structure is shown in picture 5. In this algorithm, ***Transduction*** and ***Reverse-transcription*** are two main operators of virus transfer which have been used as the transfer operators. According to reverse transcription operator, a virus transfers into host antibody. Then the amount of virus and host fitness function are measured and from them the amount of $LF_{i,t+1}$ is obtained. If the amount of $LF_{i,t+1}$ is negative , this virus is deleted and a new virus which is by chance chosen from a host antibody is created and in case this amount is positive, virus is created by one of the antibodies which are transferred to it. In Figure 4 and 5, the main transfer operators are demonstrated. In the proposed algorithm, the virus and host population are considered as the sequence of waiting to be processed and newly arrived jobs.

Transduction Operator

In this operator, one virus is made of one host. That is, after completion of each generation and doing the immune system operators, a set of viruses is made of hosts. Figure 4 shows an example of this operator.

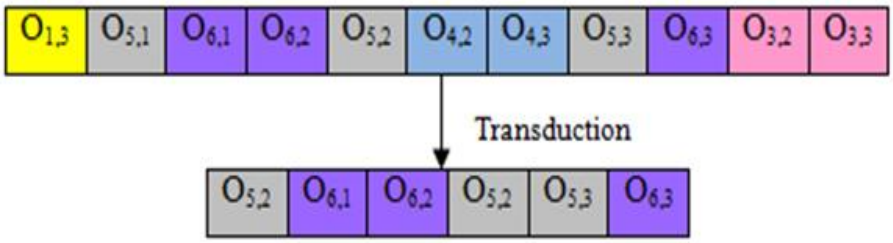

Figure 4. Transduction Operator

Reverse Transcription Operator

In this operator, a virus is selected based on the amount of its lifetime and is transferred into a host. The number of the points that a virus transfers to a host is coincidentally selected. Figure 5 indicates an example of this operator. As said above, in this algorithm the new jobs are counted as the viruses population that arrive the environment and the initial scheduling is considered as the host population.

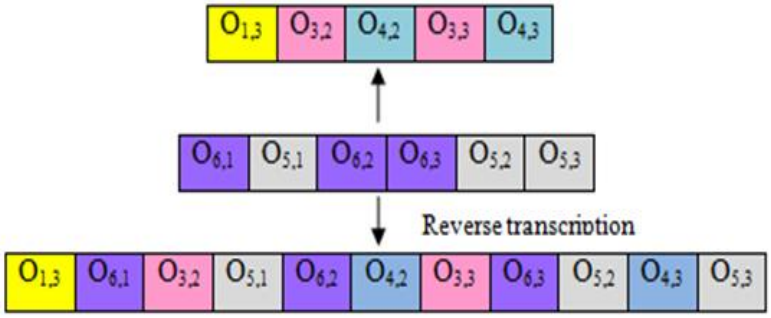

Figure 5. Reverse Transduction Operator

4.1.4. Immune System Operators

As the expressive algorithm in Figure 3 shows, after viruses transfer into hosts, the amount of hosts fitness function is measured and the immune system operators are applied on the present population . A host with the best fitness function is localized in memory (M). The available antibodies in M have higher probability of proliferation. The ***nm*** copies of them are created and put in set C. The available antibodies in set C mutate and replace the antibodies with less proximity in host population [12]. In this algorithm, the following mutation operators are used to make variation in population:

- *PPS11 (*precedence preserving shift) mutation operator that is offered by lee [8]:
- Gene shift operator: In this operator, two subsets of antibody in same length are selected and these two subsets are replaced.
- Gene inversion operator: A subset of genes is selected in this operator and their inversion is placed in the host antibody.

4.2. Part 2: Algorithm 2

As it was said in the previous section, two occurred incidents which cause the environment to be dynamic are: the new jobs arrival and slowdown in machines efficiency. In this part, the effect of slowdown in machines efficiency has been studied. Like the offered algorithm in the previous section, to solve this problem, a Virus Immune Algorithm (VIA) is also suggested.

In this problem, each machine has two parameters: Mean Time between Failures

(MTBF) and Mean Time To Repair (MTTR). In addition, the amount of these two parameters is presumed not to be fixed for different machines.

The proposed algorithm is like the offered algorithm in Section 4-1, but the inversion transfer operator and the way to assign operations to machines are different in it. Suppose that at moment *T*, a number of *MB* machine fails; the operations that are between machines MTBF and MTTR form BR set and the rest of the operations which are waiting to be processed from moment *T* on are put in *RE* set. The present operations in *BR* form the virus population and the remain operations waiting to be processed comprise the host population.

To understand the algorithm better, an example is given in the following. Suppose that in Figure 6 when $T=4$, m_6 and m_7 machines are broken down. At that moment m_6 and m_7 machines respectively process $O_{15,3}$ and $O_{9,3}$ operations. M_6 machine needs 2 and m_7 machine requires 4 time units to be repaired. So, $MTTR_6$ equals 2 and $MTTR_7$ 4 time units. After repair, m_6 machine is processing $O_{15,3}$ and m 7 machine processes $O_{12,4}$. In Table 5-5, the order of the available operations in *BR* and *RE* sets being obtained from the initial scheduling is indicated.

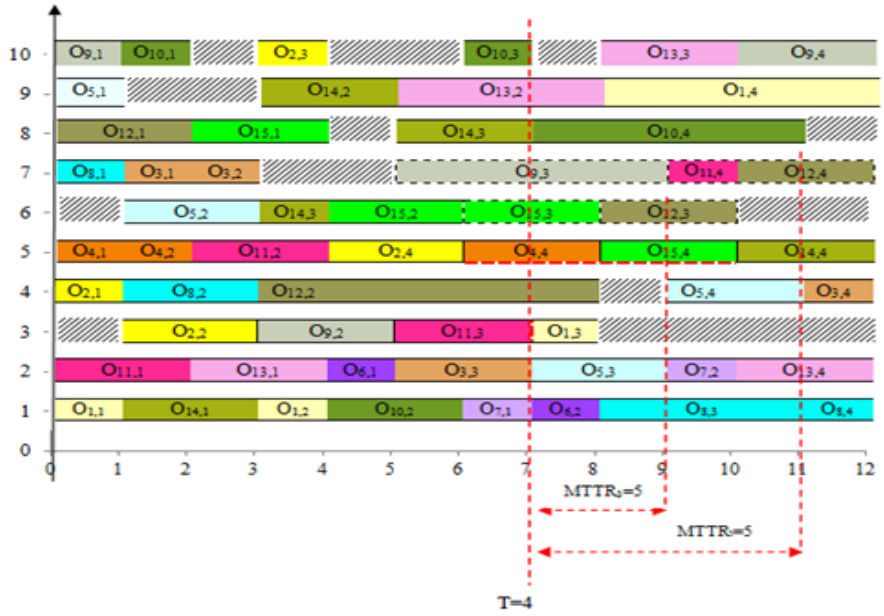

Figure 6. An Example of Breakdown m_6 and m_7 in Time T=4

Table 1. The Amount of BR and RE from Previous Example in Figure 6

BR	$\{O_{15,3}, O_{12,3}, O_{9,3}, O_{11,4}, O_{12,4}\}$
RE	$\{O_{6,2}, O_{8,3}, O_{8,4}, O_{5,3}, O_{7,2}, O_{13,4}, O_{1,3}, O_{5,4}, O_{3,4}, O_{15,4}, O_{14,4}, O_{10,4}, O_{1,4}, O_{13,3}, O_{9,4}\}$

Because there is relation between the operations of jobs in virus and host populations and since the precedence priorities at transferring time of virus into host must be observed, by following these priorities, the transferring operation of virus into host population need to be carried out.

4.2.1. Reverse Transfer Operator

As stated above, there are prerequisite priorities among operations in *BR* and *RE* sets. These priorities must be observed as viruses transfer into host. Suppose that virus v_i wants to transfer into host h_i. The following procedure indicates how operations v_i transfer to *hi*.

1- For each operation like $O_{i,j}$ in v_i, repeat stages 2 and 3:

2- Determine the place of preceding operation ($O_{i,j-1}$) and antecedent operation ($O_{i,j+1}$) in the host antibody if it exists and respectively name *lmp* and *rmp* . If this operation has no preceding element, *lmp* equals 1 and if it does not have the antecedent element, *rmp* equals *nu-op* where *nu-op* is the number of the host antibody operations.

3- Select a coincidental number in range [rmp, *lmp*], include $O_{i,j}$ in this place and shift the rest of operations to the right . One unit adds to *nu-op*.

Picture 18-5 shows an example of this operator. Suppose that the virus which wants to transfer into host includes the operations sequence $\{O_{6,3},O_{5,3},O_{7,4},O_{9,4}\}$. This picture shows how $O_{6,3}$ is infect to this host.

Figure 7. Infecting of $O_{6,3}$ into the Host

4.2.2. Operation Assignment to Machines

As previously said, the host antibodies only contain the data of the operation sequence and no data of assigning machines exists in them. These data are obtained in antibodies decoding phase. In this Section, an innovative algorithm is offered to assign operations to machines as following:

- **Assigning Algorithm of Operations to Machines**

1- for each machine like *i* , its Release Time (*RT*)is measured according to equation 8:

$$RT(i) = BRT(i) + MTTR(i) \quad (8)$$

In this equation, *MTTR (i)* is the required time to repair the *i th* machine and *BRT (i)* is the time when the *i th* machine fails.

2- Repeat stages 2 to 4 for each $O_{i,j}$ operation in an order that was given in h_i.

3- Suppose that $U_{i,j} \subseteq U$ is the set of machines which $O_{i,j}$ can process on. The completion time $O_{i,j}$ on each machine is measured through the equation 9.

$$assign_{i,j} = \{(m, F_{i,j,m}) | F_{i,j,m} = pt_{i,j,m}.\, y_{i,j,m} + st_{i,j,m} \ , m \in U_{i,j}\}$$

Where

$$St_{i,j,m} = \begin{cases} RT(m) & max(F_{i,(j-1)}, F_{i',j',m}) < RT(m) \ \ if \\ max(F_{i,(j-1)}, F_{i',j',m}) & otherwise \end{cases} \quad (9)$$

$$\forall\, i = 1, \dots, n; j = 1, \dots, n_i$$

In this equation $St_{i,j,m}$ ،$F_{i,(j-1)}$ and $F_{i',j',m}$ are possible starting time on machine *m*, process completion time and the completion time of the last assigned operation to machine *m* respectively. Also, *n* is the jobs number, n_i the *i th* job operations number and *RT (m)* also the release time of machine *m* after the repair.

4- The minimum amount of $assign_{i,j}$ is measured by the equation 10.

$$(m', F_{m'ij}) = min(arg_{f_{i,j,m}}(assign_{i,j})) \quad \forall i = 1, \ldots, n; j = 1, \ldots, n_i \tag{10}$$

5- $O_{i,j}$ assigns to machine m for processing.

5. Experimental Results

In general, the results include two parts. First algorithm 1 is evaluated over different data set and finally we study the results of algorithm 2.

Section I:

In this section proposed algorithm in section 4-1 is evaluated. The required parameters for solving the problem are shown in Table 2. To evaluate the proposed algorithm and comparing with other methods, it is required to specify the arrival rate of jobs to environment. The average time of jobs arrival to environment is gained through equation 11 [9].

$$\alpha = \frac{\mu_p \mu_g}{\rho M} \tag{11}$$

Table 2. The Required Parameters for Proposed Algorithms

parameters	values
Population size	500
Number of host population	200
Number of virus population	100
Rate of PPS mutation	0.2
Rate of shift gen	0.2
Rate of gene inversion	0.2
Memory size(M)	50
Number of clone(nc)	40

In this equation, α is the average time of jobs arrival, μ_p the average time of operations process, μ_g the average number of jobs operations in job shop, ρ the job shop benefit clone and M the number of the available machines in job shop. According to this equation, if the average time of each action process and the average number of each action operations are low, the jobs arrive into system with low intervals and system is able to process the jobs more and vice versa. Furthermore, the number of the machines has a reverse proportion to the arrival time of jobs to environment. In other words, in case the number of machines is high, the job shop can process more jobs and consequently the interval of jobs arrival to environment is low and vice versa.

In addition, for assigning the due date of each job, *TWK* (Total work content) has been employed which is measured by using the equation 12.

$$D_i = A_i + k \sum_{j=1}^{n_i} p_{ij} \tag{12}$$

In this equation, A_i is the *ith* job arrival time, K the flow allowance factor, n_i the *ith* number of job operations and the process time of $o_{i,j}$. Each job that arrives the job shop is delivered to that due date by the equation 12.

- **First Experiment**

In his experiment, the set of Kacem data is used. The w_s amounts are also initialized as following:

$$w_1 = 0.2, \quad w_2 = 0.2, \quad w_3 = 0.4, \quad w_4 = 0.2$$

This set consists of 10 problems in different sizes. The results obtained from the algorithm are presented in Table 3. In this Table, the results are compared with the method proposed by Fattahi [10] in which the data including the problem size , number of jobs being arrived the job shop , the arrival time , the amount of objective function and CPU spent time for the proposed algorithm and the algorithm proposed by Mr. Fattahi are available. To compare these two algorithms, the amount of relative standard deviation has been measured by the equation 10.

$$\text{Dev} = [(C_f - C_{best})/C_f] \times 100\% \quad (13)$$

Table 3. The Comparison of Proposed Algorithm and Fattahi Method [10]

				Result of Proposed algorithm		Results of Fattahi method		
problem	*size*	Newj	Time(s)	Best objective	Cpu Times	Best Objective	Cpu Times	Dev (%)
MK01	*6×10*	*3*	*18*	*130*	*18.02*	*131*	*19.02*	*+0.7*
MK02	*6×10*	*5*	*10*	*58*	*5.76*	*59*	*7.77*	*+1.6*
MK03	*8×15*	*8*	*150*	*78.21*	*12.62*	*75.28*	*13.42*	*-3.8*
MK04	*8×15*	*8*	*50*	*230*	*6.98*	*248*	*9.89*	*+7.2*
MK05	*4×15*	*5*	*150*	*590*	*5.15*	*598*	*8.45*	*+0.1*
MK06	*15×10*	*5*	*40*	*154*	*15.25*	*152*	*15.69*	*-1.3*
MK07	*5×20*	*7*	*200*	*560*	*26.83*	*581*	*27.02*	*+3.6*
MK08	*10×20*	*6*	*400*	*625*	*15.89*	*633*	*16.31*	*+1.2*
MK09	*10×20*	*8*	*250*	*654*	*20.121*	*674*	*21.13*	*+2.9*
MK10	*15×20*	*10*	*150*	*842*	*25.84*	*873*	*27.71*	*+3.55*

In this equation, C_{best} and C_f are the best amount of fitness function which are respectively obtained by the proposed algorithm and the algorithm proposed by Fattahi. If the deviation amount is positive, it shows the desirability of the proposed algorithm results and in case of negative amount, it means that the algorithm has acted more badly in that case. Zero amount of this variable denotes that the algorithm has had the results similar to the previous methods. As this table indicates, in 8 cases the proposed algorithm has the positive standard deviation which is indicative of the better operation of the proposed algorithm.

-Second Experiment

In this experiment, the effect of job shop profit (ρ) function on the amount of jobs lateness and the number of delayed jobs has been studied. The proposed algorithm has been tested on $\rho = \{0.3, 0.5, 0.7, 0.9\}$. The results gained from the algorithm are compared with three extra creative methods *FIFO*, *SPT*, and *EDD* in pictures 11 to 13. Since in the stated rules there is only the criterion of efficiency and the amount of resistance is not discussed, so the comparison is made based on the efficiency level of algorithms. That is, the fitness function in this experiment is defined as the equation 14.

$$F = w_1 \times F_1 + w_2 \times F_2 + w_3 \times F_3 \quad (14)$$

Figure 8 shows the tardiness of jobs, number of delayed jobs and objective function for $\rho = \{0.3,0.5,0.7,0.9\}$. As shown this Figures, the proposed algorithm acts better than three other rules.

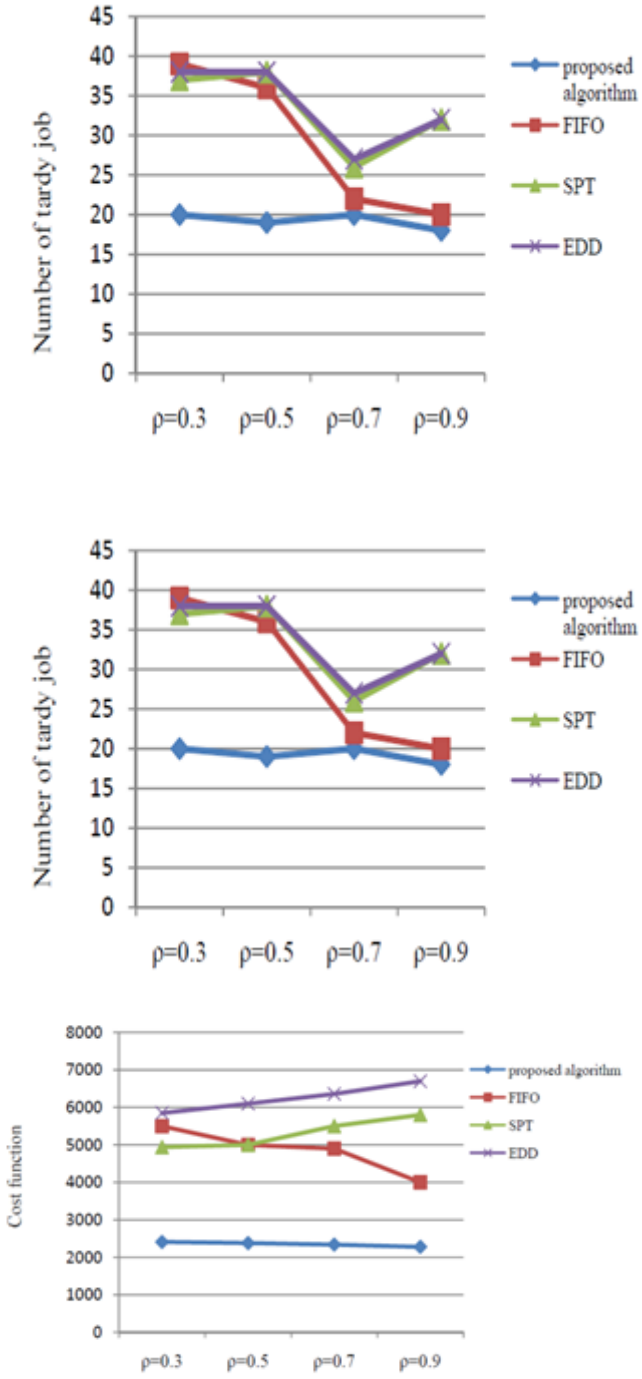

Figure 8

Section II:

This section deals with studying the experimental results of the algorithm 2 in Section 4-2. The required parameters for solving the problems are same to Table 2.

-First Experiment

This experiment has been tested on three groups of problems in sizes *Low*, *Mid*, and *High* which their characters are given in Table 4.

Table 4. Three Problem with Low, Medium and High Size

	Level 1(Low)	Level 2 (medium)	Level 3 (high)
Number_job	Uniform[10-15]	Uniform[10-20]	Uniform[20-30]
Number_Machine	Uniform[4-8]	Uniform[10-15]	Uniform[15-20]
Number_operation	Uniform[70-120]	Uniform[150-250]	Uniform[250-300]
MTTR	60	100	200
MTBR	100	150	300

In this experiments, two criterions are studied: efficiency and stability. The efficiency is the makespan of the new scheduling that is obtained by the equation 15.

$$EFF = \left\{1 - \frac{MS_{rev} - MS_{org}}{MS_{org}}\right\} \times 100 \quad (15)$$

In this equation MS_{rev} and MS_{org} are the length of new and old scheduling respectively. Also, in this experiment, the stability criterion is measured by the equation 16.

$$Dev = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}|SR_{i,j} - SO_{i,j}|}{N \times M} \quad (16)$$

In this equation, N and M are the number of the available jobs and machines in the environment in the old scheduling. $SR_{i,j}$ and $SO_{i,j}$ are the starting time of the action $O_{i,j}$ in the new and old scheduling respectively. Table 5 compared the results obtained from the proposed algorithm with the three algorithms in three levels 1, 2 and 3: *GDRS* proposed by Fahmy [11], *MAOR* proposed by Subramanian [12], and *TR* proposed by Honghong [13]. As said in the equations 12 and 13, more *EFF* and less *DEV* show the proposed algorithm is performed better. As shown this table, the proposed algorithm in comparison to *mAOR* is better in both efficiency and stability criteria. But about algorithm *TR*, proposed algorithm is better in stability criterion and worse in efficiency. Finally, it has better efficiency and stability rather than *GDRS*.

Table 5. The Comparison of New Algorithm with GDRS, MAOR and TR Algorithms

	Level 1		Level2		Level3	
	EFF (%)	DEV	EFF (%)	DEV	EFF (%)	DEV
Proposed algorithm	87.7	101	89.4	102.43	92.5	115.32
mAOR [12]	75.6	108.5	72.42	104.2	74.23	119.2
TR [13]	91.1	159.0	90.02	153.0	94.1	156.0
GDRS [11]	85.23	99.9	78.45	101.34	80.98	110.67

-Second Experiment

In this experiment, a problem 10*10 size is proposed and simulation result is tested on it. The results has compared with 3 dispatching rules: SPT, FIFD, EDD and VNS method

offered by Adibi [4]. In this experiment objective function **is** efficiency in which the weighting sum of the makspan and tardiness of jobs as following:

$$F = W_1 \times F_1 + W_2 \times F_2 \quad (17)$$

In this experiment, $w_{1\ and}$ w_2 are proposed 5 and 10 respectively. Also it is assumed that all machines have same MTTR and MTBF. To compare the algorithm with others, a factor called A_g is proposed. It is defined as the equation 15 and shows the percent of time when machines are faced with break down.

$$A\_g = MTTR / (MTTR + MTBF) \quad (18)$$

The required parameters are considered based on [4]. The other parameters are same to Table 2. This experiment is tested in three cases that are shown in Table 6. Figure 9 indicates the average objective function obtained from the algorithm that shows the proposed algorithm perform better than the other methods.

Table 6. The Required Parameters for Dynamic Scheduling

	a	*MTBF*	*MTTR*
S_1	*400*	*5700*	*300*
S_2	*300*	*5700*	*300*
S_3	*200*	*5700*	*300*

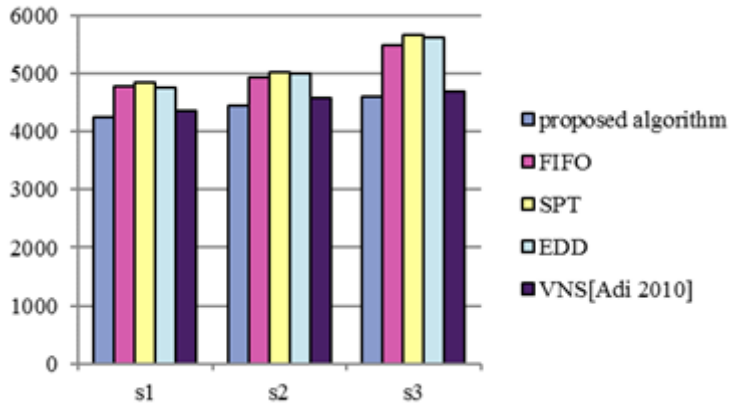

Figure 9. The Comparison of Proposed Algorithm with FIFO, SPT, EDD and VNS Methods

6. Conclusion

In This paper, multi objective *FJSSP* problem in dynamic environment was considered. To solve the problem, the combination of Artificial Immune (AIS) and virus Evolutionary Algorithm (VEA) was utilized. Two real time events were considered: Random arrival of jobs and break down machine efficiency. To face with these two events, two algorithms were offered. In these two algorithm, firstly the static scheduling of jobs and machines is done and then the static scheduling is improved by the real time events. Finally the proposed algorithms were tested on the set of different data. The experimental results have shown in the most cases the proposed algorithm has had a better performance in comparison to other algorithms.

References

[1] C. A. Holloway, R. T. Nelson, "Job Shop Scheduling with Due Dates and Variable Processing Times", Management Science, vol. 20, **(1974)**, pp. 1264–1275.

[2] J. Fang and Y. Xi, "A Rolling Horizon Job shop Rescheduling Strategy in the Dynamic Environment", International Journal of Advanced Manufacturing Technology, vol. 13, **(1997)**, pp. 227–232.
[3] G. Chryssolouris and V. Subramaniam, "Dynamic Scheduling of Manufacturing Job Shops Using Genetic Algorithms", Journal of Intelligent Manufacturing, vol. 12, **(2001)**, pp. 281–293.
[4] M. A. Adibi, M. Zandieh and M. Amiri, "Multi-objective Scheduling of dynamic Job Shop Using Variable Neighborhood Search", Expert Systems with applications, vol. 37, **(2010)**, pp. 282–287.
[5] R. Rangsaritratsamee, W. Ferrell Jr. and M. Kurz, "Dynamic Rescheduling that Simultaneously Considers Efficiency and Stability", Computers & Industrial Engineering, vol. 46, **(2004)**, pp. 1–15.
[6] Z. Davarzani and T. Akbarzadeh, "A novel Quantum Immune Algorithm for multiobjective flexible job shop scheduling", International Journal on Artificial Intelligence Tools, vol. 23, no. 5, **(2014)**.
[7] H. Z. Zhao, X. Gao and L. Zhu, "A Virus Co-evolution Genetic Algorithm Based on Niche Technology", International Conference on Information Acquisition, **(2006)**.
[8] K. M. Lee, T. Yamakawa and K. Lee, "A genetic algorithm for general machine scheduling problems", International Journal of Knowledge-Based Electronic, **(1998)**.
[9] V. Vinod and R. Sridharan, "٢٠١١: Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system", Int. J. Production Economics. P. ١٢٧-١٤٦.
[10] P. Fattahi and A. Fallahi, "Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability", CIRP Journal of Manufacturing Science and Technology, **(2009)**.
[11] S. A. Fahmy, T. Y. ElMekkawy and S. Balakrishnan, "Job shop deadlock-free scheduling using mixed integer programming and rank matrices", In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, **(2007)**, pp. 2776–2781.
[12] V. Subramaniam, A. Raheja and R. Reddy, "Reactive repair tool for job shop schedules", International Journal of Production Research, **(2005)**.
[13] Y. Honghong and W. Zhiming, "The application of adaptive genetic algorithms in FMS dynamic rescheduling", International Journal of Computer Integrated Manufacturing, **(2003)**, pp. 382–397.

