**Iraqi Journal of Statistical Sciences (23) 2013**
**PP[11-26]**

# A GENETIC ALGORITHM TO MINIMIZE THE MAKESPAN FOR THREE MACHINE FLOW SHOP SCHEDULING PROBLEMS

MANAL ABDULKAREEM ZEIDAN*

**خوارزمية جينية لتقليل وقت التنفيذ لمسائل جدولة الورشة الانسيابية ذات الثلاث مكائن**

الهدف من هذا البحث هو اقتراح خوارزمية جينية تقوم بإيجاد أفضل جدولة لـ n من الأعمال في بيئة الورشة الانسيابية بثلاث مكائن بحيث يكون وقت التنفيذ (makespan) اقل ما يمكن. إذ اقترح أسلوب جديد لعملية الانتقاء (selection)، عملية تداخل جديدة (3PLOX) وكذلك معيار توقف جديد يعتمد على الحد الأدنى (LB) لوقت التنفيذ. وكذلك اقترح أسلوب جديد لحساب وقت التنفيذ لـ ـ n من الأعمال عند معالجتها بثلاث مكائن في بيئة الورشة الانسيابية .

ولمعرفة كفاءة الخوارزمية الجينية المقترحة قورنت مع خوارزمية جوهانسن وبعد التطبيق على عدة مسائل ولدت عشوائيا من التوزيع المنتظم، أظهرت النتائج أن الخوارزمية الجينية المقترحة أفضل من خوارزمية جوهانسن بنسبة 94% في إيجاد امثل متتابعة لجدولة الإعمال والتي تعطي اقل وقت تنفيذ.

**ABSTRACT**

The aim of this paper is to propose genetic algorithm to finding the optimal schedule with minimum makespan for n jobs in flow shop environment with three machines.

In this pape, a new approach for selection, a new crossover operation (3PLOX) and a new stopping criteria based on the lower bound of the makespan (LB) are proposed.Also, a new procedure of calculating the make span for n jobs at processing by three machine in flow shop environment is suggested. In order to examine the effectiveness of the proposed GA, a comparison was made with Johnson's algorithm. After the application on several problems which generated randomly by uniform distribution, the results showed that the proposed GA is better than Johnson's algorithm with rate 94% in finding the optimal sequence for scheduling jobs which gives optimal makespan.

* Assit. Lecturer \ Department of operational research & intelligent techniques \ College of Computers Sciences and Mathematics, Mosul University.

**Received Date 25/6/2012 Accept Date 17/9/2012**

## 1- INTRODUCTION

Scheduling problem plays an important role in manufacturing and service industries . It is concerned with setting the permutation for a set of jobs on a set of machines to obtain the optimal value for a certain measure of performance [9].

The permutation flow shop problem with n jobs and m machines as studied by many researchers is commonly defined as follows, Each of n jobs is to be sequentially processed on machine 1,2,…..,m . The processing time $p_{ij}$ of job j on machine i is given. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. The objective widely used is to find a permutation of jobs to minimize the maximum completion time i.e. makespan $C_{max}$.

Flow shop scheduling problem belongs to areas of combinatorial optimization problem and it is proved to be a very complex and difficult combinatorial optimization problems. It is NP hard [15].For such NP-hard combinatorial optimization problems heuristics play a major role in searching for near-optimal solutions. Genetic algorithms are used in scheduling leading to efficient heuristic method for large sized problems. The efficiency of a GA is closely related to the quality of the used GA scheme and the GA operators: selection, crossover, mutation and stopping criteria.

In this paper, a new genetic algorithm is proposed for three machine flow shop problem with makespan $C_{max}$ as the criterion. One novel crossover operator (3PLOX) is designed, a new approach for calculation the makespan is presented and a new stopping criteria based on the lower bound of the makespan is presented.

This paper is organized as follows, In section 2 , the flow shop problem will first be described. In section 3, we present a new approach for calculation the makespan. In section 4, we show the lower bound of the makespan for flow shop scheduling problem. In section 5, the concepts of GAs will be described and in section 6, we applied it to n jobs , 3 machines with makespan ($C_{max}$) as the criterion. In section 7 we analyses the performance of the GA by compare the results with Johnsons algorithm . Finally, section 8, conclusions are discussed.

### 1.1- PREVIOUS WORKS

Johnson proposed an easy algorithm for two and three machine flow shop problem[7]. Since then, several researchers have focused on

solving flow shop problems. Etiler,O. et al [5] develop a genetic algorithm- based heuristic for flow shop scheduling problem with makespan as the criterion. Tang,J. et al [13] presented a hybrid optimization algorithm for flow shop scheduling problem, they introduce the Particle Swarm Optimization algorithm (PSO) to find better initial population. This method is validated on a series of benchmark dataset and the experimental results indicate that this method is efficient and competitive compared to some existing method. Chang,P. et al [4] presented a novel genetic algorithm for the flow shop scheduling problem by combining mutation- based local search with traditional genetic algorithm. Adusumilli ,K. et al [1] presented a genetic algorithm for the two machine flow shop problem, they propose a heuristic for approximating the solution for the $F_2//\Sigma C_i$ problem using a genetic algorithm and they calibrated the algorithm using optimal results obtained by branch and bound technique. Liao,X. [9] proposed an orthogonal genetic algorithm for no-wait flow shop problem with total flow time minimization. Aggoune,R. and portmann,M. [2] presented a heuristic approach for flow shop scheduling problem with limited machine availability where they proposed a heuristic approach to approximately solve the problem that consists in scheduling the jobs two by two according to an input sequence and using a polynomial algorithm. This algorithm is an extension of the geometric approach developed for the two- job shop scheduling problem .

## 2- PROBLEM DESCRIPTION

Flow shop problems are a distinct class of shop scheduling problems, where n jobs (j=1,2,….,n) have to be performed on m machines (i=1,2,….,m) as follows, A job consists of m operations, the $i^{th}$ operation of each job must be processed on machine i and has processing time $p_{ij}$. A job can start only on machine i if it's operation is completed on machine (i-1) and if machine i is free . All jobs have the same machine sequence and all machines have the same job sequence. The completion time of job j ,$C_j$ is the time when it's last operation has completed[1],[13]. It's easy to see that the total processing time makespan is

$$C_{max}=\max\{c_j\} \qquad j=1,2,\ldots\ldots,n$$

Figer(1) shows flow shop problem which is mentioned above with five jobs and three machines.

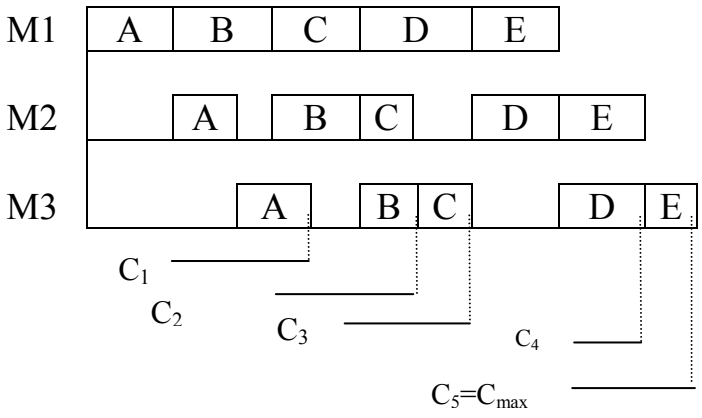



Figure (1) An example of a feasible solution of a five-job and three- machine flow shop scheduling problem

## 3- CALCULATION THE MAKESPAN

In this paper, we proposed a new procedure of calculation of makespan for N jobs at the processing by three machines in flow shop environment, the procedure depends on calculation of idle time at the second and third machines then calculates the makespan as below:-

$C_{max}$= Processing time of the first operation of the first job at the first machine + Processing time of the second operation of the first job at the second machine + Total of idle times at third machine + Total of processing time of the third operation (last) to all jobs at the third machine

We will show this procedure as the example below :-

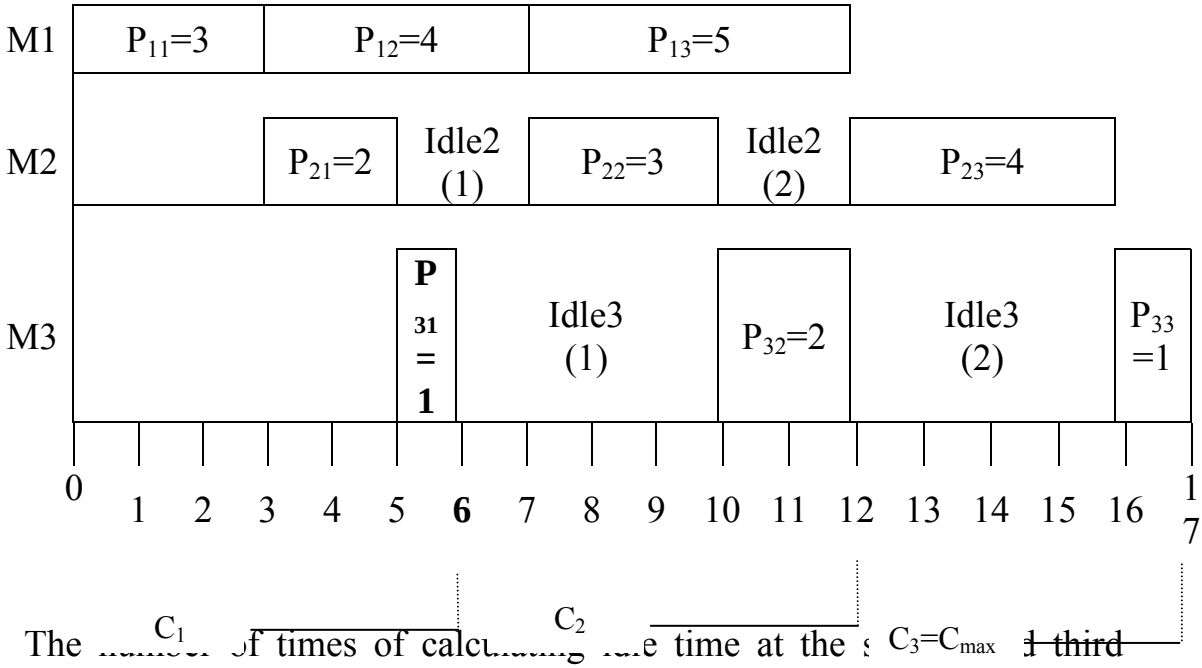
We suppose that we have 3 jobs, and we will process them with 3 machines. The processing time for the 3 jobs at the 3 machines is:

Processing time for the first job at the three machines $J_1$=[ 3 2 1 ].

Processing time for the second job at the three machines $J_2$=[ 4 3 2 ].

Processing time for the third job at the three machines $J_3$=[ 5 4 1 ].

Figure (2) shows flow shop problem which is mentioned above, This figure shows the processing of the first job firstly, then the second job then the third .




The number of times of calculating idle time at the s third machines is n-1, we will calculate the idle time for the previous example as below:

$idle_2(1) = P_{12} - P_{21} = 2$

$idle_2(2) = (P_{12} + P_{13}) - (P_{21} + P_{22} + idle_2(1)) = 2$

$idle_3(1) = (P_{21} + P_{22} + idle_2(1)) - (P_{21} + P_{31}) = 4$

$idle_3(2) = (P_{21} + P_{22} + idle_2(1) + idle_2(2)) - (P_{21} + P_{31} + P_{32} + idle_3(1)) = 4$

The makespan $C_{max}$ according to previous question will be :-

$$C_{\max} = P_{11} + P_{21} + \sum_{k=1}^{2} idle_3(k) + \sum_{j=1}^{3} P_{3j}$$

$$= 3+2+8+4=17$$

So, we programmazied " Function " by using matlab. This " Function " will calculate the makespan for N job at the processing by three machines, then the Function's inputs are the number of jobs and matrix of processing time for jobs at the three machines that the row represents the machines and the columns represent the jobs as in the following steps:-

1- Define the $idle_2$ as empty matrix.
2- Calculate $idle_2(1)=P_{12}-P_{21}$
3- Define variable X, such that $X=P_{12}+P_{13}$
4- Define variable XX, such that $XX=P_{21}+P_{22}$
5- For k=2 to n-1
   5.1 If $idle_2(k-1)>=0$ Then $XXX=XX+idle_2(k-1)$;
   ELSE XXX=XX;
   END IF
   5.2 $Idle_2(k)=X-XXX$;
   5.3 IF k=n-1 Then break
   Else
   $X=X+P_{1,k+2}$
   $XX=XXX+P_{2,k+1}$
   END IF
   END FOR
6- Define the $idle_3$ as empty matrix.
7- Define a variable $Y=P_{21}+P_{22}$
8- Define a variable $YY=P_{21}+P_{31}$
9- For r=1 to n-1
   9.1 If $idle_2(r) >=0$, then $YYY=Y+idle_2(r)$
   ELSE YYY=Y;
   END IF
   9.2 IF r=n-1 then break.
   END IF
   9.3 $Y=YYY+P_{2,r+2}$
   9.4 IF $idle_3(r)>=0$ then $YYYY=YY+idle_3(r)$
   ELSE
   YYYY=YY;
   END IF
   9.5 $YY=YYYY+P_{3,r+1}$
   END FOR
10- We can define the variable Z as the total of idle time at the third machine.
11- We can define the variable S as the total of processing time for the third operation (last) for all jobs at the third machine.
12- $C_{max}=P_{11}+P_{21}+Z+S$;

## 4- LOWER BOUND

In most of NP-hard problems there is a difficulty to evaluate the performance of the algorithm that solves this kind of problems, flow shop

problem is one of this problems [6],[8].The lower bound is an effective tool in estimating the optimal makespan for this problem when the solution of this problem is unknown, Also the lower bound is used in finding the optimal solution to small and medium size problems[10]. The lower bound of the makespan for flow shop problem is as presented below:

Let $b_i$ be the minimum amount of time before machine i starts to work and $a_i$ be the minimum amount of time that it remains inactive after it's work up to the end of the operations, and Let $T_i$ be it's total processing time we have:

$$b_i = \min_j \left( \sum_{k=1}^{i-1} P_{kj} \right)$$

$$a_i = \min_j \left( \sum_{k=i+1}^{m} P_{kj} \right)$$

$$T_i = \sum_{j=1}^{n} P_{ij}$$

Clearly, the optimal makespan $C^*_{\max}$ is greater than or equal to:

$LB = \max_i (b_i + T_i + a_i) \le C^*_{\max}$ [12].

## 5- GENETIC ALGORITHM

Genetic algorithm is a random search algorithm for simulation of the process of biological evolution [15]. In order to apply GA to a particular problem, the first step is to convert the feasible solution of that problem into a string type structure called chromosome [4].In order to find the optimal solution of a problem, a standard GA will perform the following steps:-

- Random generation of the initial population.
- Selection.
- Reproduction (crossover and mutation ).
- Replacement of the current population.

The first step consists of generating the initial population. This is done randomly. Then a selection is performed. The most adapted individuals are selected for reproduction. The best individuals are always selected. This might lead to a premature convergence to a local

optimum. Crossover is then applied, with a certain probability, to selected individuals. This operator combines two solutions to produce two new ones. Mutation, an operator not as important as the crossover, is applied to individuals at a defined rate. The role of this operator is to change the characteristics of solution. As a matter of fact, it tries to diversify the population by introducing new solution in it. From the current population and the generated one(after crossover and mutation), a set of individuals has to be chosen to form the new generation. This population will enclose the best individuals (solutions). After few generations, the GA tend to converge rapidly to the optimal solution[3].

## 6- GENETIC ALGORITHM FOR FLOW SHOP SCHEDULING

When applying GAs to a scheduling problem there is an obvious practical difficulty, we need a different string representation and genetic operators. These are shown below in detail.

### Step1: Ecoding [4],[5],[13],[15]

The traditional representation of GA which contains 0's and 1's does not work for scheduling problems. In order to apply any GA to the flow shop scheduling problem, each gene presents a job number and a chromosome presents the processing sequence of all jobs.

For a 5 jobs m machines problem, the chromosome representation example in Fig.(3) denotes that the job processing sequence on each machine is job2, job1, job3,job5,job4.

| 2 | 1 | 3 | 5 | 4 |
|---|---|---|---|---|

**Figure(3), An example of chromosome representation**

### Step 2: Generate the initial population

Randomly arrange the jobs to each chromosome. These chromosomes will generate the initial population. In this GA, we use population size equal to 100.

### Step 3: Compute the fitness Function

There are different criteria used as fitness function for the flow shop scheduling problem. The most popular of these are makespan (maximum completion time ) and total flow time. We use the makespan criterion in our GA $f(s) = C_{max}(s)$.

Where S is the chromosome, $C_{max(s)}$ is the makespan of the chromosome S and we compute it as in paragraph(3), and f(s) is the fitness function of chromosome S .

**Step 4: Selection**

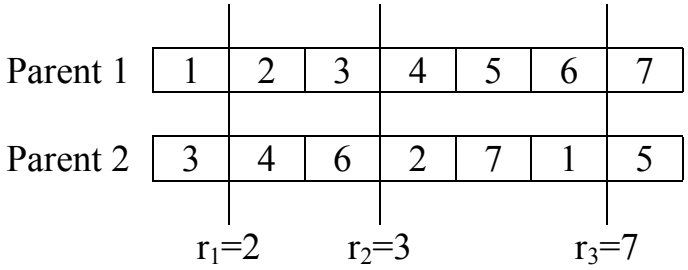
In this proposed GA we used a new procedure of selection. This procedure gives a chance to individuals which have low fitness in order to select them. After computed fitness function to all population individuals, we will arrange the individuals according to fitness function increasingly then we will choose the first parent pair of high fitness ($C_{max}$(low)) for making crossover operation upon them. Then we will choose the second parent pair which has high fitness but is already less than the first parent pair and so over until we reach the last parent pair which has the lowest fitness.

**Step 5: crossover**

The crossover operator has been considered to be the central component of GA and makes GA distinctively different from other problem solvers, by using this operator a pair of solutions (parents) generates new solution ( offsprings) by mutually exchanging and recombining information[3].

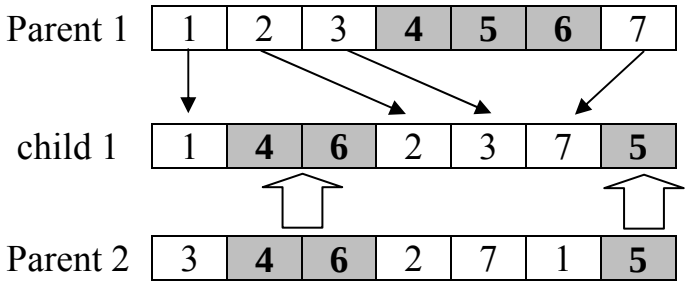
We proposed in this GA a new crossover operation ( Three – point linear order crossover (3PLOX)). This crossover operation, based on Three random cut points at each parent structure. The procedure of the three-point linear order crossover operator is as follows:

1. Choose, Three random cut points at each parent structure, we can see in example (3) thatthe cut points are at the second, third and seventh position of the parent structures.

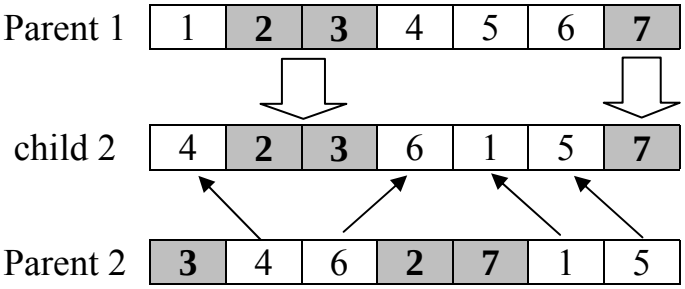



2. Child1 will take the elements between $r_1$ to $r_2$ and $r_3$ to the last position from parent2 structure and put them in the same order (position), while the remaining position in the child1 will be

filled up by the elements from parent1 one by one without repetition to the elements which are taken from parent2.




3. Child2 will take the elements between $r_1$ to $r_2$ and $r_3$ to the last position from parent1 structure and put them in the same order (position), while the remaining position in the child2 will be filled up by the elements from parent2 one by one without repetition to the elements which are taken from parent1.




**Step 6: Mutation**

In this algorithm we use exchange mutation. Exchange mutation was a simple exchange of two elements of the structure, chosen at random[5].

Before mutation: 2 **4** 1 6 5 **7** 3

After mutation :2 **7** 1 6 5 **4** 3

We apply the exchange mutation to the children with probability 0.001.

**Step 7: Replacement**

The new population generated by the previous steps up dates the old population.

**Step 8: Stopping criteria**

Most of the GAs suggested by the researchers are based on the number of generations as the stopping criteria[15]. But in fact if the number of generations is low then the probability of finding the best result is low too. Otherwise if the number of generations is too high, the iteration time is too long [5]. Therefore, we suggest a new stopping criteria Q where Q is a proportion of lower bound of the makespan (LB) to $C_{max}$ , because usually the optimal makespan is unknown but we know that $0 < LB \le C^*_{\max}$, where $C^*_{\max}$ is the optimal makespan [14].

We calculate the proportion Q as follows:-

$$Q = \frac{LB}{C_{\max}(s)}$$

where $C_{max}(s)$ is the minimum makespan in the population for chromosome (s), LB is the lower bound of the makespan and we compute it as in paragraph 4.

Clearly, we can see that $0 < Q \le 1$. The aim is to find out the optimal makespan $C^*_{\max}$ by trying out to get Q equal to 1, if we get it before the completing of 100 generation, GA will stop! In this case $C_{max}$ will be optimal $C^*_{\max}$ , because $C_{\max} = LB$ .

Otherwise, we should reduce the proportion 1 by 0.01 in order to get Q equal to 0.99 to find the makespan $C_{max}$ where its deviation about LB is minimum. In case of 100 generation is completed and we couldn't get Q equal to 0.99, we reduce the proportion 0.99 by 0.01 in order to get Q equal to 0.98 and continue with the same steps until we found $C_{max}$ where it's deviation about LB is minimum.

## 7- NUMERICAL RESULTS

In order to examine the effectiveness of the proposed GA, a comparison was made over a wide range of jobs with the Johnson's algorithm. We use Johnson's algorithm because it gives optimal solution for the problems with two and three machines [7]. Five different problems were examined, they correspond to the different job number 10,25,50,75 and 100, respectively.For each problem, 10 tests are generated by set different processing time randomly and at the same interval, so all together 50 tests are performed.

The job processing times $P_{ij}$ of job j on machine i $(1 \le i \le m, 1 \le j \le n)$ are uniformly distributed integers in the range between 1 and 40, we have generated the values of $P_{ij}$ in the following way[11],[12].

For i=1 to m

For j=1 to n

$P_{ij}$=U[1,40]

The results of the comparison are presented in the following tables.

Table (1) has seven columns, the first column shows the number of jobs n. The second column shows the generated problems. The third column shows the lower bound of the makespan as computed in paragraph 4. The fourth column shows the makespan $C_{max}$ as computed by proposed GA. The fifth column shows the makespan $C_{max}$ as computed by the Johnson's algorithm. The sixth column shows the average percentage deviation for the $C_{max}$ as computed by the GA. The seventh column shows the average percentage deviation for the $C_{max}$ as computed by the Johnson's algorithm.

Table (2) has seven columns, the first column shows the number of jobs n. The second column shows the number of generated problems. The third and fifth column illustrate the number of times of best solution obtained by the proposed GA and Johnson's algorithm, respectively. The fourth column shows the number of times that two algorithms in a comparison give the same makespan. The last two columns show the percentage of success of each algorithms, The total number of times that the algorithm gives the best solution (number of advantage + number of even) divided by the number of generated problem. ( the method of Etiler,O.,Toklu,B.,Atak,M. and Wilson,J.[5]).

## Table(1) Results of all problems

| No. of jobs n | Generated problems | Lower bound LB | $C_{max}$ by GA | $C_{max}$ by Johnson | $Er = \frac{C_{max} - LB}{LB} * 100$ by GA | $Er = \frac{C_{max} - LB}{LB} * 100$ by Johnson |
|---|---|---|---|---|---|---|
| 10 | Prob.1,1 | 273 | 273 | 278 | 0.00% | 1.83% |
| | Prob.1,2 | 280 | 280 | 295 | 0.00% | 5.36% |
| | Prob.1,3 | 210 | 216 | 252 | 2.85% | 20.00% |
| | **Prob.1,4** | **271** | **280** | **271** | **3.32%** | **0.00%** |
| | Prob.1,5 | 260 | 277 | 271 | 4.23% | 4.23% |
| | Prob.1,6 | 252 | 254 | 274 | 0.79% | 8.73% |
| | Prob.1,7 | 289 | 301 | 306 | 4.15% | 5.88% |
| | Prob.1,8 | 263 | 264 | 307 | 0.38% | 16.73% |
| | Prob.1,9 | 296 | 296 | 318 | 0.00% | 7.43% |
| | Prob.1,10 | 247 | 257 | 257 | 4.04% | 4.05% |
| 25 | Prob.2,1 | 561 | 561 | 568 | 0.00% | 1.25% |
| | Prob.2,2 | 637 | 637 | 653 | 0.00% | 2.51% |
| | **Prob.2,3** | **586** | **586** | **586** | **0.00%** | **0.00%** |
| | Prob.2,4 | 594 | 594 | 614 | 0.00% | 3.37% |
| | Prob.2,5 | 551 | 551 | 552 | 0.00% | 0.18% |
| | Prob.2,6 | 597 | 597 | 599 | 0.00% | 0.34% |
| | Prob.2,7 | 586 | 586 | 597 | 0.00% | 1.88% |
| | Prob.2,8 | 601 | 606 | 628 | 0.83% | 4.49% |
| | Prob.2,9 | 597 | 597 | 637 | 0.00% | 6.70% |
| | **Prob.2,10** | **664** | **664** | **664** | **0.00%** | **0.00%** |
| 50 | Prob.3,1 | 1102 | 1116 | 1141 | 1.27% | 3.54% |
| | **Prob.3,2** | **1140** | **1140** | **1140** | **0.00%** | **0.00%** |
| | Prob.3,3 | 1045 | 1045 | 1078 | 0.00% | 3.16% |
| | Prob.3,4 | 1251 | 1256 | 1264 | 0.39% | 1.04% |
| | Prob.3,5 | 1041 | 1041 | 1069 | 0.00% | 2.69% |
| | Prob.3,6 | 1083 | 1083 | 1092 | 0.00% | 0.83% |
| | Prob.3,7 | 1096 | 1113 | 1141 | 1.55% | 4.11% |
| | Prob.3,8 | 1058 | 1061 | 1093 | 0.28% | 3.31% |
| | Prob.3,9 | 1154 | 1154 | 1200 | 0.00% | 3.99% |
| | **Prob.3,10** | **1030** | **1030** | **1030** | **0.00%** | **0.00%** |

| No. of jobs n | Generated problems | Lower bound LB | $C_{max}$ by GA | $C_{max}$ by Johnson | $Er = \frac{C_{max} - LB}{LB} * 100$ by GA | $Er = \frac{C_{max} - LB}{LB} * 100$ by Johnson |
|---|---|---|---|---|---|---|
| 75 | **Prob.4,1** | **1681** | **1681** | **1681** | **0.00%** | **0.00%** |
| | **Prob.4,2** | **1566** | **1571** | **1575** | **0.31%** | **0.00%** |
| | Prob.4,3 | 1591 | 1591 | 1648 | 0.00% | 0.57% |
| | **Prob.4,4** | **1478** | **1482** | **1478** | **0.27%** | **0.00%** |
| | Prob.4,5 | 1619 | 1633 | 1668 | 0.86% | 3.02% |
| | Prob.4,6 | 1845 | 1845 | 1890 | 0.00% | 2.43% |
| | Prob.4,7 | 1576 | 1576 | 1616 | 0.00% | 2.53% |
| | Prob.4,8 | 1686 | 1686 | 1693 | 0.00% | 0.41% |
| | Prob.4,9 | 1613 | 1613 | 1661 | 0.00% | 2.97% |
| | Prob.4,10 | 1751 | 1759 | 1759 | 0.45% | 0.45% |
| 100 | Prob.5,1 | 2045 | 2045 | 2078 | 0.00% | 1.61% |
| | Prob.5,2 | 2189 | 2189 | 2202 | 0.00% | 0.59% |
| | Prob.5,3 | 2113 | 2117 | 2204 | 0.18% | 0.68% |
| | **Prob.5,4** | **2258** | **2258** | **2258** | **0.00%** | **0.00%** |
| | Prob.5,5 | 2174 | 2179 | 2262 | 0.22% | 4.04% |
| | Prob.5,6 | 2112 | 2133 | 2205 | 0.99% | 4.40% |
| | Prob.5,7 | 2083 | 2102 | 2119 | 0.91% | 1.72% |
| | Prob.5,8 | 2069 | 2078 | 2106 | 0.43% | 1.78% |
| | Prob.5,9 | 2187 | 2198 | 2259 | 0.50% | 3.29% |
| | **Prob.5,10** | **2174** | **2174** | **2174** | **0.00%** | **0.00%** |

**Table(2) comparison of proposed GA with Johansson's algorithm**

| No. of jobs n | Generated problems | Advantage GA | Even | Advantage Johnson | GA% | Johnson% |
|---|---|---|---|---|---|---|
| 10 | 10 | 7 | 1 | 2 | 80 | 30 |
| 25 | 10 | 8 | 2 | 0 | 100 | 20 |
| 50 | 10 | 8 | 2 | 0 | 100 | 20 |
| 75 | 10 | 7 | 2 | 1 | 80 | 30 |
| 100 | 10 | 8 | 2 | 0 | 100 | 20 |
| **total** | **50** | **38** | **9** | **3** | **94** | **24** |

According to the results in Table (1), the proposed GA obviously yields better sequence for scheduling jobs because this sequence gives the optimal makespan which is equal to the lower bound as shown in shadowy rows in the table where the average percentage deviation from LB is 0%.

While there are only a few states the Johnson's algorithms yields better sequence for scheduling jobs, where this sequence gives optimal makespan which is equal to lower bound, as shown in bold font rows in the table where the average percentage deviation from LB is 0%.

Also we notice in table (1), even if the two algorithms can't reach the optimal makespan, the sequence of scheduling jobs which choosed

by proposed GA is better than Johnson, where the average percentage deviation of makespan given by proposed GA is less than by Johnson.

Table( 2) shows that in the 50 generated problems, the Johnson's algorithm gets better results than the proposed GA only 3 times out of 50, while the proposed GA is 38 times better out of 50. The two algorithms give the same results 9 times out of 50 .

## 8- CONCLUSION

1. According to the numerical results, the proposed GA success rate is 94% (Table (2)).
2. The proposed GA is better than Johnson's algorithm in finding the optimal sequence for scheduling jobs which has optimal makespan $C^*_{\max}$ where $C^*_{\max} = LB$ , if proposed GA couldn't achieve the mentioned above, then it will find out the best sequence for scheduling jobs which has best makespan because of it's average percentage deviation from LB is minimum.

## references


1. Adusumilli,K.,Bein,D. and Bien,W.,(2008),"A Genetic Algorithm for two machine flow shop problem", **IEEE, proceedings of the 41 st Hawaii International conference on system sciences**.
2. Aggoune,R. and Portmann, M.,(2006),"flow shop scheduling problem with limited machine availability: A heuristic approach", **International journal of production economics**, Vol.99,PP. 4-15.
3. Benbouzid-sitayeb,F.,Varnier,C. and Zerhouni,N., (2006), "proposition of new Genetic operator for solving joint production and maintenance scheduling: Application to the flow shop problem",**IEEE, International conference on service systems and service management**, Vol.1,PP. 607-613.
4. Chang,P.,Liu,C. and Fan,C.,(2006),"A Depth- first mutation-based Genetic Algorithm for flow shop scheduling problems", **IEEE, International conference on hybrid Information technology (ICHIT'06).**
5. Etiler,O.,Toklu,B.,Atak,M. and Wilson,J.,(2004),"A Genetic Algorithm for flow shop scheduling problems", **the Journal of the operational research society**, Vol.55,No.8,PP.830-835.

6. Jatinder,N. and Gupta,D.,(1988),"two-stage hybrid flow shop scheduling problem", **Journal of the operational research society**, Vol.39, No.4,PP. 359-364.
7. Johnson,S.M.,(1954),"optimal two and Three stage production schedules with set-up times included", **Naval Research logistics Quaterly**, Vol.1, PP. 61-68.
8. Kis,T. and Pesch, E.,(2005)," A review of exact solution methods for the non-preemptive multiprocessor flow shop problem", **European Journal of operational research**, Vol.164,PP. 592-608.
9. Liao,X.,(2009),"An orthogonal genetic algorithm with total flow time minimization for the no-wait flow shop problem", **IEEE, proceedings of the eighth International Conference on machine Learning and Cybernetics, Baoding**, 12-15 July 2009.
10. Oğuz,C. and Fikret Ercan, M.,(2005),"A genetic algorithm for hybrid flow shop scheduling with multiprocessor tasks", **Journal of scheduling**, Vol.8,PP.323-351.
11. Oğuz,C. ,Fikret Ercan, M., Edwin chang,T.C. and Fung, Y.F.,(2003)," Heuristic algorithms for multiprocessor task scheduling in two-stage hybrid flow shop ",**European Journal of operational research**, Vol.149,PP. 309-403.
12. Taillard, E.,(1993)," Benchmarks for basic scheduling problems " **European Journal of operational research**, Vol.64,No.4,PP.278-285.
13. Tang,J., Zhang,G. and Zhang,B.,(2010)," hybrid Genetic Algorithm for flow shop scheduling problem ", **IEEE, International conference on Intelligent computation Technology and Automation.**
14. Xiao,W.,Hao,P.zhang,S. and Xu,X.,(2000)," Hybrid flow shop scheduling using genetic algorithms " , **IEEE, proceeding of the 3rd world congress on intelligent control and automation**, June 28- July 2, Hefei, P.R. china.
15. Zhang,S.,(2010),"Large-Scale flow shop scheduling based on genetic algorithm ", **IEEE, 2010 2nd International conference on Education Technology and computer(ICETC).**