International Journal of Artificial Intelligence and Applications for Smart Devices
Vol.2 , No.1 (2014), pp. 57-72
http://dx.doi.org/10.14257/ijaiasd.2014.2.1.05

A Novel Quantum Immune Algorithm for Multiobjective Flexible Job Shop Scheduling

ZohrehDavarzani and Mohammad-R Akbarzadeh-T

Department computer engineering, FerdowsiUniversity of Mashhad, Mashhad, Iran
Z_davarzani@yahoo.com, akbarzadeh@ieee.org

Abstract

In this study, a hybrid of Quantum Evolutionary and Artificial Immune Algorithms (QIA) is proposed for solving Multiobjective Flexible Job Shop Scheduling Problem (MFJSSP). This problem is formulated as a three-objective problem which minimizes completion time (makespan), critical machine workload and total work load of all machines. The quantum coding is shown to improve the immune strategy. The proposed algorithm overcomes the problem by increasing the speed of convergence and diversity of population. Three benchmarks of Kacem and Brandimart are examined to evaluate the performance of the proposed algorithm. The experimental results show a better performance in comparison to other approaches.

Keywords: *Flexible Job Shop Scheduling, Immune Algorithm (IA), Quantum Evolutionary Algorithm*

1. Introduction

Task scheduling can be defined as the assignment of resources to tasks so that a set of predefined performance measures are optimized. In production scheduling, resources and tasks are commonly referred to as machines and jobs, and the performance measures are the completion times of jobs, workload of the critical machine and total workload of all machines. One of the most common scheduling problems is job shop scheduling problem (JSSP), where a set of independent jobs must be processed on a set of available machines. JSSP is one of the important combinatorial optimization problems because it is used in most planning and managing of manufacturing processes. Each job is a sequence of operations, each operation requiring a predefined machine. Moreover, machines are available in time zero and can process just one operation at a time without interruption. The problem is how to sequence the operations on the machines (sequencing) so that a predefined performance measure is optimized.

The Flexible Job-Shop Scheduling problem (FJSSP) is a special kind of classical JSSP, where operations are required to be processed only on a subset of the available machines. Thus, FJSSP is more difficult than the classical JSSP because it has routing problem in addition to sequencing problem. Routing problem is to select a suitable machine from among the available ones to process each operation. This problem has been demonstrated to be NP-hard.So far, no exact approach has been introduced to solve FJSSP within a reasonable time.

To date, many approaches have been used to solve FJSSP, such as Tabu Search (TS)[1,4,5], Immune Algorithm (IA)[7-10,13,20], Branch-and-Bound (B&B), Genetic Aalgorithm (GA)[19,22], Simulated Annealing (SA) and hybrid of these method[12,17].These available methods can be classified into two main categories of: hierarchical approaches and integrated approaches.

ISSN: 2288-6710 IJAIASD

In hierarchical methods, the sequencing of operations on the machines and routing are treated separately. Hierarchical methods decompose the original problem to several sub-problems in order to reduce its complexity. In 1993, Brandimarte [1] first used this approach for the FJSSP. He solved the routing problem using dispatching rules and then solved the sequencing problem by TS heuristics. Kacem*et al.*, [18]presented a GA to optimize the assigned model which was generated by localization approach localization. Xia and Wu [3] proposed a hybrid algorithm for the multi-objective FJSSP. They used Particle Swarm Optimization (PSO) for the routing problem on the machines and SA algorithm for the sequencing problem.

In contrast integrated approaches solve sequencing and routing problems simultaneously. They are shown to reach better results than hierarchical methods, but are more difficult to solve. In 1994, Hurink*et al.*, [4] presented a TS algorithm in which reassignment and rescheduling are considered simultaneously. Dauzere-Peres and Paulli proposed a TS heuristic based on neighborhood structure solving problem [5]. In 2002, Mastrolilli and Gambardella [6] improved Dauzere-Peres' TS approaches and proposed two neighborhood functions.

Among the above approaches, Artificial Immune Algorithm (AIA) is a well-known meta-heuristic which have been used for many optimization problems. Many Immune Algorithms have been proposed for solving scheduling problems [7- 9, 13, 20].Ong *et al.*, [10] proposed an IA called ClonaFLEX which was based on the Clonal selection principle, to solve FJSP.

The immune strategy depends on heavy use of the mutation operator in reproduction stage thereby reaching globally optimal solutions more consistently. In contrast, Quantum Evolutionary Algorithm (QEA) depends on quantum representation to maintain population diversity. QEA was first proposed by Han and Kim [11] in 2002 as a new Evolutionary Algorithm (EA). It utilized the concept of superposition states. The main operator for updating Q-bits is quantum rotation. This operator could guide the search direction toward the best position thereby increasing the convergence rate of algorithm. Han and Kim *et al.*, [11] proposed a QEA for the 0–1 knapsack problem. Jinweia*et al.*, [12] proposed a hybrid of QEA and GA (QGA) for solving the deterministic flow shop scheduling problem.

This paper presents a hybrid of Quantum Evolutionary and Artificial Immune Algorithms (QIA) based on a hierarchical method to solve a stochastic multiobjectiveFJSSP. This problem is formulated as a three-objective problem which minimizes completion time (makespan), workload of critical machine (machine with maximum load) and total work load of all machines. The proposed algorithm uses a novel heuristic to assign operations to machines and QIA to sequence operations. The quantum coding is shown to improve the immune strategy. The proposed algorithm overcomes the problem of IA by increasing the speed of convergence and diversity of population.

The remainder of this paper is as follows. Definition and formulation of problem are described in Section 2. Quantum evolutionary and immune algorithms are briefly reviewed in Sections 3 and 4 respectively. The proposed algorithm is presented in Section 5. Numerical results are given in Section 6, and finally Section 7 includes conclusions and future works.

2.Flexible Job Shop Scheduling Problem

The problem is to execute $N= \{J_1 \ldots J_N\}$ jobs on $U= \{M_1,\ldots,M_m\}$ machines. Each job $J_i$is a set of $n_i$operations $\{O_{i,1}\ldots O_{i,ni}\}$. Each operation$O_{i,j}$can be processed on any subset $U_{i,j} \subseteq U$ of available machines. Each job is completed when its operations are executed one by one in a given sequence.

FJSSP consists of the following two sub-problems that need to be solved simultaneously. The first subproblem is the routing problem which is defined as determining the suitable machine from among the available machines for each operation, and the second problem is the problem of sequencing where the sequence of the assignment of operations to machines is determined over a required time span.

We wish to solve these sub-problems simultaneously in order to achieve the objectives which are minimizing makespan (i.e., the maximum job completion time), workload of critical machine (the machine with the highest workload) and workload of all machines.

FJSSP is classified into two sub-problems of: Partial FJSSP (P-FJSSP), and total FJSSP (T-FJSSP). We have partial flexibility if there exists a proper subset $U_{i,j}$of U, for at least one operation $O_{i,j}$, while we have $U_{i,j}=U$ for each operation $O_{i,j}$in the case of total flexibility [22].

Assumptions of this paper are as follows [4]:

1. Every machine processes only one operation at a time.
2. Machines are independent from each other.
3. Jobs are independent from each other.
4. There are no precedence constraints among the operations of different jobs.
5. Every operation is processed on only one machine at a time.
6. All jobs and their operations are available initially.

Table 1 below presents the notations of this model.

Table 1. The Notations of Conceptual Model

M	Number of machines
N	Number of jobs
$O_{i,j}$	jth operation of job$i$
$T_{i,i}$	Processing time of $O_{i,i}$
$F_{i,i}$	Finish time of $O_{i,i}$
C_i	Completion time of job$i$
n_i	Number of operation of job$i$
$y_{m,i,i}$	assigned machine to operation $O_{i,i}$
W_i	workload of machine j
W_{max}	Total workload of all machine
C_{max}	Maximum completion time of all job

According to the notations above, model of FJSSP model can be defined as follows:

$$F_{ij} \leq T_{i,j+1} i = 1, \dots, n; j = 1, \dots, n_i \quad (1)$$

$$F_{ij} \leq C_{max} \, i = 1, \dots, n; j = 1, \dots, n_i \quad (2)$$

$$y_{mij} \leq a_{mij} \, m = 1, \dots, M \quad (3)$$

$$\sum_{m=1}^{M} y_{mij} = 1 \quad (4)$$

Constraint (1) indicates precedence constraints among the operations in each job so that operations can be executed when its precedence operation is executed.Constraint (2) defines the makespan(C_{max}) and Constraint (3) indicatesthat each operation can be assigned to just one machine from among the given machines.Eq(4) shows that only one machine from the available alternatives can be assigned to each operation.

In this paper, the following criteria are to be minimized:

$$f_1 = \sum_{i=1}^{N} C_i \tag{5}$$

$$f_2 = \sum_{i=1}^{M} \mathrm{w_k} \tag{6}$$

$$f_3 = \max(W_k) \quad k = 1, \dots, M \tag{7}$$

Eqs (5-7) indicate makespan or maximal completion time of machines,total workload of the all machines and workload of critical machine.

Table 2 shows data related to a sample problem. In this table, rows and columns represent operations and machines respectively. Symbol '∞'means that a machine cannot process the corresponding operation. In other words, it does not belong to the subset of compatible machines for that operation.

Table 2. A problem with Size 3*3

Job		m_1	m_2	m_3
J_1	O_{11}	1	2	2
	O_{12}	2	2	∞
	O_{13}	3	3	2
J_2	O_{21}	4	3	2
	O_{22}	2	∞	2
	O_{23}	3	2	3
J3	O_{31}	2	2	5
	O_{32}	∞	1	3

3. Quantum Evolutionary Algorithm (QEA)

QEA is inspired from the principles of quantum computation. It uses a new representation which is based on the concept of Q-bits and superposition of states. A Q-bit is the smallest unit of information stored in a two state "0" or "1". A Q-bit can be represented as follows:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{8}$$

Where α and β are complex numbers, which denote probability amplitudes of the corresponding states,$|\alpha|^2$and$|\beta|^2$are the probabilities for the Q-bit to be found in "0" state and in the "1" state respectively and $|\alpha|^2 + |\beta|^2 = 1$.

The Q-bit representation has the advantage that it can represent a linear superposition of states. It may be in the "1" or "0" states, or in any linear superposition of them [21]. As a string of Q-bit, aQ-bit individual is defined as follows:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_l \\ \beta_1 & \beta_2 & \cdots & \beta_l \end{bmatrix} \text{where} |\alpha_i|^2 + |\beta_i|^2 = 1,\ i = 1,2, \dots, l \tag{9}$$

For example, a three Q-bit system is defined:

$$\begin{bmatrix} -1/\sqrt{2} & 1/2 & 1/3 \\ 1/\sqrt{2} & -\sqrt{3}/2 & 2\sqrt{2}/3 \end{bmatrix}$$

Then the states of the system can be described as follows:

$$\frac{-1}{6\sqrt{2}}|000\rangle - \frac{1}{6}|001\rangle + \frac{\sqrt{3}}{6\sqrt{2}}|010\rangle + \frac{\sqrt{6}}{3\sqrt{2}}|011\rangle + \frac{1}{6\sqrt{2}}|100\rangle + \frac{1}{3}|101\rangle - \frac{\sqrt{3}}{6\sqrt{2}}|110\rangle - \frac{\sqrt{3}}{3}|111\rangle$$

This means that the probabilities for states $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$ are $\frac{1}{72}, \frac{1}{36}, \frac{1}{24}, \frac{1}{6}, \frac{1}{72}, \frac{1}{9}, \frac{1}{24}, \frac{1}{24}$ respectively. As a consequence, the above three Q-bit system includes the information of 8 states.

3.1. Quantum Rotation Gate

QEA is updated by a quantum rotation gate operator. This operator changes the state of a Q-bit and finds the best solution gradually. Here, a rotation gate $U(\Delta\theta_i)$ is employed to update a Q-bit individual. The *i*-th Q-bit (α_i, β_i) is updated as follows:

$$\begin{bmatrix}\alpha_i' \\ \beta_i'\end{bmatrix} = U(\theta_i)\begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix} = \begin{bmatrix}\cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\end{bmatrix}\begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix} \quad (10)$$

Where θ_i is the rotation angle and is equal to:

$$\theta_i = s(\alpha_i, \beta_i)|\theta_i| \quad (11)$$

Where $s(\alpha_i, \beta_i)$ and $|\theta_i|$ are determined in Table 3 [12].

Table 3.Look Up Table of $\Delta\theta_i$ for FJSSP

r_i	b_i	$\mathbf{f(r) > f(b)}$	$\|\boldsymbol{\theta_i}\|$	$\mathbf{s(\alpha_i, \beta_i)}$			
				$\alpha_i\beta_i>0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i=0$
0	0	False	0.2π	-1	+1	0	1 or -1
0	0	True	0	0	0	0	0
0	1	False	0.5π	+1	-1	0	0
0	1	True	0	0	0	0	0
1	0	False	0.5π	-1	+1	1 or -1	0
1	0	True	0	0	0	0	0
1	1	False	0.2π	+1	-1	0	1 or -1
1	1	True	0	0	0	0	0

4. Artificial Immune Algorithm

The immune system is an adaptive, self organizing and distributed system. In addition, it is a complex functional system that defends the human body from foreign agents such asviruses or bacteria called pathogens. It categorizes all cells or molecules into two kinds within the body: First are those that belong to its own kind (self-cell) and the second have a foreign origin (non-self-cell) [23].

Patterns expressed on pathogens are called antigens. The immune system contains cells for recognizing them. These cells are called antibodies. The disease procedure involves the attack of an antigen and its proliferation within the human body. After the proliferation of the antigen, antibodies are randomly distributed throughout the immune system.

When a pathogen invades the human body, a number of cells that recognize pathogens proliferate. These cells can be classified into two kinds: First are effecter cells and second are memory cells. The effecter cells secrete antibodies in large numbers and the memory cells have long life spans so as to act faster and more effectively in future exposures to the same or a similar pathogen [13]. During cellular reproduction, the cells suffer somatic mutations at high rates, together with a selective force; the cells with higher affinity to the invading pathogen differentiate into memory cells. This whole process of somatic mutation plus selection is known as affinity maturation [14].

Hence, AIS has two important processes: cloning and affinity maturation. Their combination is known as the Clonal Selection Principle. This principle is used to explain how the immune system reacts to infection of antigens [2]. Moreover, this theory forms the basis for an AIS-based metaheuristic to solve optimization problems. Figure 1 shows this principle.

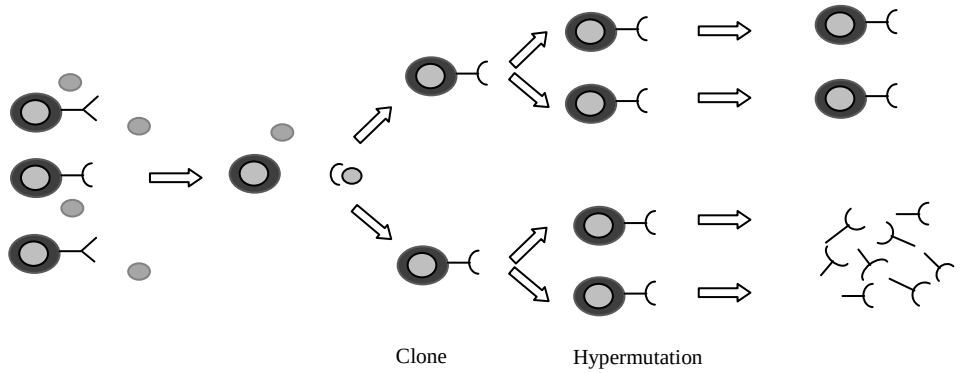

Figure 1. Clonal Selection Principle

5. Proposed Algorithm

In this study, a hybrid of Quantum and Immune Algorithms (QIA) is proposed for solving multiobjectiveFJSSP. This hybrid algorithm is based on a hierarchical method. There is a set of Q-bits strings in the population. Each Q-bits string is supposed to be an antibody and indicates a feasible solution to the optimization problem. The Q-bits strings cannot be used directly to formulate the problem. Therefore, it is necessary to map the Q-bits into the search space. The procedure to convert Q-bits strings into a feasible solution is presented in what follows. Figure 2 shows the proposed algorithm.

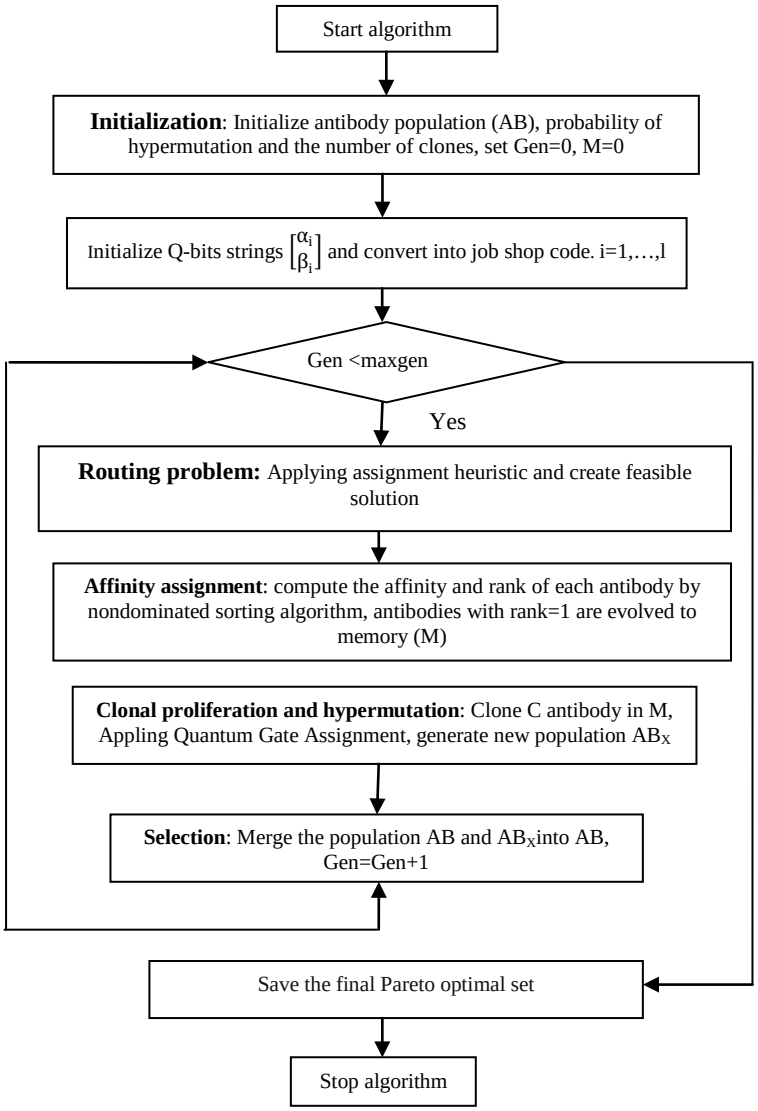

Figure 2. Flowchart of Proposed Algorithm

5.1. Representation of Q-bits

Each antibody is represented by a string of Q-bits. These Q-bits only include job sequencing information, and no job routing information is included. Each antibody represents a solution to the FJSSP. The length of each Q-bits string is $(([log_2 N] + 1) * nu_op)$where N and *nu_op* are the number of jobs and the total number of operations respectively. In the step of initialization,$\begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix}, i = 1, \dots, l$ are randomly set between 0 and 1, where$|\alpha_i|^2 + |\beta_i|^2 = 1$. Although each Q-bit can represent a linear superposition of solutions, it cannot be used directly.

5.2. Job Shop Code Converting Mechanism

Because the Q-bit string is in $\begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix}$ $i = 1, \dots, L$ form, it is needed to convert each Q-bit string into job shop code. Therefore, a converting mechanism is necessary to convert Q-bit string into the search space. This mechanism is used by Jinweiaet.al [12] described as bellows:

Step 1- **Binary String:** For each$\begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix}$, μ_i is generated between 0 and 1. Binary string $X(t)$ is set as follows:

$$X_i(t) = \begin{cases} 1 & \text{If } |\alpha_i|^2 \le \mu_i \;\; i = 1, \dots, ([\log_2 N] + 1) * nu\_op \\ 0 & otherwise \end{cases} \quad (12)$$

The length of binary string $X(t) = [x_i(t)]$ is equal to$([\log_2 N] + 1) * nu_op$.

Step 2- **Decimal Number**: Each $[\log_2 N] + 1$ bit of binary string is converted into a decimal number called$D_i(t)$. The length of the decimal string is *nu_op* where *nu_op* is the total number of operations.

Step 3- **Decimal string**: The bits in *D(t)* are ordered in increasing magnitude, thus forming a string of numbers called decimal string *M (t)*.

Steps 4 - For every bit of *M (t)*, denoted by y, "mod(y, n) +1" is calculated, where "mod" is remainder y divided by *M*. This string is called *O (t)*.

Step 5 - So far, an operation-based representation is obtained. In this representation each job must occur n_i times in the$O(t)$, where $n_i$is the number of operations of job *i*. However, the representation obtained after step 4 does not have this quality and creates an illegal solution, therefore, it is necessary to adjust the jobs in $O(t)$ by deleting the extra job number and adding the missing ones.

For example, in a 3*3 problem, the binary string *X(t)* [0 1 1| 1 0 1| 1 1 0|1 1 1|0 0 0|0 1 0|1 0 1|0 1 1] is converted from Q-bits string representation. The *D(t)* [3 5 6 7 0 2 5 3] is then taken from *X(t)* and finally *D(t)* is converted into *M(t)* [3 5 7 8 1 2 6 4].The Whole procedure is shown in Fig3.

$$\begin{bmatrix} -1/\sqrt{2} & 1/2 & -1/\sqrt{2} \\ 1/\sqrt{2} & -\sqrt{3/2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/2 \\ 1/\sqrt{2} & 1/\sqrt{2} & \sqrt{3/2} \end{bmatrix} \begin{bmatrix} \sqrt{2/3} & 1/2 & -1/\sqrt{2} \\ 1/\sqrt{3}\sqrt{3/2} & & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{4/5} & \sqrt{3}/2 & 1/\sqrt{2} \\ 1/\sqrt{5} & 1/2 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{15}/4 & 1/4 & \sqrt{4/5} \\ 1/4 & \sqrt{15}/4 & 1/\sqrt{5} \end{bmatrix} \begin{bmatrix} -1/\sqrt{2} & 1/2 & -1/\sqrt{2} \\ 1/\sqrt{2} & -\sqrt{3/2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} -1/\sqrt{2} & 1/2 & -1/\sqrt{2} \\ 1/\sqrt{2} & -\sqrt{3/2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{4/5} & \sqrt{3}/2 & 1/\sqrt{2} \\ 1/\sqrt{5} & 1/2 & 1/\sqrt{2} \end{bmatrix}$$

[0 1 1 | 1 0 1 | 1 1 0 | 1 1 1 | 0 0 0 | 0 1 0 | 1 0 1 | 0 1 1]
[3 | 5 | 6 | 7 | 0 | 2 | 5 | 3]
[3 | 5 | 7 | 8 | 1 | 2 | 6 | 4]
[1 | 3 | 2 | 3 | 2 | 3 | 1 | 2]
[1 | 3 | 2 | 3 | 2 | 1 | 1 | 2]

Figure 3. The Procedure of Converting Q-bit to Job Shop Code

5.3. Non-dominated Sorting

In order to calculate the rank of each antibody *j*, a procedure is used in which the population is sorted into different nondomination levels. This procedure is proposed by Kalyanmoy [15] that is described in Fig4. In this algorithm any antibody *i*has a *Counter* equal to zero, a *Rank*, and a *SD*. Also *Front (index)* is a set of antibodies in front index. For each antibody *i*, the *Counter (i)* is the number of solutions which dominate the antibody *i*, and *SD(i)* is a set of solutions that the solution *i* dominates.

In the first level, all antibodies have counters equal to zero. For each member *SD* of antibodies with *counter=0*, their counter is reduced by one. Then, any member with a zero counter is placed at the next front. The process is continued until all levels are obtained.

```
Input:   Antibodies population,     Number of antibodies
SD(i)=∅, Counter(i) =0   ∀ i=,…, number_antibodies
Output: Nondomination Front
        For each antibody (a)
           Compare to all antibody (b)
             If   (a dominates b)
                   SD (a)=SD (a)∪ b
             Else
             Counter (a)=Counter (a)+1
        If Counter (a) =0
          Rank(a)=1
          Front(1)=Front(1)  ∪ a
 index=1
for each antibody a in Front(index)
   for each antibody in SD (b)
     counter(b)=counter(b)-1
   if counter(b)==0
     Rank(b)= index+1
      Front(index)=Front(index) ∪ b
  index=index+1
```

Figure 4. Algorithm Fast Non Dominated Sorting

5.4. Heuristic for Operation Assignment

As describe above, the antibodies consist of job assignment information and show the order of the Q-bits string related to the priority of each operation. No job routing information is given. This information is obtained in the decoding phase. As a result, we propose a heuristic for the assignment operation to the machines. This heuristic is carried out in decoding phase and is describe as follows:

Step 1. In each antibody, for every operation $O_{i,j}$ in a given order, repeat Steps 2 and 3.

Step 2. Let $U_{i,j}$ be a set of compatible machines so that$O_{i,j}$ can be processed on it. For any $m \in U_{i,j}$ the completion time of $O_{i,j}$ on *m* is calculated as follows:

$$assign(i,j) = \{(m, F_{m,i,j}) | F_{m,i,j} = pt_{m,i,j} + st_{m,i,j} \quad \forall m \in U_{i,j}\} \quad s.t \qquad (13)$$

$$st_{m,i,j} = min\{F_{i,(j-1)}, F_{m,i',j'}\}$$

Where $F_{m,i,j}$, $pt_{m,i,j}$ and $st_{m,i,j}$ are finish time, processing time and starting time of operation $O_{i,j}$ on machine *m* respectively, $F_{m,i',j'}$ is the finish time of the last operation assigned to machine *m*.

Step 3. Let minimum value of *assign (i, j)* be$(m', F_{m',i,j})$. Tuple$(m', F_{m',i,j})$describes finish time of operation $O_{i,j}$on machinem'. Therefore $O_{i,j}$can be assigned tomachine m'.

$$(m', F_{m',i,j}) = min(assign(i,j)) \qquad (14)$$

Table4 shows an example for this heuristic. Let job sequencing be (1, 3, 2, 3, 2, 1, 1, 2), the operation assignment is shown in this Table as well. Fig 5 shows the gaunt chart of this example.

Table 4. An Example for Operation Assignment

Operation($O_{i,j}$)	predecessor	Assign(i,j)	min
1 ($O_{1,1}$)	-	{(1,1),(2,2),(3,2)}	(1,1)
3 ($O_{3,1}$)	-	{(2,3),(2,2)}	(2,2)
2 ($O_{2,1}$)	-	{(1,4),(3,2)}	(3,2)
3 ($O_{3,2}$)	$O_{3,1}$ (2,2)	{(2,3),(3,5)}	(2,3)
2 ($O_{2,2}$)	$O_{2,1}$(3,2)	{(1,4),(3,4)}	(1,4)
1 ($O_{1,2}$)	$O_{1,1}$(1,1)	{(1,6),(2,5}	(2,5)
1 ($O_{1,3}$)	$O_{1,2}$(2,5)	{(1,6),(3,7)}	(1,6)
2 ($O_{2,3}$)	$O_{2,2}$(1,4)	{(2,7),(3,7)}	(2,7)

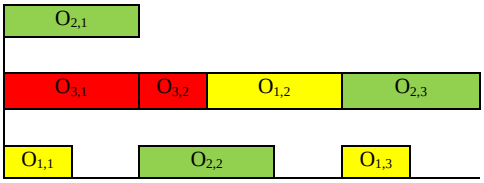

Figure 5. Gaunt Chart of this Example

5.5. Generating the Next Population

In this section, we describe how the next population is created. After applying rotation gate assignment to the antibodies, they are added to current generation. Then next generation ($G(t+1)$) is created from current generation $G(t)$ with size N. Let $Q(t)$ be a set of antibodies with size NR which rotation gate assignment is applied to them. First, a combined population $R(t) = G(t) \cup Q(t)$ is formed. Therefore $G(t+1)$ is created from $N + NR$ antibodies. Then, $R(t)$ is sorted accordance to nondomination sorting algorithm. Now, solutions belonging to the F_1 are best solutions in the $R(t)$ and are inserted to the next generation. Remainder solutions are selected from F_2. This procedure is repeated until current front has the number of solutions more than remainder solutions. This frond is called F_l. Then solutions in $F_l$are sorted using crowding distance. Remainder solutions are selected from $F_l$in descending crowding distance. This procedure is given in Figure 6 and 7.

```
Input: R(t), Number of Population(N), Number_Front(NF)
Output: next population ((P(t+1))
Count=N;
 For i=1 to Number_Front
   For  j=1 to N
     While  (number of antibody in Fi is smaller than Count)
        Move all antibodies in Fi to next generation
        Decrease Count from number of solution in Fi
      End of while
   End for
End for
If (Count is not equal to zero)
     Calculate Crowding distance of antibodies in Fl
End if
For i=1 to Count
   Insert an antibody from Fl with lower(better) crowding distance to next generation
End if
```

Figure 6. Algorithm of Selection Procedure

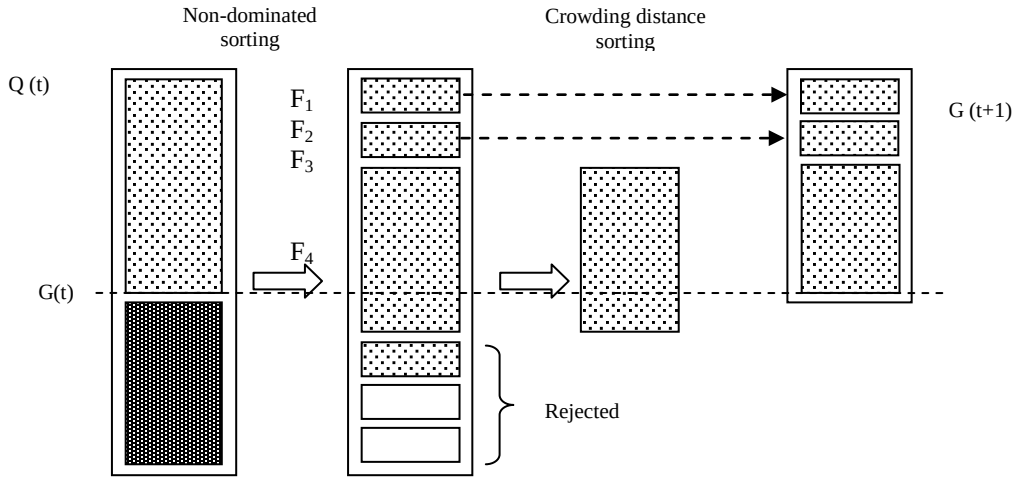

Figure 7. Generating of the Next Population Procedure

6. Experimental Result

This section describes the computational results to evaluate the efficiency of the proposed algorithm. The proposed algorithm is implemented in Matlab environment and run on a PC with 2.1 GHz. Three data sets have been considered:

1- Kacem data: The first data set is available at http://www.ec-lille.fr/ ~ kacem/testsPareto.pdf. It consists of four problems I_1 to I_4 with size 4×5, 10×7, 10×10, 15×10. All of the problems have total flexibility.

2- Kacemdata2: This data set consists of three problems with different sizes: problem 8×8, problem 10×10 and problem 15×10 from Kacem*et al.*, [16]. Problem 8×8 has partial flexibility that consists of eight jobs with 27 operations which can be processed on 8 machines, Problem 10×10 has total flexibility that consists of 10 jobs with 30 operations which can be implemented on 10 machines, and Problem 15 ×10 has total flexibility that consists of 15 jobs with 56 operations which can be performed on 10 machines.

3- BRdata: This data set consists of 10 problems from Brandimarte [1] that are randomly generated using a uniform distribution between two given limits. The number of jobs ranges from 10 to 20, the number of machines ranges from 4 to 15, the number of operations for each job ranges from 5 to 15, and the number of operations for all jobs ranges from 55 to 240.

Values of some parameters of algorithm are different for various problems as presented in Table 5.

Table 5. Parameter of QIA

Problem	Pop_size	Number_clone	Number_generation	Rate of Quantum gate assignment
8×8	300	50	300	0.4
10×10	400	100	500	0.4
15×10	500	150	500	0.4
I_1-I_4	300	100	500	0.4
MK01-MK10	500	200	1000	0.2

The first data set is Kacem data. It consists of four problems I_1 to I_4 all of which are total flexibility. Table 6indicates the best obtained solutions of 10 runs of the proposed algorithm on this data set. In this table, the problem name are given in the first column, the second column refer to the size of problem. The third, fifth and seventh columns show the values of W_{td}, W_{max} and C_{max} obtained from *"EA+FA"* of Kacem*et al.*, [18] and ninth, eleventh and thirteenth columns show the best result obtained from the proposed algorithm. In this table deviation criterion represented by *(Dev)*is employed. This criterion is defined as follows:

$$Dev = \frac{f_{ap} - f_{best}}{f_{best}} \times 100 \quad (15)$$

As shown in this table, *QIA* has better C_{max} than *"EA+FA"* of Kacemand equal to it in W_{td} and worse than it in W_{max}.

Table 6. Comparison Result with the Kacem

		EA+FA[Kac2002a]						Proposed Algorithm					
		W_{TD}	Dev%	W_{max}	Dev%	C_{max}	Dev%	W_{TD}	Dev %	W_{max}	Dev%	C_{max}	Dev%
I_1	4×5 I_1	32	3.22	7	0	16	45.45	31	0	10	42.85	11	0
I_2	7×10 I_2	60	0	9	0	15	25	60	0	11	22.22	12	0
I_3	10×10 I_3	41	0	5	0	7	0	41	0	7	40	7	0
I_4	10×15	91	0	10	0	23	64.28	91	0	12	20	14	0
			0.8		0		35.5		0		31.26		0

Table 7 shows the number of nondominated solutions for the proposed algorithm and Kacem approach on this data set. As shown in this table, the number of nondominated solutions in the proposed algorithm is more than Kacem approach.

Table 7. Number of Nondominated Solutions

	I_1	I_2	I_3	I_4
Proposed algorithm	6	9	6	7
Approach by Kacem[18]	5	1	4	3

The second data set also belongs to Kacem data [16]. Figures 8-10 show the average of the best C_{max},W_{max}, and W_{TD} obtained for 10 runs of the proposed algorithm in comparison to five algorithms. In these figures, our algorithm is compared to Temporal Decomposition [16], approach by localization and *"AL+CGA"* of Kacem*et al.*, [18, 16], *"PSO+SA"* of Xia and Wu [17], "*moGA*" of Zhang and Gen [19] and "*ClonaFLEX*" of Ong *et al.*, [20]. It is shown that the algorithm performs better than or equal to the other algorithms in the three objectives.

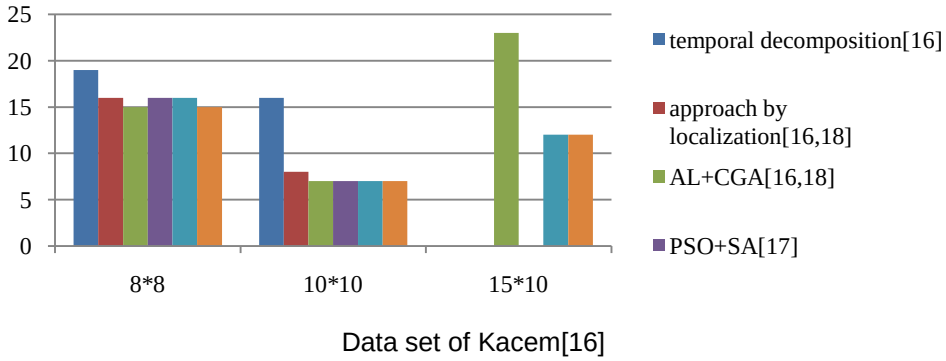

Figure 8. Average of Solutions for F_1

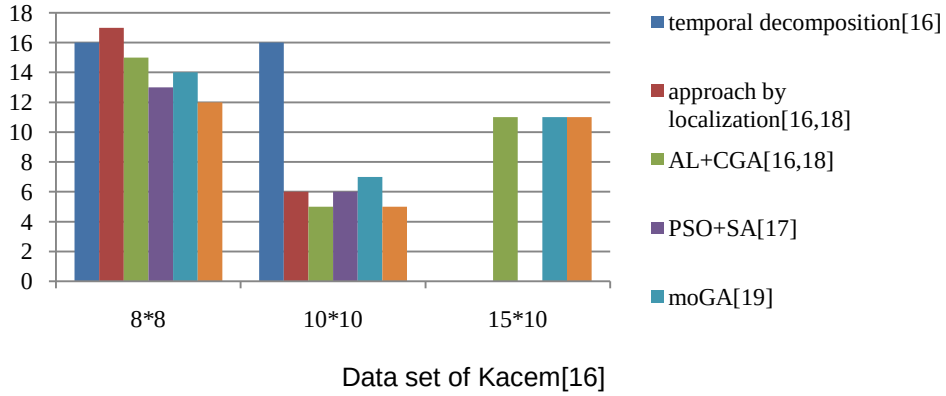

Figure 9. Average of Solutions for F_2

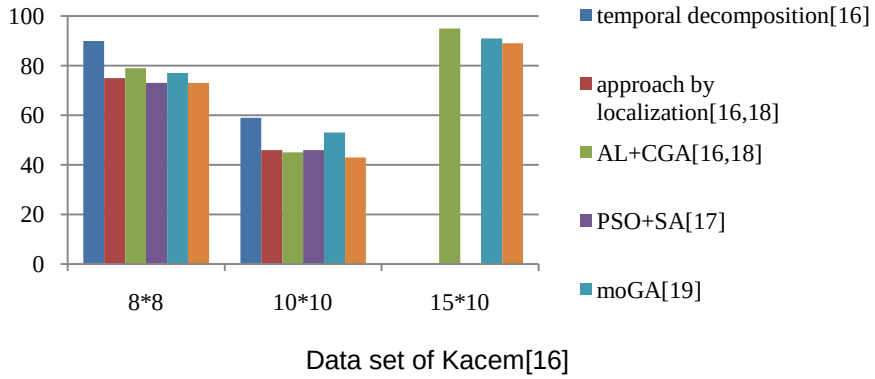

Figure 10. Average of Solutions for F_3

Also, the Gantt charts of the solutions obtained by proposed algorithm for problems 10×10 and 15×10are illustrated in Figures 11 and 12respectively.

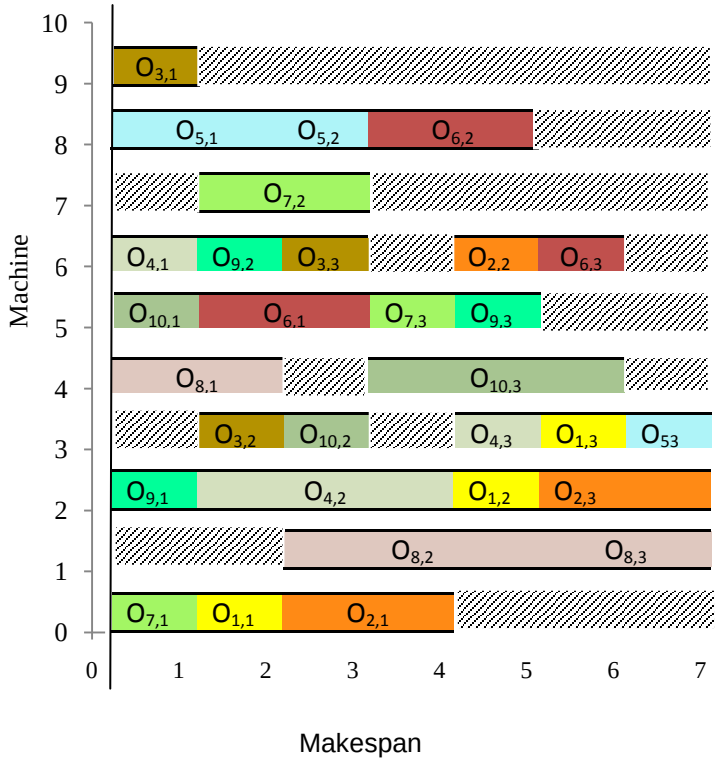

Figure 11.Gantt Chart of Solution of Problem 10 ×10

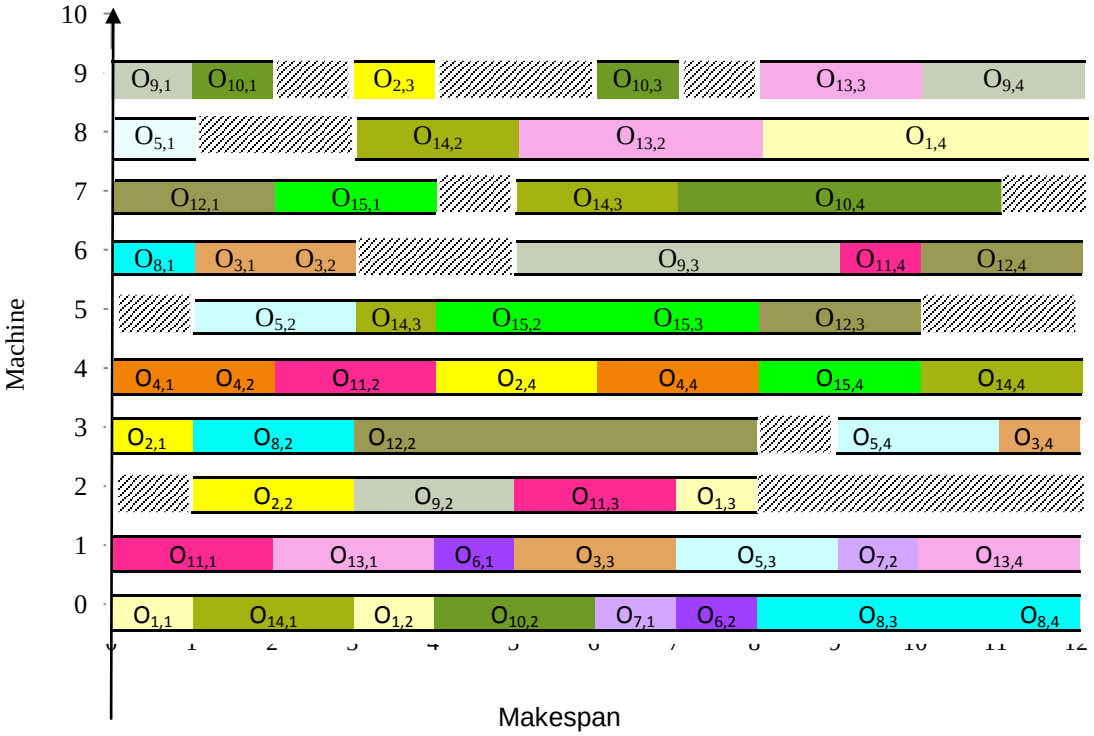

Figure 12.Gantt Chart of Solution of Problem 15×10

Other data set is BRdata [1] that are randomly generated using a uniform distribution between two given limitsby Brandimart. Table 8 indicates the best results obtained by the proposed algorithm after 10 runs on this data set. In this table, the problem names are presented in the first column and the second column shows the size of problem. C_{max}, W_{max} and W_{TD}values are respectively shown in third, fourth and fifth columns for proposed algorithm. Sixth, seventh and eighth columnsrespectively represent the best results obtained from AIA proposed by Bagheri [13]. As shown in this table, QIA has better W_{td} than AIA and worse C_{max} than it.

Table 8.Camparision Result of QIA on BRdata to AIA proposed by Bagheri [13]

problem	Size	Proposed algorithm			AIA[13]		
		C_{max}	W_{max}	W_{TD}	C_{max}	W_{max}	W_{TD}
MK01	10×6	42	42	166	40	36	171
MK02	10×6	26	26	154	26	26	154
MK03	15×8	204	204	1060	204	204	1207
MK04	15×8	74	73	365	60	60	403
MK05	15×4	173	173	686	173	173	686
MK06	10×15	65	56	457	63	56	470
MK07	20×5	143	140	683	140	140	695
MK08	20×10	533	533	2630	523	523	2723
MK09	20×10	306	305	2587	306	306	2591
MK10	20×15	214	206	2192	214	206	2121

7. Conclusion

In this study, a hybrid of quantum and immune algorithm (QIA) is proposed for solving stochastic multiobjective flexible job shop scheduling problem. The quantum representation is shown to improve the immune strategy. QIA overcomes the problem of IA by increasing the speed of convergence and diversity of population. In this algorithm, each Q-bits string indicates an antibody (solution) indirectly and is converted into job shop code. Hypermutation of each antibody is applied by quantum gate assignment. Antibodies has job assignment information and the order of the Q-bits relates to the priority of each operation. No job routing information is given. This information is obtained in decoding phase by a heuristic method. The proposed algorithm is tested on three data sets of Kacem and Brandimart. In comparison to other algorithms, the proposed algorithm is shown to be more effective.

References

[1] P. Brandimarte, “Routing and scheduling in a flexible job shop by Taboo search”, Annals of Operations Research,**(1993)**,pp. 157-183.
[2] G.L. Ada and G.J.V. Nossal, “The clonal selection theory”, Scientific American,**(1987)**,pp. 50-57.
[3] W. Xia and Z.M.Wu, “An effective hybrid optimization approach for multiobjective flexible job-shop scheduling problems”, Computers & Industrial Engineering,**(2005)**,pp. 409-425.
[4] E.Hurink,B. Jurisch and M. Thole, “Tabu Search for the Job Shop Scheduling Problem with multi-purpose machines”, Operations Research Spektrum,**(1994)**,pp. 205-21.
[5] S. Dauzere-Peres and J. Paulli, “An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using Tabu Search”, Annals of Operations Research,**(1997)**,pp. 281-306.
[6] L. Mastrolilli and L.M. Gambardella, “Effective neighborhood functions for the flexible job shop problem”, Journal of Scheduling,**(2002)**,pp. 3-20.
[7] G.Chun and C.Chueh, “A multi-modal immune algorithm for the job-shop scheduling problem”,Journal of information science,**(2009)**, pp. 1516-1531.
[8] S. Darmoul, H. Pierreval and S.H.Gabouj, “Scheduling using artificial immune system metaphors”, IEEE International Conference on Service Systems and Service Management,**(2006)**,pp. 1150-1155.
[9] E. Hart and P. Ross, “An immune system approach to scheduling in changing environments”,Proceedings of Genetic and Evolutionary Computation Conference, Orlando, Florida,**(1999)**,pp. 1559-1565.

[10] Z.X. Ong, J.C. Tay and C.K. Kwoh,"Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules",**(2005)**,pp. 42-455.
[11] K.H. Han and J.H.Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization",IEEE Transactions on Evolutionary Computation,**(2002)**.
[12] G. Jinweia,G. Manzhan,C. Cuiwena and G. Xingshenga, "A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem", Computers & Operations Research,**(2010)**,pp. 927-937.
[13] M. Bagheri, B. Zandieh, I. Mahdavia and M. Yazdani, "An artificial immune algorithm for the flexible job-shop scheduling problem", Future Generation Computer Systems,**(2010)**,pp. 533-541.
[14] L.N.D. Castro and J. Timmis, "An artificial immune network for multimodal function optimization", Evolutionary computation (Proceedings of the 2002 Congress),pp. 699-704.
[15] D. Kalyanmoy, "A Fast and Elitist Multiobjective Genetic Algorithm:NSGA-II,**(2002)**.
[16] I. Kacem, S. Hammadi and P. Borne, "Approach by localization and multi objective evolutionary optimization for flexible job-shop scheduling problems", IEEE Transactions on Systems, Man, and Cybernetics,**(2002)**,pp. 1-13.
[17] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problem", Computers and Industrial Engineering,**(2005)**,pp. 409-425.
[18] I. Kacem, S. Hammadi and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic Mathematics and Computers in Simulation",**(2002)**,pp. 245-276.
[19] H. Zhang and M. Gen, "Multistage-based genetic algorithm for flexible job-shop scheduling problem",Journal of Complexity International,**(2005)** January,pp. 223-232.
[20] Z.X. Ong,J.C. Tay and C.K. Kwoh, "Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules",**(2005)**,pp. 442-455.
[21] M.-H. Tayarani-N and M.-R. Akbarzadeh-T, "A Sinusoid Size Ring Structure Quantum Evolutionary Algorithm", IEEE conference,**(2008)**.
[22] F. Pezzella, G. Morganti and G. Ciaschetti, "A genetic algorithm for the Flexible Job-shop Scheduling Problem", Computers & Operations Research,**(2008)**,pp. 3202-3212.
[23] D. Dasgupta, "Special issue on artificial immune system", IEEE Transactions on Evolutionary Computation,**(2002)**,pp. 225-256

