Job Shop Scheduling Using Genetic Algorithim

Er. Shobhit Gupta[1] Ruchi Gaba[2]
[1]Assistant Professor, Department of CSE & IT, [2]PG Scholar
[1, 2]KITM,Kurukshetra

Abstract—*Genetic algorithms (GA) are wide class of global optimization methods. Many genetic algorithms have been applied to solve combinatorial optimization problems. In job-shop scheduling problem (JSSP) environment, there are j jobs to be processed on m machines with a certain objective function to be minimized. JSSP with j jobs to be processed on more than two machines have been classified as a combinatorial problem. In this paper we used genetic algorithm (GA) with some modifications to deal with problem of job shop scheduling. GA once proposed by John Holland is a stochastic search technique based on Darwin's principle of the survival of the strongest. An initial population is generated randomly including the result obtained by some well-known priority rules such as shortest processing time and longest processing time. From there, the population will go through the process of reproduction, crossover and mutation to create a new population for the next generation until some stopping criteria defined were reached*

I. Introduction

Scheduling is a challenging research topic in the Operations Research and Computer Science domain. Scheduling is the allocation of shared resources over time to competing activities. It has been the subject of a significant amount of literature in the operations research field. Emphasis has been on investigating machine scheduling problems where jobs represent activities and machines represent resources; each machine can process at most one job at a time.

Scheduling refers to the act of assigning resources to tasks or assigning tasks to resources. A schedule is a complete set of operations, required by a job, to be performed on different machines, in a given order. In addition, the process may need to satisfy other constraints such as (i) no more than one operation of any job can be executed simultaneously and (ii) no machine can process more than one operation at the same time[14].

The job-shop scheduling problem is one of the most complicated and typical. It aims to allocate m machines to perform n jobs in order to optimize certain criterion. The scheduling problem is a problem of deciding which job in which order to be done with which machine, when several jobs are processed with several machines. In general, the scheduling problem is classified into several types, if the problem has plural machines. The job shop scheduling problem is one of the most difficult problems of those types. The job shop scheduling problem has several machines of which each machine has each function, and each job is processed according to the specific order. Many works have been reported for this problem using Genetic Algorithm (GA).

II. Job Shop Scheduling Problem

The JSSPs are well known combinatorial optimization problems, which consist of a finite number of jobs and machines. Each job consists of a set of operations that has to be processed, on a set of known machines, and where each operation has a known processing time.The objectives usually considered in JSSPs are the minimization of makespan, the minimization of tardiness, and the maximization of throughput. The total time between the starting of the first operation and the ending of the last operation, is termed as the *makespan*.

The n x m minimum-makespan general job-shop scheduling problem, hereafter referred to as the JSSP, can be described by a set of n jobs $f\{J_i\}_{1\leq j\leq n}$which is to be processed on a set of m machines $\{M_r\}_{1\leq r\leq m}$. Each job has a technological sequence of machines to be processed. The processing of job $J_j$on machine M_r is called the operation O_{jr}. Operation O_{jr}requires the exclusive use of $M_r$for an uninterrupted duration p_{jr}, its processing time. A schedule is a set of completion times for each operation $\{C_{jr}\}1\leq j\leq n;1\leq r\leq m$ that satisfies those constraints[15]. The time required to complete all the jobs is called the makespan L. The objective when solving or optimizing this general problem is to determine the schedule which minimizes L The data includes the routing of each job through each machine and the processing time for each operation (in parentheses).Factors to Describe Job Shop Scheduling Problem is:

A. Arrival Pattern: There are two types of Arrival Patterns:

- Static - n jobs arrive at an idle shop and must be scheduled for work
- Dynamic – intermittent arrival (often stochastic)

B. Number of Machines (work stations)

C. Work Sequence: There are two types of Work Sequence

- Fixed, repeated sequence - flow shop
- Random Sequence – All patterns possible

*D. Performance Evaluation Criterion:*Some Performance Evaluation criterions:

- Makespan – total time to completely process all jobs (Most Common)
- Average Time of jobs in shop
- Lateness
- Average Number of jobs in shop
- Utilization of machinesUtilization of workers

III. Solution Techniques To Handle JSSP

A number of solution techniques to handle the JSSP have been developed over the years. A broad classification of the scheduling techniques is given in Jain [1998]. Initially the techniques are divided into two classes as approximation and optimization techniques.

Optimization based techniques are further classified as efficient techniques and enumerative techniques. Enumerative approach has further two subclasses as branch and bound algorithms and mathematical optimization (mixed and linear integer

programming) based algorithms[13]. On the other side approximation techniques are initially classified as general algorithms and tailored algorithms. Tailored algorithms are either dispatching rules or heuristic based algorithms, whereas general algorithms are classified as AI-based techniques (ANN, GA and Expert Systems) and local search based algorithms.

Fig. 1: Solution approaches to JSSP

A. Optimization Based Approaches

A lot of research work has been carried out, in this area, in the last fifty years. Details are given in the following sections.

1) Efficient Techniques

Johnson [1954] is one of the earliest research works in this area. He developed a heuristic based efficient method for finding an optimal solution for the two and three stage production scheduling problem, while considering the setup time as well. Akers [1956] developed a heuristic based graphical approach for solving the production scheduling problem. Jackson [1956] extended the Johnson's rule and developed a heuristic based approach for handling the job lot scheduling. Hefetz and Adiri [1982] developed an approach for two machines unit time Job-Shop schedule length problem. These techniques are applicable to very small problems and cannot handle a big problem of more than 3 machines efficiently.

2) Enumerative Techniques

Mathematical programming based approaches have been extensively used to solve the JSSP. Balas [1965] developed an addictive algorithm for solving the linear programming model that had 0-1 variables. In his [Balas (1969)] further work he developed a mixed integer programming model for machine scheduling using the disjunctive graphs. Mixed integer programming models for the JSSP were also formulated by Mann [1960], Giffler and Thompson [1960] and Balas [1978]. Some of the researchers [Giffler and Thompson (1960), Nemhauser and Wolsey (1988) and Blazewicz et al (1991)] argued that mixed integer programming had not been leading solution approaches to the practical methods of solution. However, the approach presented in Harjunkoski et al. [2000] is a hybrid one, in which they formulated the JSSP as a combination of mixed integer and constraint logic programming.

B. Approximation Based Approaches:

Approximation based approaches offer a good alternatives for solving the JSSP in terms of the quality of solution and computational time.

1) Tailored Algorithms

A problem of ***N*** jobs to be scheduled on ***M*** machines with the objective ofminimizing Makespan (time elapsed between the start of the first operation and thecompletion of the last operation), ***Cmax***, can be handled in two steps. In first step jobsare assigned to each machine according to their processing requirements. While, insecond step those assigned jobs are sequenced on each machine in such a way that***Cmax*** is minimized. The second step is normally handled by algorithms known as Tailored Algorithm (Bedworth and Bailey [1987]).

2) General Algorithms/ Artificial Intelligence:

General algorithms have been initially classified as AI based approaches and techniques based on local search. According to Jain [1998] the AI-based approaches, that mainly include GA and ANN, have proved to be more efficient especially in the case of NP-Hard problems, where heuristic based solutions are difficult to find.

C. Artificial Neural Networks (ANN)

A number of researchers applied ANN to the JSSP. A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

IV. GENETIC ALGORITHM WITH JSSP

Genetic algorithms have been originally developed by Holland [1]. They work with a population composed of individuals. The genetic structure of a biological individual is composed of several chromosomes. Each of these chromosomes is composed of several genes each of which consists of a number of alleles. In combinatorial optimization, an individual is usually identified with a chromosome. A chromosome is further split into a number of genes (in some papers a gene is also identified with an allele). Before applying a genetic algorithm to scheduling problems, an appropriate encoding (or representation) of the solution must be introduced.

Genetic algorithm is a stochastic searching method with the features of generating and testing. It works on a randomly generated candidate solution pool, which is usually called "population". Each encoded candidate solution is called "chromosome"[10]. During the searching process, the selection, crossover and mutation operators are executed repeatedly until the stop criteria is satisfied.

A. Genetic Algorithm

{
Generate initial population ***P***t
Evaluate population ***P***t
While stopping criteria not satisfied **Repeat**
{
Select elements from ***P***t to copy into ***P***t+l
Crossover elements of ***P***t and put into ***P***t+l
Mutation elements of ***P***t and put into ***P***t+l
Evaluate new population ***P***t+l
Pt = ***P***t+l
}
}

Fig. 2: A standard genetic algorithm.

This kind of problem is known as NP-hard problem that can be solved more efficiently with GA and other metaheuristic methods. In this study, GA was selected to solve it. The major mechanisms of GA are[8]:

1. Creating the chromosome representation,
2. Creating the primary population,
3. Creating the adjustment function for fitness evaluation,
4. Defining the selection strategy,
5. Selecting genetic operators for creating a new generation,
6. Defining the parameter values

The four basic steps in the application of GA to a problem are: representation, selection, crossover and mutation. A number of research papers have been produced by different researchers showing different representations and selection procedures with a variety of crossover and mutation schemes. Representation is considered to be the first step in the implementation of GA to a problem. According to Cheng et al [1996] a total of nine different types of representation have been used while applying GA to the JSSP. Representation based on:

Operations
Jobs
Preference list
Job-pair relation
Priority rule
Disjunctive graph
Completion time
Machines
Random keys

According to Cheng et al. [1996] the first five types are termed as direct representations, whereas, the last four are termed as indirect representations. In all the direct type of representations a production schedule for a JSSP is directly encoded as a chromosome, whereas, in case of indirect representation a sequence of decisions related to scheduling a system (for example dispatching rules) is encoded as a chromosome. Morshad [2006] carried out a comprehensive review of the different representation schemes used by researchers.

After finalizing representation the next step is selection of chromosomes that may take part in crossover and mutation.

B. Chromosome Representation and Decoding

The first step in constructing the GA is to define an appropriate genetic representation (coding). A good representation is crucial because it significantly affects all the subsequent steps of the GA. Many representations for the JSP problem have been developing. In this study, two representations based on working sequence and machine distribution are constructed. If the number of the machine type t(t>1), the genes in each chromosome will be divided into t parts in turn. Each part represents one type of machine. Each operation can only be assigned to the machines which can handle it.

For example, suppose a chromosome is given as [4221134233114234] in 4 job×4 machines problem. Here, 1 implies operation of job J1 , and 2 implies operation of job J2 , and 3 implies operation of job J3 , and 4 implies operation of job J4. Because there are four operations in each job, it appears the four times in a chromosome. Such as number 2 being repeated the forth in a chromosome, it implies four operations of job J2. The first number 2 represents the first operation of job J2 which processes on the machine 1. The second number 2 represents the second operation of job J2which processes on the machine 2, and so on. Therepresentation for such problem is based on two-row structure, as following:

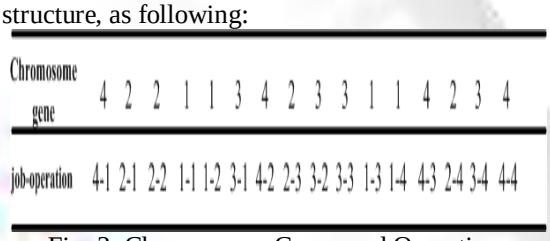

Fig. 3: Chromosome Genes and Operations

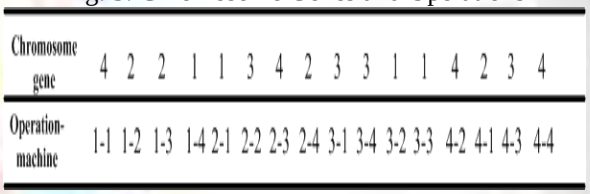

Fig. 4: Chromosome Genes and Machines

So the chromosome of machine 1 is 4 1 3 2, and the chromosome of machine 2 is 2 3 1 4, and the chromosome of machine 3 is 2 4 1 3, and the chromosome of machine 4 is 1 2 3 4.

C. Critical Block and DG distance

In job-shop scheduling problem, we also could find the critical path. Critical path is defined as the longest paths taken from the first operation processed until the last operation leaves the workplace. All operations in this path are called critical operations. Plus, the critical operations on the same machine are called critical block. We measure the distance between two schedules *S*1 and *S*2 by the number of differences in the processing orders of operation on each machine (Mattfeld, 1996). By doing so, it is just like summing the disjunctive arcs whose directions are different between schedules *S*1 and *S*2. This distance also known as disjunctive graph (DG) distance.

D. Data Structures and Representation

Genetic algorithms process populations of strings. We construct a population as an array of individuals where each individual contains the *phenotype*, *genotype* (artificial chromosome) and the *fitness* (objective function). In job-shop scheduling problem we noted the job order on each machine as *phenotype*, the machine schedule as *genotype* and makespan value as *fitness*. The representation used is

called the *permutation representation* (Yamada and Nakano, 1997). JSSP can be viewed as an ordering problem just like the Travelling Salesman Problem (TSP). A schedule can be represented by the set of permutations of jobs on each machine which are called *job sequence matrix*. Figure shows the *job sequence matrix* for 3× 3 problem. Rows will represent the machines and columns represent the order of jobs.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

Fig. 5: A job sequence matrix for 3× 3 problem.

E. Initial population

Most of the research papers in this area prefer to generate the initial population randomly. This technique will allow the search process in a wide space. However, this kind of technique will take a lot of time to get the optimal value. In this paper, we chose to start the initial population with a randomly generated schedules, including some of the schedules obtain by the well-known priority rules such as the shortest processing time and the longest processing time.

F. Crossover

Yamada and Nakano (1997) in most of their papers have introduced plenty of techniques that could be used in solving the job-shop problem. One of them is by making use of CB neighbourhood and DG distance. The idea of this technique is to evaluate a point x by the distance $d(x, p2)$. Let's denote *parent1* and *parent2* as $p1$ and $p2$. First, set $x = p1$. Then, we generated the CB neighbourhood for x,$N(x)$. For each member, yi , in $N(x)$ we calculated the distance between the members and $p2$ to produce $( ) D yi , p2$. Then, we sort $( ) D yi , p2$ in ascending order. Starting from the first index in $( ) Dsort yi , p2$, we accepted yi with probability one if the fitness value is less than the current fitness value$V(yi) \leq V(x)$. Otherwise, we accepted it with probability 0.5. Starting from $p1$, we modified x step by step approaching $p2$. After some iteration, we will find that x will gradually loses $p1$'s characteristics and started to inherit $p2$'s characteristics although in a different ratios. We choose the *child* depending on the less DG distance between the *child* and both its parents. The algorithm for this procedure is shown in Algorithm 1.

Algorithm 1: Crossover
1. Let $p1$ and $p2$ be the parent solution.
2. Set $x = p = q$ 1 .
3. Find CB Neighbourhood for x , $N(x)$.
4. Do
   a. For each member $yi \in N(x)$, calculate the distance between yi and $p2$, $( ) D yi , p2$.
   b. Sort the distance value in ascending order, $( ) Dsort yi , p2$.
   c. Starting from i =1, do
      i. Calculate the fitness value for yi , $( ) V yi$.
      ii. If $V(yi) \leq V(x)$ accept yi with probability one,and with probability 0.5 otherwise.
      iii. If yi is not accepted, increase i by one.
   Repeat i-iii until yi is accepted.
   d. Set $x = yi$
   e. If $V(x) \leq V(q)$ then set $q = x$.
   Repeat 3-4 until number of iterations.
5. q is the child.

G. Mutation

Instead of using some random probability, we apply mutation if the DG distance between *parent1* and *parent2* are less than some predefined value. It is also defined based on the same idea ascrossover. However, we choose the *child* which has the largest distance from the neighbourhood.The algorithm for mutation is shown in Algorithm 2.

Algorithm 2: Mutation
1 Set $x = p1$.
2 Find Neighbourhood for x , $N(x)$
3 Do
   a. For each member $yi \in N(x)$, calculate the distance between yi and $p1$, $( ) D yi , p1$.
   b. Sort the distance value in descending order, $( ) Dsort yi , p1$.
   c. Starting from i = 1, do
      i. Calculate the fitness value for yi , $( ) V yi$.
      ii. If $V(yi) \leq V(x)$ accept yi with probability one, and with probability 0.5 otherwise.
      iii. If yi is not accepted, increase i by one.
   Repeat i-iii until yi is accepted.
   d. Set $x = yi$.
   e. If $V(x) \leq V(q)$ then set $q = x$.
   Repeat 2-3 until number of iteration.
q is the child.

H. Acceptance Criterion

The final and the most important step in the GA procedure is to choose the individual to be replaced by *child*. It is widely known that we always choose the fittest individual to reproduce in the next iteration. In Yamada and Nakano (1997), they did not consider or choose the child with the same fitness value with other population members. However, by not even considering that child, we may lose the good individual without considering its abilities to be evaluated. So, in this paper, after considering the worst individual in the population, we also consider if the child has the same fitness value with the member of population. Instead of dropping that child, we replaced the old one with the child assuming that we have given chance for the old individual to reproduce. Noted that we could not take both of the individuals to avoid having problem later, we might have problem falling in the local optima. The algorithm for the whole procedure is shown in Algorithm 3.

Algorithm 3: Genetic Algorithm
1. Initialize population: Randomly generated a set of 10 schedules including the schedules obtained by some priority rules.
2. Randomly select two schedules, named them as $p1$ and $p2$. Calculate DG distance between $p1$ and $p2$.
3. If DG distance is smaller from some predefined value, apply Algorithm 2 to $p1$. Generate child. Then go to step 5.
4. If DG distance is large, we apply Algorithm 1 to $p1$ and $p2$. Generate child.

5. Apply neighbourhood search to child to find the fittest child in the neighbourhood. Noted it as child'.
6. If the makespan for the child' is less than the worst and not equal to any member of population, replace the worst individual with child'. If there is a member having the same makespan value, replace the member with the child'.
7. Repeat 2-6 until some termination condition are satisfied.

V. METHODOLOGY

Having explained the approaches we provided in our solution to solve the general job shop-scheduling problem this section is dedicated to explain the steps of the methodology. Our approach can be explained according to the Figure 3.

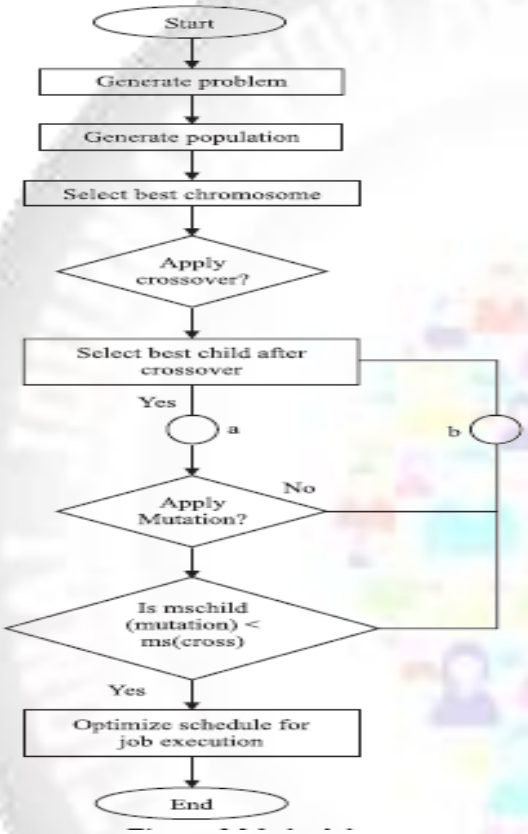

Fig. 6: Methodology

This is the methodology by which our solution provides the solution of the job shop scheduling problem.

VI. RELATED STUDY

Mahanim Omar, Adam Baharum, Yahya Abu Hasan[2] describes job-shop scheduling (JSS) is a schedule planning for low volume systems with many variations in requirements. In job-shop scheduling problem (JSSP) environment, there are j jobs to be processed on m machines with a certain objective function to be minimized. JSSP with j jobs to be processed on more than two machines have been classified as a combinatorial problem. They cannot be formulated as a linear programming and no simple rules or algorithms yield to optimal solutions in a short time. In this paper we used genetic algorithm (GA) with some modifications to deal with problem of job shop scheduling. GA once proposed by John Holland is a stochastic search technique based on Darwin's principle of the survival of the strongest.

E.Falkenauer and S.Bouffouix[3]Job shop scheduling is a combinatorial problem of considerable industrial importance. Despite its notoriety, there are few results offering a satisfying solution to the general problem, although some simplified versions have been successfully treated. The difficulty is due mainly to the high number of constraints, unfortunately unavoidable in the real-world applications. On the other hand, *genetic algorithms* (GAs) constitute a technique that has been applied with advantage to a variety of combinatorial problems. This paper shows how the GAs can be used to optimize the job shop problem with many tasks, many machines and the precedence constraints.

James C. Werner,Mehmet E. Aydin,Terence C. Fogarty[5] addresses an attempt to evolve genetic algorithms by a particular genetic programming method to make it able to solve the classical Job Shop Scheduling problem (JSSP), which is a type of very well-known hard combinatorial optimisation problems. The aim is to look for a better GA such that solves JSSP with preferable scores. This looking up procedure is done by evolving GA with GP. First we solve a set of job shop scheduling benchmarks by using a conventional GA and then an association of GP to evolve a GA.

Jorge Magalhães-Mendes[6]describes Genetic algorithms (GA) are wide class of global optimization methods. Many genetic algorithms have been applied to solve combinatorial optimization problems. One of the problems in using genetic algorithms is the choice of crossover operator. The aim of this paper is to show the influence of genetic crossover operators on the performance of a genetic algorithm. The GA is applied to the job shop scheduling problem (JSSP). To achieve this aim an experimental study of a set of crossover operators is presented. The experimental study is based on a decision support system (DSS). To compare the abilities of different crossover operators, the DSS was designed giving all the operators the same opportunities. The genetic crossover operators are tested on a set of standard instances taken from the literature. The makespan is the measure used to evaluate the genetic crossover operators. The main conclusion is that there is a crossover operator having the best average performance on a specific set of solved instances.

Liang Sun, Xiaochun Cheng, Yanchun Liang[7] presents a genetic algorithm with a penalty function for the job shop scheduling problem. In the context of proposed algorithm, a clonal selection based hyper mutation and a life span extended strategy is designed. During the search process, an adaptive penalty function is designed so that the algorithm can search in both feasible and infeasible regions of the solution space. Simulated experiments were conducted on 23 benchmark instances taken from the OR-library. The results show the effectiveness of the proposed algorithm.

VII. CONCLUSIONS

The study on GA and job shop scheduling problem provides a rich experience for the constrained combinatorial optimization problems. Application of genetic algorithm gives a good result most of the time. Although GA takes plenty of time to provide a good result, it provides a flexible

framework for evolutionary computation and it can handle varieties of objective function and constraint. In this paper GA for solving the job shop scheduling to minimize the makespan. Two-row chromosome structure is designed based on working procedure and machine distribution. The relevant crossover and mutation operation is also given. Finally, The computational result shows that GA can obtain better solution.

REFERENCES

[1] Holland, J.A. Adaptation In Natural And Artificial Systems. Ann Arbor: University Of Michigan, 1975.
[2] Mahanim Omar, Adam Baharum, Yahya Abu Hasan," A Job-Shop Scheduling Problem (Jssp) Using Genetic Algorithm (Ga)"Precedings Of The 2nd Imt-Gt Regional Conference On Mathematics,Statistics And Application.Universtiy Sains Malaysia Penag,13-15 June 2006.
[3] E.Falkenauer And S.Bouffouix "A Genetic Algorithm For Job Shop".
[4] Bhuvan Sharma,Xin Yao"Characterising Ga Approaches To Jssp"
[5] James C. Werner,Mehmet E. Aydin,Terence C. Fogarty" Evolving Genetic Algorithm For Job Shop Scheduling Problems" Proceedings Of Acdm 2000,Pedc, Unviersity Of Plymouth, Uk.
[6] Jorge Magalhães-Mendes "A Comparative Study Of Crossover Operators For Genetic Algorithms To Solve The Job Shop Scheduling Problem"Wseas Transactions On Computers. Issue 4, Volume 12, April 2013.
[7] Liang Sun, Xiaochun Cheng "Solving Job Shop Scheduling Problem Using Genetic Algorithm With Penalty Function " International Journal Of Intelligent Information Processing Volume 1, Number 2, December 2010
[8] Ab. Rahman Ahmad And Faisal. Rm Job Shop Scheduling Using Ga. National Conference Management Science And Operations Research 2003. (2003).
[9] C. Bierwirth , D. C. Mattfeld And H. Kopfer. On Permutation Representations For Scheduling Problems. Parallel Problem Solving From Nature Iv, Springer-Verlag, Pp . 310-318 (1996).
[10] G. Mintsuo And C. Runwei.. Ga And Engineering Design, John Willey & Sons Inc, New York Usa.(2002).
[11] J.F. Gonçalves, , Jorge José De Magalhães Mendes And M. C. Resende. A Hybrid Genetic Algorithm For The Job Shop Scheduling Problem..At&T Labs Research Technical Report Td- 5eal6j, September 2002. (2002).
[12] M. Pinedo And X. Chao. Operation Scheduling With Applications In Manufacturing And Services. Mcgraw-Hill International Editions. (1999).
[13] M. T. Jensen And T. K. Hansen. Robust Solutions To Job Shop Problems. Proceedings Of The 1999 Congress On Evolutionary Computation, Pages 1138-1144. (1999).
[14] S. French. Sequencing And Scheduling : An Introduction To The Mathematics Of The Job Shop. John Willey & Sons Inc, New York Usa.(1982).
[15] T. Yamada And R. Nakano. Genetic Algorithms For Job-Shop Scheduling Problems. Proceedings Of Modern Heuristic For Decision Support. Pp. 67-81, Unicom Seminar, 18-19 March 1997,London. (1997)