CHAPTER 4

GENETIC ALGORITHM BASED MODEL FOR FLOW SHOP SCHEDULING PROBLEMS

4.1 INTRODUCTION

Genetic Algorithms are a class of optimization algorithms that seek improved performance by sampling areas of the solution space that have high probability of leading to a good solution. They imitate the natural evolutionary process is that, in each generation, the fittest individuals have a better chance to produce off-springs by mixing features of the parents or by altering one or more of the parents characteristics whereas, the worst individuals are most likely to die.

Since their introduction, genetic algorithms have been applied to a wide variety of combinatorial optimization problems including the well known Travelling Salesman Problem (Goldberg and Lingle 1985) and the Scheduling Problem (Biegal and Davem 1990, Vempati et al 1993, Neppalli et al 1994 and Chen et al 1995). A survey on applications of GAs can be found in Goldberg (1989). Researchers have used successfully this GA in a wide variety of applications including packing, scheduling, neural networks, traveling salesman, and transport problems (Yenlay 2001, Jaszkiewicz 2002 and Ruben Ruiz 2003).
4.2 SIMPLE GENETIC ALGORITHM

The GA attempts to simulate nature's genetic processes by representing a solution to the problem as a string of genes that can take on some value from a specified finite range or alphabet. This string of genes, which represents a solution, is known as a chromosome. Then an initial population of legal chromosomes is constructed at random. At each generation, the fitness of each chromosome in the population is measured. The fitter chromosomes are then selected to produce offspring for the next generation, which inherit the best characteristics of both the parents. After many generations of selection for the fitter chromosomes, the result is hopefully a population that is substantially fitter than the original. Figure 4.2 shows the schematic representation of GA operation. Simple Genetic algorithm (SGA) consists of the following main components.

4.2.1 Chromosome Representation

Each chromosome represents a legal solution to the problem and is composed of a string of genes. In this research, each solution is encoded as a string of job sequence. For example, the string “2356417” represents a job sequence in which job “2” precedes job “3”, job “3” precedes job “5” and so on (Figure 4.1). The requirement is that repeating a job in a sequence is not allowed and all jobs should appear in the sequence.

2 3 5 6 4 1 7

Figure 4.1 Chromosome Representation
4.2.2 Initial Population

Once a suitable representation has been decided upon for the chromosomes, it is necessary to create an initial population to serve as the starting point for the genetic algorithm. This initial population can be created randomly. From empirical studies, a population size of 30 is selected.

4.2.3 Fitness Evaluation

Since Genetic algorithm is used for maximization problems, a minimization problem can be suitably converted into a maximization problem using a fitness function. The fitness function is,

\[ F = \frac{1}{C_{\text{max}}} \]  

(4.1)

where \( C_{\text{max}} \) is the makespan, which has to be minimized.

4.2.4 Genetic Operators

Chromosomes are selected from the current population for reproduction. The selection procedure picks out parent chromosomes, based on their fitness values, which are then used by the crossover and mutation operators to produce offsprings for the new population. This selection/crossover/mutation cycle is repeated until the new population contains 30 chromosomes. The lower the fitness value the higher the probability of that chromosome being selected for reproduction. Once pairs of chromosomes have been selected, crossover and mutation can take place to produce offsprings. This completes one cycle of genetic algorithm. The fitness of each chromosome in the new population is evaluated and the whole procedure is repeated.
Randomly created initial population

Selection

Crossover

Mutation

Generation end?

End

Yes

No

1 – Crossover Probability

1 – Mutation Probability

Figure 4.2 Simple Genetic Algorithm
4.3 IMPLEMENTATION OF SIMPLE GENETIC ALGORITHM

The implementation of simple genetic algorithm is described in the following sections.

4.3.1 Initialization

In SGA, initialization is often carried out randomly. The objective of this algorithm is to find the optimal sequence so as to minimize the given performance criteria. A job-permutation based encoding scheme has been widely used by many authors for permutation flow shop scheduling. Hence similar encoding scheme is adopted in this work. For example, in a seven-job problem a chromosome could be [4751632]. This represents the sequence where job 4 gets processed first on all the machines followed by job 7 and so on. The permutation has to be feasible, i.e., there can be neither missing nor repetition of jobs.

4.3.2 Fitness Value

A fitness value (objective function) quantifies the performance level of an individual chromosome. In order to mimic the natural process of the survival of the fittest, the fitness evaluation function assigns to each member of the population a value reflecting their relative superiority (or inferiority). Each chromosome has an evaluation criterion based on the objective function. Since GA is used for maximization problems, a minimization problem can be suitably converted into a maximization problem using a fitness function. The fitness function is same as equation (4.1).
4.3.3 Fitness evaluation function for Multi-Objective Criteria

In solving multi-objective optimization problems, one of the popular approaches is to convert the multi-criteria into a single objective using a weighted sum of the criteria, and then solving the problem as a single criterion problem (Venkata Ranga Neppalli et al 1996, Murata et al 1996b and Ponnambalam et al 2004). The weight assigning aspect of this approach is vital and usually projects the relative importance of each criteria.

Venkata Ranga Neppalli et al (1996) used the Vector Evaluated Approach (VEA) and Weighted criteria approach (WCA) based GA algorithms to solve the same set of problems. He stated that the computational results of both the Vector Evaluated Approach and Weighted Criteria Approach are effective in solving the two-stage bicriteria flowshop problem. However, one of the two approaches may be better than the other. The difference in performance for VEA and WCA is not significant and hence, either of the two approaches can be used to measure the fitness of a solution in the population without affecting the performance of the GA based approach (Venkata Ranga Neppalli et al 1996).

Murata et al (1996b) found that their approach with variable weights was capable of approximating the Pareto optimal set in non-convex fronts and produced better results than the VEA.

Combined fitness function

\[ f(x) = \frac{1}{(w_1 \times f_1(x) + \ldots + w_n \times f_n(x))} \]  

(4.2)

where \( f(x) \) is a combined fitness function, \( f_1(x), \ldots, f_n(x) \) are objective functions of performance criteria considered.
In general, value of each weight can randomly be determined. Because the weights $w_i$'s are not constant but variable, the selection probability of each string is also variable even in a single generation (Ponnambalam et al 2001b). This realizes various search directions in the multi-objective improved genetic algorithm.

4.3.4 Selection

Selection in genetic algorithms is defined by selection (reproduction) probabilities for each individual within a population. This selection probability can depend on the actual fitness values and hence they change between generations, or they can depend on the rank of the fitness values only which results in fixed values for all generations. After the offspring has been generated, a number of individuals from both the parents and the offspring have to be selected in order to keep the population size constant.

A fairly different approach to selection is known as tournament selection (Hancock, 1994). Tournament selection is particularly convenient when it is easier to compare individuals than to compute their absolute performance. In Tournament Selection, predetermined numbers of chromosomes are randomly selected from the population and the chromosome with the best fitness value is considered to be regenerated. Here selection is based on a competition within a subset of the population.

4.3.5 Crossover

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the
parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. The individual with highest fitness will be considered first for crossover operation. The next step is to carry the crossover operation which is a reproduction method. This involves two steps, selection of chromosomes for crossover and crossover operation.

The probability of crossover has been assumed to be 0.3-0.8, so that 30-80% of the chromosomes selected for the new population will undergo crossover operation and produce off spring. The procedure for this selection is as follows. Random numbers between 0 and 1 are generated for all chromosomes and those chromosomes that get random number less than $p_c$ value are selected for crossover. If the number of selected chromosome is odd then the above procedure is repeated till one more chromosome gets selected or the number of selected chromosomes become an even number. There are so many crossover methods available in literature. On the basis of a number of pilot runs conducted, two point crossover is selected for SGA. Procedure of two point crossover is given below.

Two cross points are selected and the ends of parent 1 are then copied directly into the child string. The remainder of the child is taken in the same order as parent 2 avoiding duplicate genes. The procedure is explained below (Figure 4.3).

Figure 4.3 Two Point Crossover
Step 1: Two points are randomly selected for dividing the parents (Figure 4.3).
Step 2: The jobs outside the selected two points are directly inherited from parent to the child.
Step 3: The remaining elements in the child are filled by scanning the other parent from left to right and entering the elements not already present.

4.3.6 Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. Mutation applies small random changes to a single chromosome. In the evolution process, this serves the important role of providing new genetic information which was not present in the original population or which has been lost during the selection process and can be tried in a new context. This can result in entirely new gene values being added to the gene pool. With these new gene values, the Genetic Algorithm may be able to arrive at better solution that was previously not possible. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low.

The purpose of mutation is the introduction of new genetic material, or the recreation of good genes that were lost by chance through poor selection of mates. To do this effectively, the effect of mutation must be profound. At the same time, the valuable gene pool must be protected from wanton destruction. There are so many mutation methods available in literature. On the basis of a number of pilot runs conducted, Shift Change
mutation is selected for SGA. Procedure of shift change mutation is given below.

In this type of mutation, a job at a random position is removed and inserted at another random position as shown below (Figure 4.4).

![Figure 4.4 Shift Change Mutation](image)

**Before mutation** → 4 7 1 6 5 2 3

**After mutation** → 4 5 7 1 6 2 3

4.3.7 Stopping Condition

Theoretically, to guarantee the convergence of a GA, a long enough Markov chain is required, which may lead to a large computation requirement (Rudolph 1994). As there is no practicable rule to set a suitable stopping condition, the usual way is to set a maximum number of generations. In this case, the stopping condition is the total number of generations fixed as 10000.

4.4 Framework of the Improved Genetic Algorithm

As for the basic (simple) genetic algorithm, the probability of crossover and mutation are fixed. However, GA suffer the drawbacks including premature convergence, low search efficiency, and difficulty for parameter setting.
GA is naturally parallel and exhibits implicit parallelism, which does not evaluate and improve a single solution, but analyses and modifies a set of solutions simultaneously (Goldberg 1989). The ability of a GA to operate on many solutions simultaneously and gather information from all current solutions to direct search reduces the possibility of being trapped in a local optimum. Unfortunately, a GA may lose solutions and substructures because of disruptive effects of genetic operators, and it is not easy to regulate a GA’s convergence and hence a pure GA may easily produce premature and poor results (Rudolph 1994 and Leung et al 1997).

To enhance the performance of genetic searches and to avoid premature convergence, an Improved Genetic Algorithm (IGA) is proposed with the following changes.

The well known NEH heuristic from Nawaz, Enscore and Ham proposed in 1983 has been recognized as the highest performing method for the permutation flow shop scheduling problem under the makespan minimization criterion. This NEH heuristic’s sequence is incorporated in the generation of initial population since the NEH sequence can generate suboptimal solution rapidly. The diversity of the initial population can be maintained to a certain extent, because the other solutions are still generated randomly.

a. For reproduction of population, two different selection methods are used each with the given probability.

b. A set of crossover operators are used each with the given probabilities for retaining useful information and maintaining diversity if the evolution tends to be premature. Multiple crossover operators
ensure that the diversity can be enhanced and the search region can be extended.

c. Similarly a set of mutation operators are applied in IGA. This will enrich the search template so that the exploration and exploitation abilities can be simultaneously enhanced on the basis of the advantage of combining several different search mechanisms.

d. Elitist strategy: Remove the worst chromosome from the current population and add the best chromosome into that population.

e. Hypermutation: When the number of generations without improving the best solution is greater than a pre-specified constant ($G_m$), premature convergence can be assumed. Then increase the probability of mutation (Hypermutation), and continue the search (Venkata Ranga Neppalli et al 1996). The purpose of increasing the probability is to diversify the population of IGA.

f. Re-Assign (Insert) new randomly generated population if makespan is not converging for the generation $G_p$.

Scheme of IGA is shown in Figure 4.5 and illustrated in sections 4.4.1 and 4.4.3.
Figure 4.5 The Framework of the Improved Genetic Algorithm
Figure 4.5 (Continued)

Is mutation required?

Yes

Mutation
Use Arbitrary 3 job change, Arbitrary 2 job change and Shift change Mutation based on probability attached to each

No

If makespan not converging for $G_m$
Generation change mutation probability (multiply it by $\gamma$)

If makespan not converging for $G_p$
Generation insert new 75% of randomly generated population

Next generation $k++$
4.4.1 Implementation of the IGA for Flowshop Scheduling

The details of various parameter used in the design of IGA for flowshop scheduling problem are given below.

4.4.1.1 Solutions encoding and population initialization

In SGA the population is generated randomly. In IGA NEH heuristic’s sequence is used as one of the chromosomes in the initial population. The remaining are generated randomly and they are subject to feasibility condition. NEH heuristic can be described by the following three steps:

1. For each job compute the total processing time on the m machines:

\[ \forall j, \sum_{i=1}^{m} P_{ij} \]  \hspace{1cm} (4.3)

2. Sort the jobs in descending order of \( P_j \). Let the resulting sequence be \( \phi \). Next, the first two jobs (that is, \( \phi(1) \) and \( \phi(2) \)—the ones with the largest total processing time) are chosen and the two possible sequences of these two jobs are evaluated.

3. Finally, repeat the following steps until all jobs are sequenced. In the \( i^{th} \) step, the job \( \phi(i) \) at position \( i \) is taken and tentatively inserted into all the possible \( i \) positions of the sequence of the jobs that are already scheduled. Select the sequence that results in the minimum makespan. For example, if \( i = 4 \), the previously built sequence would contain the first three jobs of the list \( \phi \) generated in step 2; then, the fourth job could be placed either in the first, the
second, the third or the last position of the sequence. The best sequence of the resulting four sequences would be selected for the next iteration.

4.4.1.2 Fitness evaluation function for Single Objective Criteria

Each chromosome has an evaluation criterion based on the objective function. Since IGA is used for maximization problems, a minimization problem can be suitably converted into a maximization problem using a fitness function. The fitness function is same as equation (4.1) in Section 4.2.3.

4.4.1.3 Fitness evaluation function for Multi-Objective Criteria

Fitness function for multi-objective IGA is same as SGA (section 4.3.3).

4.4.1.4 Selection Scheme

In IGA, two classical selection schemes, namely roulette wheel selection and tournament selection are considered and either of the method is used based on predefined probability.

4.4.1.4.1 Roulette Wheel Selection

In roulette wheel selection, each string is given a segment of the roulette wheel proportional to its fitness. In roulette wheel selection, parents are selected according to their fitness value. The better the fitness, the more chances to be selected.
The following procedure is used for selection. Chromosome $X_i$ is selected if

$$
\sum_{j=1}^{i-1} \frac{f(X_j)}{\sum_{j=1}^{i} f(X_j)} < r \leq \frac{\sum_{j=1}^{i} f(X_j)}{\sum_{j=1}^{\text{p}} f(X_j)}
$$

(4.4)

where $f(X_i)$ is fitness value of chromosome and $r$ is the random number between 0 and 1.

4.4.1.4.2 Tournament Selection

Tournament selection is explained in Section 4.3.4

4.4.1.5 Crossover

Many different general and specific crossover operators have been proposed for the PFSP in the literature. Since the sequence is a permutation of elements there should be neither repeated nor missing elements to maintain feasibility. In this work, following types of crossover operators are considered.

1. Two point Crossover
2. Partially Mapped Crossover (PMX)
3. Similar Job Order Crossover (SJOX)
4. Linear Order Crossover (LOX)

A probability is attached to each type of crossover and each time anyone type is selected. After crossover the resultant chromosomes are
verified for feasibility condition. Crossovers used in this research are explained below.

4.4.1.5.1 Two point Crossover

Two point crossover is explained in section 4.3.5.

4.4.1.5.2 Partially Mapped Crossover (PMX)

Partially Mapped/Matched Crossover or PMX for short. Two positions are chosen on the string at random (as with Two Point Crossover). The substrings between these two points are called the Mapping Sections. These sections are then mapped across onto each other as shown. The procedure is explained below (Figure 4.6).

![Figure 4.6 Partially Mapped Crossover](image)

**Figure 4.6 Partially Mapped Crossover**

Step 1: Two points are randomly selected for dividing the parents. The section of the parents between these two points is called the mapping section (Figure 4.6).

Step 2: Exchanges the mapping section of the parent to the offspring. i.e., mapping section of the first parent is copied into the second offspring and so on.
Step 3: Define one to one mapping between genes of mapping section of the two parents. For the above example mapping is as follows (Figure 4.7).

3→1 4→6 5→5

Figure 4.7 One to One Mapping

Step 4: Offspring is filled up by copying the genes from direct parent by scanning from left to right. In case a gene is already present in the offspring, it is replaced according to the mapping.

4.4.1.5.3 Similar Job Order Crossover (SJOX)

SJOX crossover is based on the idea of identifying and maintaining building blocks in the offspring (Ruben Ruiz et al 2003). In this way similar blocks or occurrences of jobs in both parents are passed over to child unaltered. If there are no similar blocks in the parents the crossover operator will behave like the single point order crossover. SJOX crossover operator can be explained as follows:

Step 1: Both parents are examined on a position-by-position basis. Identical jobs at the same positions are copied over to both the offspring (Fig 4.8(a)).

Step 2: The offspring directly inherits all jobs from the corresponding parents upto a randomly chosen cut point. That is, Child1 inherits directly from Parent1 and Child2 from Parent2 (Fig 4.8 (b)).
Step 3: Missing elements at each offspring are copied in the relative order of the other parent (Fig 4.8(c)).

4.4.1.5.4 Linear Order Crossover (LOX)

Linear Order Crossover, initially suggested by Falkenauer and Bouffouix (1991), works as follows:
Step 1. Select a subsequence of operations from one parent at random.

Step 2. Produce a proto-offspring by copying the subsection sequence into the corresponding positions of it.

Step 3. Delete the operations which are already in the subsequence from the second parent. The resulted sequence of operations contains the operations that the proto-offspring needs.

Step 4. Place the operations into the unfixed positions of the proto-offspring from left to right according to the order of the sequence to produce an offspring.

This procedure is illustrated in Fig. 4.9. Crossover LOX tries to preserve both the relative positions between genes as much as possible and the absolute positions relative to the extremities of parents.
4.4.1.6 Mutation

Mutation generates an offspring solution by randomly modifying the parent’s feature. It helps to preserve a reasonable level of population diversity, and provides a mechanism to escape from local optima. For each child obtained from crossover, the mutation operator is applied independently with a probability $p_m$. In this work, three types of mutation operators are used, which are described as follows. A probability is attached to each type of mutation and each time anyone type is selected using Monte-Carlo simulation.

4.4.1.6.1 Arbitrary Three-job Change Mutation

The three jobs are randomly selected, and replaced at random locations amongst the selected jobs (Figure 4.10).

![Figure 4.10 Arbitrary Three-job Change Mutation](image_url)
4.4.1.6.2 Arbitrary Two-job change Mutation

Two jobs are selected at random and their positions are interchanged as shown below (Figure 4.11).

Before mutation  →  4 7 1 6 5 2 3
After mutation  →  4 7 2 6 5 1 3

Figure 4.11 Arbitrary Two-job Change Mutation

4.4.1.6.3 Shift Change Mutation

Shift Change Mutation is explained in section 4.3.6.

4.4.1.7 Restart Scheme

Finally, to avoid premature convergence and to escape local optimum, two counters called countmut and cntrndpop are introduced as explained below.

1. a. Let $M_k =$ Current generation best makespan and $M_{k-1} =$ Previous generation best makespan.

   If $M_k = M_{k-1}$ then, set countmut=countmut+1 and cntrndpop=cntrndpop+1. Else set countmut =0 and cntrndpop =0.

   b. If there is no improvement in the best solution so far for the pre-specified generations $G_m$, then increase the probability of mutation and continue the search. The purpose of increasing the probability is to diversify the population of IGA. i.e., dynamically change the mutation
rate if makespan is not converging for the generation \( G_m \), i.e., if \( \text{countmut} > G_m \), then \( p_m = \frac{1}{2} p_m \).

II. Re-Assign (i.e., Replace) new randomly generated chromosomes into population if makespan is not converging for the generation \( G_p \). i.e., if \( \text{cntrndpop} > G_p \), then replace 75% of current population by randomly generated new chromosomes.

4.4.1.8 Elitist Strategy

In the process of crossover, mutation and restart scheme, there is a chance of losing the current best chromosome. Elitism refers to a method that copies the best chromosome (or few best chromosomes) to the new population. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution. In IGA, two worst chromosomes from the current population are replaced by the current best chromosomes.

4.4.1.9 Termination Criteria

Theoretically, it requires a long enough Markov chain to guarantee the convergence of GA (Rudolph 1994), which may lead to huge computations. Now there is no practicable rule to set suitable stopping condition and it is also impossible for GA to evolve with too long time in real application, the usual way is to set a limit to a number of generations.

4.4.2 Parameter Settings for IGA

Parameters used in the IGA are shown in Table 4.1. Among these parameters, the quality of solution is mostly influenced by probability of crossover, probability of mutation and the number of generations.
Table 4.1 Parameters of IGA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size $p_s$</td>
<td>70</td>
</tr>
<tr>
<td>Selection methods</td>
<td></td>
</tr>
<tr>
<td>Probability for roulette wheel selection $p_{ro}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Probability for tournament selection $p_{to}$</td>
<td>0.8</td>
</tr>
<tr>
<td>Probability of crossover $p_c$</td>
<td>0.9</td>
</tr>
<tr>
<td>Crossover methods</td>
<td></td>
</tr>
<tr>
<td>Probability for two point crossover $p_t$</td>
<td>0.7</td>
</tr>
<tr>
<td>Probability for PMX crossover $p_p$</td>
<td>0.125</td>
</tr>
<tr>
<td>Probability for SJOX crossover $p_{ij}$</td>
<td>0.05</td>
</tr>
<tr>
<td>Probability for LOX crossover $p_l$</td>
<td>0.125</td>
</tr>
<tr>
<td>Probability of mutation $p_m$</td>
<td>0.40</td>
</tr>
<tr>
<td>Mutation methods</td>
<td></td>
</tr>
<tr>
<td>Probability for arbitrary three-job change</td>
<td>0.05</td>
</tr>
<tr>
<td>Probability for arbitrary two-job change</td>
<td>0.15</td>
</tr>
<tr>
<td>Probability for Shift change mutation $p_{sh}$</td>
<td>0.80</td>
</tr>
<tr>
<td>Probability for arbitrary two-job change mutation $p_{2j}$</td>
<td>0.15</td>
</tr>
<tr>
<td>Probability for Shift change mutation $p_{sh}$</td>
<td>0.8</td>
</tr>
<tr>
<td>Constant for mutation multiplier $\gamma$</td>
<td>1.2</td>
</tr>
<tr>
<td>Number of generations $N_g$</td>
<td>10000</td>
</tr>
<tr>
<td>Generation limit for population re-assignment $G_p$</td>
<td>2500</td>
</tr>
<tr>
<td>Generation limit for changing mutation probability $G_m$</td>
<td>1500</td>
</tr>
</tbody>
</table>
4.4.3 Steps of Improved Genetic Algorithm (Makespan Objective)

Step 0: Given the parameters required, such as population size $P_s$, crossover probability $P_c$, mutation probability $P_m$ etc., set $k=1$ and randomly generate an initial population of size $P_s-1$ plus NEH heuristic solution (Sec. 4.4.1.1). $P(k)=\{X_1(k), X_2(k), \ldots, X_{P_s}(k)\}$.

Step 1: Evaluate Makespan for all the chromosomes in $P(k)$, and set the best two chromosomes $X^*$ and $X^{**}$ respectively.

Step 2: Set current best makespan as $M_k$.

Step 3: If $M_k = M_{k-1}$, then set $\text{countmut} = \text{countmut} + 1$ and $\text{cntrndpop} = \text{cntrndpop} + 1$. Else set $\text{countmut} = 0$ and $\text{cntrndpop} = 0$.

Step 4: Set $q = 0$.

Step 5: Generate a random number $r$. If $r \leq p_a$, select two parents $X_1$ & $X_2$ from $P(k)$ by tournament selection, else select the parents by roulette wheel selection.

Step 6: Generate $r$. If $r \leq p_c$, go to step 6.1 else go to step 6.2.

6.1: Perform anyone of the following crossover operations.

6.1.1: Generate $r$. If $r \leq p_l$, perform two point crossover.

6.1.2: If $p_l < r \leq (p_l + p_p)$ perform Partially Mapped crossover (PMX).

6.1.3: If $(p_l + p_p) < r \leq (p_l + p_p + p_u)$ perform Similar Job Order crossover (SJOX).

6.1.4: Else perform Linearly Ordered Crossover (LOX).
Let $X_1'$ and $X_2'$ be the offsprings of crossovering parents $X_1$ and $X_2$. Go to step 7.

6.2: Let $X_1' = X_1$ and $X_2' = X_2$.

Step 7: Generate $r$. If $r \leq p_m$, go to step 7.1 else go to step 7.2.

7.1: Perform anyone of the following mutation operations.

7.1.1: Generate $r$. If $r \leq p_{3j}$, perform arbitrary three-job change mutation.

7.1.2: If $p_{3j} < r \leq (p_{3j} + p_{2j})$ perform arbitrary two-job change mutation.

7.1.3: Else perform Shift change mutation.

Mutate for $X_1'$ to generate chromosome $X_1''$ and $X_2'$ to generate chromosome $X_2''$. Then, put $X_1''$ and $X_2''$ into $P(k+1)$ and let $q = q+1$. Go to step 8.

7.2: Let $X_1'' = X_1'$ and $X_2'' = X_2'$. Then, put $X_1''$ and $X_2''$ into $P(k+1)$ and let $q = q+1$.

Step 8: If $q < P_s / 2$ then go to step 5; otherwise, go to step 9.

Step 9: If countmut $> G_m$, then change the mutation probability $p_m = \eta \times p_m$ else no change in $p_m$ value.

Step 10: If cntrndpop $> G_p$, then generate new population of size $0.75 \times p_s$ randomly and replace the current population.

Step 11: Update $X^*$, $X^{**}$ and $M_k$ in $P(k)$.
Step 12: Adopt Elitist Strategy. Insert the two best chromosomes $X^*$ and $X^{**}$ into the current population by removing two worst chromosomes (having max. makespan).

Step 13: If $k > N_g$ go to step 14. Else set $k = k + 1$ and go to step 3.

Step 14: Output makespan value and the corresponding sequence as the result. Stop.

4.5 IMPROVED GENETIC ALGORITHM FOR FLOWTIME OBJECTIVE

To determine the schedule for permutation flowshop problem with flowtime objective, the SGA and IGA explained earlier in sections 4.2 & 4.4 are repeated with appropriate fitness value. The fitness function is modified for flowtime objective. The IGA steps for flowtime objective are similar to section 4.3.3.

Implementation of the IGA for flowshop scheduling with total (or mean) flowtime objective is same as discussed in the section 4.3.

4.6 SUMMARY

In this section permutation flowshop scheduling model based on simple genetic algorithm is explained. By adopting some modifications in SGA an improved genetic algorithm is proposed. The model is experimented with single as well as multiple objectives.