Chapter IV
CHAPTER 4
Scheduling with priority rules and genetic algorithm

4.1 Introduction

The research on scheduling in cellular manufacturing systems can be classified into two categories: Scheduling through flow through cells and job shop cells. The difference between these two is the internal flow pattern. In flow-through cell, all the parts are having identical routes. In job shop cell, jobs may visit the machines in any order.

Cellular manufacturing emerged as a production strategy capable of solving the problem of complexity and long manufacturing lead times in batch production systems in the beginning of the 1960s. A batch production environment has been considered in this study. Job shop scheduling involves allocation of jobs to machines to reduce the idle time of the machines. The aim of this scheduling attempt is to reduce the penalty amount that should be paid by the manufacturer if one fails to meet the demand on due date. The scheduling of jobs in a CMS is easier than scheduling in a traditional system, as there are fewer jobs in each cell. The parameters involved in the problem are processing time, due date, penalty, and batch quantity and the objective is to minimize the penalty amount.

4.2 Scheduling in CMS environment and job shop scheduling

Group scheduling has drawn the attention of few researchers. Studies indicate that there is no obvious choice of a particular scheduling method in any given environment. One of the most important issues to attain the full benefits of CMS is the effective implementation of its scheduling system (Hitomi and Ham, 1976). Due to the similarities in the design, shape function etc. parts in a part family generally visit machines in the same sequence with minor differences in setup requirements (Schallar 2001). In group scheduling, it is assumed that
each part family can be processed in one cell by duplicating bottleneck machines or subcontracting exceptional parts (Logendran et al., 1995). However subcontracting exceptional parts may not be practical or duplicating bottleneck machines may not be possible in every CMS environment due to production economics, budget and manufacturing space limitations etc. Thus in a typical environment it is difficult to form independent manufacturing cells and mostly there are some exceptional parts that create inter-cellular moves (Shankar et al., 1998). These constraints limit the applicability of group scheduling methods in real life. Thus the scheduling of CMS is to be considered permitting inter-cellular moves where parts are allowed to visit machines in other cells. The principle that emerges is that one can gain important setup savings by considering family groupings in scheduling work in manufacturing cells, as well as in basic design of such cells.

The job shop scheduling problem consists of a number of machines, \( m \), and a number of jobs, \( n \). Each job consists of \( m \) tasks, each of fixed duration. Each task must be processed on a single specified machine and each job visits each machine exactly once. There is a predefined ordering of the tasks within a job. A machine can process only one task at a time. The due dates are available for the jobs. The makespan is the time from the beginning of the first task to start to the end of the last task to finish. The aim is to find start times for each task such that the penalty/weighted tardiness is minimized.

4.3 Assumptions

Researchers commonly make the following assumptions (Franch, 1981; Baker, 1974).

1. Each machine is an entity.
2. No pre-emption is allowed
3. Each job has \( m \) distinct operations, one on each machine
4. No cancellation of jobs.
5. The processing times are independent of the schedule.

6. In-process inventory is allowed

7. There is only one of each type of machine.

8. Machines may be idle.

9. No machine may process more than one operation at a time

10. Machines never breakdown and are available throughout the scheduling period.

11. The technological constraints are known in advance and are immutable.

12. There is no randomness; all the data are known and fixed.

There are certain assumptions made while solving the JSSP.

4.4 Problem definition

The problem (Natarajan et al., 2000) is represented in Table 4.1

Table 4.1 Representation of problem

<table>
<thead>
<tr>
<th>SI No.</th>
<th>Processing sequence (Machining Time in Min)</th>
<th>Due Date</th>
<th>Batch Quantity</th>
<th>Penalty Rs/Item/Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1(10) / 2(10) / 4(5)</td>
<td>08</td>
<td>200</td>
<td>1.00</td>
</tr>
<tr>
<td>02</td>
<td>2(15) / 3(10)</td>
<td>07</td>
<td>200</td>
<td>1.00</td>
</tr>
<tr>
<td>03</td>
<td>6(10) / 5(10) / 7(5)</td>
<td>06</td>
<td>100</td>
<td>1.25</td>
</tr>
<tr>
<td>04</td>
<td>3(10) / 2(5)</td>
<td>08</td>
<td>200</td>
<td>1.50</td>
</tr>
<tr>
<td>05</td>
<td>1(5) / 4(10) / 5(5)</td>
<td>05</td>
<td>300</td>
<td>2.00</td>
</tr>
<tr>
<td>06</td>
<td>1(10) / 2(10) / 4(5)</td>
<td>10</td>
<td>200</td>
<td>1.00</td>
</tr>
<tr>
<td>07</td>
<td>4(5) / 1(5) / 2(5)</td>
<td>06</td>
<td>300</td>
<td>1.50</td>
</tr>
<tr>
<td>08</td>
<td>5(5) / 7(10)</td>
<td>15</td>
<td>250</td>
<td>1.50</td>
</tr>
<tr>
<td>09</td>
<td>3(10) / 2(10)</td>
<td>03</td>
<td>300</td>
<td>1.25</td>
</tr>
<tr>
<td>10</td>
<td>4(10) / 1(5)</td>
<td>10</td>
<td>200</td>
<td>1.00</td>
</tr>
<tr>
<td>11</td>
<td>7(15) / 5(10)</td>
<td>04</td>
<td>300</td>
<td>1.25</td>
</tr>
</tbody>
</table>

The next section discusses the proposed algorithm.
4.5 Proposed algorithm

STEP 1: Read the problem.

STEP 2: Form cells using any of the Cell Formation (CF) methods.

STEP 3: Calculate the Priority Index (Priority index 1) for all jobs using the formula.

\[
PI = \frac{\text{Quantity} \times \text{Penalty for Lateness}}{\text{Work content} \times \text{Available time}}
\]  

(4.1)

STEP 4: Arrange the jobs in descending order of PI

STEP 5: Load the jobs as per the arranged order.

4.6 Validation

The proposed priority index is applied to the problem shown in table 4.1

STEP 1: The problem in table 4.1 is read.

STEP 2: Using a cell formation algorithm two cells are formed as shown in table 4.2.

<table>
<thead>
<tr>
<th>Table 4.2 Cell details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell No.</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

STEP 3: The Priority Index (PI) for each job is calculated using equation (4.1).

The work content and available time are in days. Two shifts of 8 hours each have been considered.

For job no: 9 the PI is calculated as under:

\[
\text{PI} = \frac{(300 \times 1.25)}{(20 \times 300 / (16 \times 60)) \times 3}
\]

\[= 20\]
The PI values of Jobs are as under

<table>
<thead>
<tr>
<th>Job No</th>
<th>01</th>
<th>02</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>09</th>
<th>10</th>
<th>03</th>
<th>08</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>4.80</td>
<td>5.49</td>
<td>12</td>
<td>19.2</td>
<td>3.84</td>
<td>16.00</td>
<td>20</td>
<td>6.40</td>
<td>8.00</td>
<td>6.67</td>
<td>12</td>
</tr>
</tbody>
</table>

STEP 4: The jobs are arranged in descending order of PI.

<table>
<thead>
<tr>
<th>Job No</th>
<th>09</th>
<th>05</th>
<th>07</th>
<th>04</th>
<th>10</th>
<th>02</th>
<th>01</th>
<th>06</th>
<th>11</th>
<th>03</th>
<th>08</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>20.00</td>
<td>19.20</td>
<td>16.00</td>
<td>12</td>
<td>6.40</td>
<td>5.49</td>
<td>4.80</td>
<td>3.84</td>
<td>12</td>
<td>8.00</td>
<td>6.67</td>
</tr>
</tbody>
</table>

STEP 5: The jobs are loaded in the machine based on PI values in descending order.

Loading sequence

09 05 07 04 10 02 01 06 11 03 08

4.6.1 Penalty calculation

<table>
<thead>
<tr>
<th>M/c</th>
<th>Job</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>Max Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>09</td>
<td>09</td>
<td>10</td>
<td>20</td>
<td>00</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>00</td>
<td>05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>07</td>
<td>20</td>
<td>25</td>
<td>25</td>
<td>30</td>
<td>15</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>30</td>
<td>35</td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>35</td>
<td>45</td>
<td>50</td>
<td>60</td>
<td>60</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>45</td>
<td>55</td>
<td>60</td>
<td>70</td>
<td>70</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>20</td>
<td>30</td>
<td>00</td>
<td>15</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>30</td>
<td>40</td>
<td>00</td>
<td>10</td>
<td>40</td>
<td>45</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>08</td>
<td>40</td>
<td>45</td>
<td>45</td>
<td>55</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3 Time chart

38
To make one component (No.9), the elapsed time is 20 minutes.
Assuming working hours per day as 16 hours, for 300 components it will take 7 days as shown below.

\[
(20 \times 300)/(16 \times 60) = 6.25 \text{ say 7 days.}
\]

Delay time to meet due date is 7-3 = 4 days.

Penalty = No of delay days * batch quantity * Penalty for lateness

\[
= 4 \times 300 \times 1.25
\]

\[
= \text{Rs. 1500.00}
\]

In a similar way, the penalty for all the jobs is calculated and the total penalty for this job sequence is Rs. 10,350.00.

The following methods have also been studied and the results with these methods have been tabulated.

4.7. Priority index using slack, work content and penalty details (Priority index 2)

\[
\text{PI} = \frac{\text{Due date – processing Time}}{\text{Processing Time x Penalty for Lateness}}
\]  \hspace{1cm} \text{(4.2)}

For Job no: 9, assuming two shifts of work,

\[
\text{PI} = \frac{(3 \times 16 \times 60 – 20 \times 300)}{20 \times 300 \times 1.25}
\]

\[
= -0.416
\]

The PI values of jobs are as under.

<table>
<thead>
<tr>
<th>Job No</th>
<th>01</th>
<th>02</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>09</th>
<th>10</th>
<th>03</th>
<th>08</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>0.53</td>
<td>0.34</td>
<td>1.04</td>
<td>-0.1</td>
<td>0.92</td>
<td>0.18</td>
<td>-0.41</td>
<td>2.2</td>
<td>1.04</td>
<td>1.89</td>
<td>-0.39</td>
</tr>
</tbody>
</table>
The jobs are arranged in ascending order of PI values.

<table>
<thead>
<tr>
<th>Job No</th>
<th>09</th>
<th>05</th>
<th>07</th>
<th>02</th>
<th>01</th>
<th>06</th>
<th>04</th>
<th>10</th>
<th>11</th>
<th>03</th>
<th>08</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>-0.41</td>
<td>-0.1</td>
<td>0.18</td>
<td>0.344</td>
<td>0.536</td>
<td>0.92</td>
<td>1.04</td>
<td>2.2</td>
<td>-0.39</td>
<td>1.04</td>
<td>1.89</td>
</tr>
</tbody>
</table>

The schedule is found to be as under.

Loading Sequence: 09 05 07 02 01 06 04 10 11 03 08

The penalty for lateness is Rs.13,450.00.

4.8 Critical ratio method

This method calculates the critical ratio as detailed below.

\[
\text{CR} = \frac{\text{Total work content} \times \text{Penalty for lateness}}{\text{Available makespan}} \quad (4.3)
\]

CR (Priority index 3) is calculated as under.

For Job no: 9, assuming two shifts of work,

\[
\text{CR} = \frac{(300 \times 20 \times 1.25)}{3 \times 16 \times 60}
\]

\[
= 2.60
\]

The CR values of jobs are as follows.

<table>
<thead>
<tr>
<th>Job No</th>
<th>01</th>
<th>02</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>09</th>
<th>10</th>
<th>03</th>
<th>08</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>0.65</td>
<td>0.74</td>
<td>0.58</td>
<td>2.5</td>
<td>0.52</td>
<td>1.17</td>
<td>2.60</td>
<td>0.31</td>
<td>0.54</td>
<td>0.39</td>
<td>2.44</td>
</tr>
</tbody>
</table>

The jobs are arranged in descending order of CR values. The schedule using Critical Ratio has been found to be as follows with a penalty for lateness as Rs. 14,650.00.

Loading Sequence: 09 05 07 02 01 04 06 10 11 03 08
4.9 Results and discussions

The results obtained by various techniques are summarized in table 4.4. From the table, it is observed that the proposed method minimizes the penalty amount.

Table 4.4 Comparison of results

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Penalty in Rs.</th>
<th>Cycle time in Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed method with priority index 1</td>
<td>10,350.00</td>
<td>75</td>
</tr>
<tr>
<td>Method considering Slack priority index 2</td>
<td>13,450.00</td>
<td>85</td>
</tr>
<tr>
<td>Critical Ratio priority index 3</td>
<td>14,650.00</td>
<td>100</td>
</tr>
</tbody>
</table>

The above procedure has been implemented in the high level language ‘C’ and has been tested. The output of the program for the problem discussed above is given under.

JOB SHOP PROBLEM - MINIMISING THE PENALTY COST

GIVE THE NO OF JOBS ==> 11
GIVE THE NO OF MACHINES ==> 7

JOBS INVOLVED ARE.....1 2 3 4 5 6 7 8 9 10 11

JOB MACHINING TIME MATRIX

10 10 0 5 0 0 0 0
0 15 10 0 0 0 0 0
0 0 0 0 10 10 5
0 5 10 0 0 0 0 0
5 0 0 10 5 0 0 0
10 10 0 5 0 0 0 0
5 5 0 5 0 0 0 0
0 0 0 0 5 0 10
0 10 10 0 0 0 0 0
5 0 0 10 0 0 0 0
0 0 0 0 10 0 15

JOB SEQUENCE DETAILS ARE AS UNDER

1 2 4 0 0 0 0 0
2 3 0 0 0 0 0 0
6 5 7 0 0 0 0 0
3 2 0 0 0 0 0 0
1 4 5 0 0 0 0 0
1 2 4 0 0 0 0 0
4 1 2 0 0 0 0 0
5 7 0 0 0 0 0 0
3 2 0 0 0 0 0 0
4 1 0 0 0 0 0 0
7 5 0 0 0 0 0 0
DUE DATE DETAILS ARE AS UNDER
8 7 6 8 5 10 6 15 3 10 4

BATCH QUANTITY DETAILS ARE AS UNDER
200,200,100,200,300,200,300,250,300,200,300,

THE PENALTY FOR LATE ARE AS UNDER
1.0000, 1.0000, 1.2500, 1.5000, 2.0000, 1.0000,
1.5000, 1.5000, 1.2500, 1.0000, 1.2500,

OPTIMUM JOB SEQUENCE ===>> 9-5-7-4-11-3-10-8-2-1-6

TOTAL TIME FOR JOB NO 9 = 20 min PENALTY = Rs 1500.00
TOTAL TIME FOR JOB NO 5 = 20 min PENALTY = Rs 1200.00
TOTAL TIME FOR JOB NO 7 = 30 min PENALTY = Rs 1800.00
TOTAL TIME FOR JOB NO 4 = 35 min PENALTY = Rs 0.00
TOTAL TIME FOR JOB NO 11 = 30 min PENALTY = Rs 2250.00
TOTAL TIME FOR JOB NO 3 = 45 min PENALTY = Rs 0.00
TOTAL TIME FOR JOB NO 10 = 35 min PENALTY = Rs 0.00
TOTAL TIME FOR JOB NO 8 = 55 min PENALTY = Rs 0.00
TOTAL TIME FOR JOB NO 2 = 60 min PENALTY = Rs 1200.00
TOTAL TIME FOR JOB NO 1 = 65 min PENALTY = Rs 1200.00
TOTAL TIME FOR JOB NO 6 = 75 min PENALTY = Rs 1200.00

TOTAL PENALTY AMOUNT = Rs 10350.00

A GA based procedure has been applied for the above problem and the details are presented in the next section.

4.10 Solution through Genetic Algorithm

An attempt has been made to minimize the total penalty amount considering the parameters, viz processing time, due date, penalty cost, and batch quantity using GA. GAs are computerized search and optimization algorithms based on the mechanics of natural genetics and natural selection, which was first developed by professor John Holland (1975) and his student of the University of Michigan.
4.11 General optimization procedure

The figure 4.1 presents an outline of the steps usually involved in an optimal design formulation process.

Figure 4.1 A flow chart of optimal design procedure.

The steps involved in optimal design depicted in figure 4.1 can be applied to scheduling problems as discussed in the following sections.

4.11.1 Classes of search techniques

Search techniques can be broadly classified into three types as shown in figure 4.2.

They are

1. Calculus based methods.
2. Enumerative methods.
4.11.2 Calculus based techniques

Calculus based techniques use a set of necessary and sufficient conditions to be satisfied by the solutions of an optimization problem. These techniques are subdivided into indirect and direct methods.

Indirect methods seek local extrema by solving the nonlinear set of equations resulting from setting the gradient of the objective function equal to zero. The search for possible solutions (function peaks) starts by restricting itself to points with zero slopes in all directions.

Direct methods such as those of Newton and Fibonacci seek local optima (Extrema) by hopping on the function and moving in a direction related to the local gradient. This is simply the notation of the “hill climbing”, which finds the best local point by climbing the
function in the steepest permissible direction. These techniques can be used only on a restricted set of problems.

Both methods are local in scope; the optima the methods seek are the best in a neighborhood of the current point. Starting the search or zero finding procedures in the neighborhood peak of the lower will cause missing the main event (the higher peak). Furthermore, once the lower peak is reached, further improvement must be sought through random restart or other suitable means.

4.11.3 Enumerative techniques

Enumerative schemes have been considered in many shapes and sizes. The idea is fairly straightforward; within a finite search space or a discretized infinite search space, the search algorithm starts looking at objective function values at every point in the space, one at a time. Dynamic Programming is a good example of an enumerative technique.

Although the simplicity of this type of algorithm is attractive and enumeration is a very human kind of search (when number of possibility is small), such schemes must ultimately be discounted in the robustness race for the simple reason, that is, lack of efficiency.

4.11.4 Guided random search techniques

Guided random search algorithms are based on enumerative schemes but use additional information to guide the search. They are quite general in scope and can solve very complex problems. The two major subclasses are simulated annealing and evolutionary algorithms and both are evolutionary processes. Simulated annealing uses a thermodynamic evolution process to search minimum energy states.

Evolutionary algorithms are based on natural selection process. This form of search evolves through generations, improving the features of potential solutions by means of
biologically inspired operations. These techniques subdivide, in turn, into Evolution Strategies and GAs. Rechenberg and Schwefel proposed evolutionary Strategies in early 1970s. They adapted the process of "Artificial Evolution" to the requirement of the local response surface. This means that unlike traditional GAs, evolutionary strategies can adapt their major strategy parameters according to the local topology of the objective function.

Figure 4.3 shows the four stages in a typical GA cycle. Each cycle produces a new generation of possible solutions for a given problem. At the first stage, an initial population of potential solutions is created as a starting point for each search. Each element of the population is encoded into a string (the chromosome) to be manipulated by the genetic operators. In the next stage, the performance (or fitness) of each individual is evaluated with respect to the constraints imposed by the problem. Based on each individual's fitness, a selection mechanism chooses "mates" for the genetic manipulation process. The selection policy is ultimately responsible for assuring the survival of the best-fitted individuals. The combined evaluation and selection process is called reproduction.

The manipulation process uses genetic operators to produce a new population of individuals (offspring) by manipulating the "genetic information" referred to as genes, possessed by members (parents) of the current population. It comprises two operations:
crossover and mutation. Crossover recombines a population’s genetic material. The selection process associated with recombination assures that special genetic structures, called building blocks, are retained for future generations. The building blocks then represent most fitted genetic structures in a population.

The recombination process alone cannot avoid the loss of promising building blocks in the presence of other genetic structures, which could lead to local minima. It cannot explore search space sections not represented in the population’s genetic structures. Here mutation operation comes into action. The mutation operation introduces new genetic structures in the population by randomly modifying some of the building blocks, helping the search algorithm escape from local minima traps. Since the modification is not related to any previous genetic structure of the population, it creates different structures representing other sections of the search space.

The crossover operator takes two chromosomes and swaps part of their genetic information to produce new chromosomes. This operation is analogous to sexual reproduction in nature. The offspring produced by the genetic manipulation process are the next population to be evaluated. GAs can replace either a whole population (general approach) or its less fitted members only (steady state approach). The creation-evaluation-selection-manipulation cycle repeats until a satisfactory solution to the problem is found or some other termination criteria are met.

GAs are different from conventional search procedures in four ways as given below:

1. GAs work with a coding of the parameter set and not with the parameters themselves.
2. GAs search from population of points and not from a single point.
3. GAs use payoff (objective function) information and not on derivatives or other auxiliary knowledge.
4. GAs use probabilistic transition rules and not deterministic rules.

Figure 4.4 A multimodal problem.

Further, GAs are best to solve the multimodal problem with many local optimum points. For example, consider the following bimodal function shown in the figure 4.4. The objective function has one local minimum and one global minimum. If the initial point is chosen to be pointing the local basin (point x(t) in the figure) the calculus based techniques will eventually find the local optimum point. Since the transition rules are rigid, there is no escape from these local optima. The only way to solve the above problem to global optimality is to have a starting point in the global basin. Since this information is usually not known in any problem, the traditional method fails to locate the global optimum. However, these traditional methods can be best applied to a special class of problems suitable for those methods. For example, the gradient search methods will outperform almost any algorithm in solving continuous, unimodal problems, but they are not suitable for multimodal problem. Thus, in general, traditional methods are not robust. A robust algorithm can be designed in such a way that it uses the steepest ascent direction (or any other direction) with some probability. Such a mixed strategy may require more number of function evaluations to solve continuous, unimodal problems, because of the extra computations involved in trying with non-descent directions. This strategy may be able to solve complex, multimodal problem,
shown in the above figure 4.4. The mixed strategy may take the point \( x(t) \) into global basin (when tried with non-descent directions) and finally find the global optimum point. GAs use similar search strategies by using probability in all their operators. Since an initial random population is used, to start with, the search can proceed in any direction and no major decisions are made in the beginning. Later on, when the population begins to converge in some positions, the search direction narrows and a near-optimal solution is achieved. This nature of narrowing search space is adaptive and is a unique characteristic of GA.

### 4.11.5. Reason for selection of GA for scheduling

A simple static sequencing problem with \( n \) jobs is considered, assuming that the computer is relatively fast, capable of generating all possible sequences and evaluating them at a rate of 1 millionth of a second per job within each possible sequence. When there are 3 jobs (say A, B and C), then there are 6 (i.e., \( n! \)) possible sequences of 3 jobs each:

\[
\begin{align*}
A - B - C & \quad A - C - B \\
B - A - C & \quad B - C - A \\
C - A - B & \quad C - B - A
\end{align*}
\]

The total time required to solve this problem by complete enumeration is given below.

\[
\frac{1}{1,000,000} \times 6 \times 3 = 0.000018 \text{ seconds}
\]

The logical expansion of this to larger values of \( n \) is presented in Table 4.5.

### Table 4.5 Solution Times for Complete Enumeration

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Problem Size (n jobs)</th>
<th>Time taken for Complete Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
<td>0.000001 second</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>0.000004 second</td>
</tr>
<tr>
<td>03</td>
<td>10</td>
<td>36.288 seconds</td>
</tr>
<tr>
<td>04</td>
<td>20</td>
<td>( 1.54 \times 10^4 ) years</td>
</tr>
<tr>
<td>05</td>
<td>50</td>
<td>( 4.82 \times 10^{32} ) years</td>
</tr>
</tbody>
</table>
Clearly, even for reasonably sized problems, complete enumeration is not a tractable approach to solving the problem. If the computational speed is increased by a billion-fold, it will still take $4.82 \times 10^{43}$ years to determine an optimal sequence of 50 jobs using complete enumeration. Hence, a search technique, which has more robustness and higher efficiency to search the optimum result among the possible $n!$ available solutions is needed. As GAs offer reasonably best solution with less number of iterations, this method is chosen to solve the scheduling problem. From the literature, it is found that, GAs are more robust and having higher efficiency to search the best within the solution space compared to conventional search method such as calculus based techniques and enumerative techniques.

4.12 The genetic algorithm

The steps in implementing the proposed GA based heuristics are listed below.

Step1. Choose a coding system to represent problem parameters, a selection operator, a crossover operator and a mutation operator. Choose suitable population size, crossover probability and mutation probability. Initialize a random population of strings and the maximum allowable generation number $N_{max}$ to a suitable number.

Step2. Evaluate each string in the population.

Step3. If current generation number $N$ is greater than permissible number of generations $N_{max}$ or other termination criteria is satisfied, terminate

Step4. Perform reproduction operation on the population.

Step5. Perform crossover on random pairs of strings.

Step6. Perform mutation on qualified strings.

Step7. Evaluate each string in the current population, set $N = N+1$ and go to Step 3.
4.12.1 Objective function

The objective of the problem is to find the job sequence, for which penalty to be paid is minimum. Objective function = penalty in Rs. = f(x)

4.12.2 Fitness function

Since GAs are meant for maximization problems by the nature of the process, the above minimization problem is converted into an equivalent maximization problem as follows.

\[ \text{Fitness function value } F(x) = \begin{cases} \text{P}_{\text{max}} - f(x) & \text{when } f(x) < \text{P}_{\text{max}}, \\ 0 & \text{otherwise} \end{cases} \]

where \( \text{P}_{\text{max}} \) is taken as the largest penalty value observed in the current population. So, \( \text{P}_{\text{max}} \) varies dynamically with varying population.

4.12.3 Representation of jobs

In order to use GA the jobs are first coded in some string structure. Since job numbers are involved in this problem, coding is done by integer coding method. The coded strings comprising 8 numbers in the first group and 3 numbers in the second group are generated randomly as unique numbers, say, 2 6 4 7 3 8 1 5 11 9 10 representing a complete string or chromosome. Proportionate selection operator, cyclic crossover operator and an interchangeable mutation operator are selected as GA operators. Initially, the crossover and mutation probabilities are assigned to be 0.80 and 0.05 respectively. It is decided to have population size of 50 and number of generations \( N_{\text{max}} \) of 100.

4.12.4 Evaluation

The next step is to evaluate each string in the population. The fitness value for each string generated has been calculated and used for reproduction operation. Time calculation
details for a typical sequence 9-5-7-4-10-2-1-6-11-3-8 is shown in table 4.6. The Gantt chart for the job sequence is shown in fig 4.5.

Table 4.6 Time calculation

<table>
<thead>
<tr>
<th>M/c Job</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>Max Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>09</td>
<td>10</td>
<td>20</td>
<td>00</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>05</td>
<td></td>
<td>15</td>
<td>15</td>
<td>20</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>07</td>
<td>25</td>
<td>25</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>04</td>
<td>30</td>
<td>35</td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>35</td>
<td></td>
<td>20</td>
<td>30</td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>02</td>
<td>35</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>01</td>
<td>35</td>
<td>45</td>
<td>50</td>
<td>60</td>
<td></td>
<td>60</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>06</td>
<td>45</td>
<td>55</td>
<td>60</td>
<td>70</td>
<td></td>
<td>70</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>30</td>
<td>00</td>
<td>30</td>
</tr>
<tr>
<td>03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>40</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>45</td>
<td>45</td>
<td>55</td>
</tr>
</tbody>
</table>

Figure 4.5 Gantt chart for the sequence 9 5 7 4 10 2 1 6 11 3 8

Model Calculation of penalty:

Taking job no 9, the elapsed time is 20 minutes.

Assuming two shifts of operation per day with 8 hours/shift total working hours per day is 16 hours. For producing 300 components, it will take 7 days as shown below.

\[
(20/(16*60))*300 = 6.25 \text{ say 7 days.}
\]
Delay time to meet due date is 7-3 = 4 days.

Penalty = No. of delay days * batch qty * penalty for lateness.

= 4*300*1.25. = Rs. 1500.00

The penalty for all the jobs are calculated in a similar fashion and the total penalty for the current job sequence is found to be Rs. 10,350.00.-.

4.12.5 Checking the termination criteria

If N (No. of cycles) < N_{max} (Allowable cycles) or the average value of fitness function differs from the minimum value of fitness function, one proceeds with the next Step 4, otherwise the process is terminated.

4.12.6 Reproduction

At this Step, good strings in the current population are selected to form the mating pool, through a selection operator. The action of reproduction operator/selection operator is to clear the inferior members from further consideration by probabilistic elimination. The commonly used proportionate reproduction operator has been used where a string is selected for its mating pool proportional to its fitness. A set of six strings have been taken for explanation. The fitness values for the strings are shown in table 4.7.

<table>
<thead>
<tr>
<th>String No.</th>
<th>Strings</th>
<th>Fitness function value in Rs.</th>
<th>Percentage of Distribution</th>
<th>Relative ranking for selection</th>
<th>Actual selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>3 9 5 10 4 2 1 1 8 6 1 7</td>
<td>17350.00</td>
<td>19</td>
<td>1.14</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>4 6 1 7 1 0 3 2 9 8 1 1 5</td>
<td>0.00</td>
<td>00</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>5 1 0 1 1 8 9 7 6 2 3 1 4</td>
<td>21000.00</td>
<td>23</td>
<td>1.38</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>1 1 4 5 8 2 1 0 6 7 3 1 9</td>
<td>12475.00</td>
<td>13</td>
<td>0.78</td>
<td>1</td>
</tr>
<tr>
<td>S5</td>
<td>5 8 6 7 4 9 3 1 0 1 1 2</td>
<td>14375.00</td>
<td>16</td>
<td>0.96</td>
<td>1</td>
</tr>
<tr>
<td>S6</td>
<td>5 9 4 7 1 0 2 1 1 6 3 8</td>
<td>27550.00</td>
<td>29</td>
<td>1.74</td>
<td>2</td>
</tr>
</tbody>
</table>

The sixth string has a higher probability of selection than any other does. Applying steady state replacement-policy, the best string is copied once into the mating pool initially,
followed by the strings copied relative to their weight/ranking as shown in table 4.7. Thus, string no 6 will be copied for being the best string. It is again copied twice considering its weight.

4.12.7 Crossover

At this Step, the strings in the mating pool are subjected to crossover operation. Cyclic crossover has been applied wherein two strings of the population are selected at random and crossed at a random site in the first string. When two strings are selected for crossover, a random number is generated to check whether a crossover is desired or not. If the outcome is true, a crossover site is found at random to perform crossover in the string. Thus, crossover is performed at selected site and two new strings are created. After crossover, the off springs are placed in the intermediate population for the use in the next generation. Cyclic crossover has been performed on the strings in the mating pool, with a crossover probability.

Cyclic crossover procedure is as under (Goldberg, 1989).

Let

\[ P_1 = 9 \ 8 \ 2 \ 1 \ 7 \ 4 \ 5 \ 10 \ 6 \ 3 \ 11 \]
\[ P_2 = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \]

be the parents for crossover with crossover site 4, then,

\[ C_1 = \_ \_ \_ \ 1 \ \_ \_ \_ \_ \_ \_ \_ \_ \]

Since one wants every job to be taken from one of the two parents, the choice of job 1 from P1 means that one must now get job 4 from P1 because of the 4 in the position of parent P2.

\[ C_1 = \_ \_ \_ \ 1 \ _4 \ \_ \_ \_ \_ \_ \_ \]

This selection in turn requires that job 6 is selected from P1. This process continues until the following pattern is obtained.

\[ C_1 = 9 \ _\_ \ _1 \ _4 \ _6 \ _\_ \_ \]
The selection of a 1 means that one should now choose a 4 from P1; however, this is not possible; a 1 having been selected as the first job. That one eventually returns to the job of origin completes a cycle gives the operator its name. Following the completion of the cycle the remaining jobs are filled from the other string. Finally, one gets new off springs C1 (Child 1) and C2 (Child 2) as follows,

C1 = 9 2 3 1 5 4 7 8 6 10 11
C2 = 1 8 2 4 7 6 5 10 9 3 11

4.12.8 Mutation

The next Step is to perform mutation on strings in the intermediate population. For interchangeable mutation, a random number is generated for every gene in a string, with a probability of acceptance, to check whether mutation is required. If a pair of outcome is true, then mutation is effected by interchanging the qualified genes in the string, else next string is selected for mutation. This process is repeated till the last string in a population is dealt with. This mutation operator in GA enables to explore the search space properly before converging to a region prematurely, reaching only a local minimum.

4.12.9 Generation Counter

The next Step is to set N = N+1. The set of steps 3 through 7 are repeated. After the required number of generations the procedure gives the optimal/near optimal solution.

The most difficult and time-consuming issue in the successful implementation of GAs is to find good parameter settings. The following are the points to be considered: Increasing the crossover probability increases the recombination of building blocks, but it also increases the disruption of good strings. Increasing the mutation probability tends to transform the genetic search into a random search, but it also helps to reintroduce lost genetic material. Increasing the population size increases the diversity and reduces the probability that the GA
will prematurely converge to a local optimum. It also increases the time required for the population to converge.

4.13 Results and discussions

The procedure of GA has been implemented using C language and tested. The software has been developed to be interactive and user-friendly. Figure-4.6 shows typical observations from the output of one of the runs of the program with population size 50 and number of generations 100.

![GENERATIONS Vs PENALTY AMOUNT](image)

**Figure 4.6** Variation of average and minimum penalty with generations

The program developed accepts inputs for sequence of jobs, machining time for various operations, penalty amount, batch quantity and due date for the jobs. The output of the program is available in three forms

(i) A text file that records the setting of the parameters for the computational experiment, the variation of the objective function value over generations, Gantt chart etc.

(ii) An EXCEL file that generates the graphical presentation of the average value and minimum value variations of objective function.
(iii) A scheduling software file to show the Gantt chart for the best sequence generated by the
program.

A partial output report of the program is given under.

JOB SHOP PROBLEM - MINIMISING THE PENALTY COST

GIVE THE NO OF JOBS ====> 11

GIVE THE NO OF MACHINES ===> 7

JOBS INVOLVED ARE..... 1 2 3 4 5 6 7 8 9 10 11

JOB MACHINING TIME MATRIX

\[
\begin{array}{cccccccc}
10 & 10 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 15 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & 10 & 5 & 0 & 0 \\
0 & 5 & 10 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 10 & 5 & 0 & 0 & 0 \\
10 & 10 & 0 & 5 & 0 & 0 & 0 & 0 \\
5 & 5 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 5 & 0 & 10 & 0 & 0 \\
0 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & 0 & 15 & 0 & 0 \\
\end{array}
\]

JOB SEQUENCE DETAILS ARE AS UNDER

1 2 4 0 0 0 0
2 3 0 0 0 0 0
6 5 7 0 0 0 0
3 2 0 0 0 0 0
1 4 5 0 0 0 0
1 2 4 0 0 0 0
4 1 2 0 0 0 0
5 7 0 0 0 0 0
3 2 0 0 0 0 0
4 1 0 0 0 0 0
7 5 0 0 0 0 0

DUE DATE DETAILS ARE AS UNDER

8 7 6 8 5 10 6 15 3 10 4

BATCH QUANTITY DETAILS ARE AS UNDER

200, 200, 100, 200, 300, 200, 300, 250, 300, 200, 300,
THE PENALTY FOR LATE ARE AS UNDER
1.00,1.00,1.25,1.50,2.00,1.00,1.50,1.50,1.25,1.00,1.25,

THE % PROBABILITY OF CROSSOVER (01-100): 80

THE % PROBABILITY OF MUTATION (01-100) : 10

THE MAXIMUM NUMBER OF GENERATIONS : 500

TOTAL TIME FOR JOB NO  5 = 20 min PENALTY = Rs 1200.00
TOTAL TIME FOR JOB NO 11 = 30 min PENALTY = Rs 2250.00
TOTAL TIME FOR JOB NO  3 = 45 min PENALTY = Rs  0.00
TOTAL TIME FOR JOB NO  9 = 20 min PENALTY = Rs 1500.00
TOTAL TIME FOR JOB NO  4 = 25 min PENALTY = Rs  0.00
TOTAL TIME FOR JOB NO  7 = 30 min PENALTY = Rs 1800.00
TOTAL TIME FOR JOB NO  2 = 55 min PENALTY = Rs 1000.00
TOTAL TIME FOR JOB NO  8 = 55 min PENALTY = Rs  0.00
TOTAL TIME FOR JOB NO 10 = 35 min PENALTY = Rs  0.00
TOTAL TIME FOR JOB NO  6 = 60 min PENALTY = Rs  600.00
TOTAL TIME FOR JOB NO  1 = 70 min PENALTY = Rs 1400.00

TOTAL PENALTY AMOUNT = Rs 9750.00

The optimum scheduling sequence is found to be 5-11-3-9-4-7-2-8-10-6-1. The optimum scheduling sequence for cell 1 is 5-9-4-7-2-10-6-1 and cell 2 is 11-3-8. The Gantt chart for the above sequence is shown in figure 4.7. In most of the runs of the program, the minimum penalty has been noted to be Rs. 9750.00. Many sequences have been observed for the same penalty amount. Two of such sequences are as under.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Cell 1</th>
<th>Cell 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9-5-4-7-11-2-10-8-3-1-6</td>
<td>9-5-4-7-2-10-1-6</td>
<td>11-3-8</td>
</tr>
<tr>
<td>5-9-4-7-10-11-2-8-3-6-1</td>
<td>5-9-4-7-10-2-6-1</td>
<td>11-8-3</td>
</tr>
</tbody>
</table>

The results of using GA have been compared with the results through other methods as shown in table 4.8. All the priority indices have been defined in the sections 4.5, 4.8 and 4.9. The comparative results have also been shown in the form of chart in figure 4.8.
Table 4.8 Comparative results

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Penalty in Rs.</th>
<th>Max Cycle time in min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic algorithm</td>
<td>9,750.00</td>
<td>70</td>
</tr>
<tr>
<td>Priority index 1 (proposed)</td>
<td>10,350.00</td>
<td>75</td>
</tr>
<tr>
<td>Priority index 2</td>
<td>13,450.00</td>
<td>85</td>
</tr>
<tr>
<td>Priority index 3</td>
<td>14,650.00</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 4.7 Gantt chart for result sequence 5-11-3-9-4-7-2-8-10-6-1

Figure 4.8 Bar chart between methods and penalty amount
The algorithm has been applied with various probabilities for crossover and mutation and number of generations. The combination of probability of crossover 0.80, probability of mutation 0.10 and the maximum number of generation 500 has been observed to give probable optimum result consistently. The algorithm has been coded in C language. The program has been made interactive with provision for changing the values of GA parameters, storing input data matrix, and recording all the results/solutions in an output file. The algorithm has been applied to other problems also and the results are encouraging.

4.14 Conclusion

An attempt has been made to study and solve a penalty involved JSSP, under cellular manufacturing environment, using priority indices and GA. Job based representation has been applied in this study. The results obtained using GAs have been compared with earlier reports. The priority index defined in the proposed method is found to give better solutions. It is concluded that, the sequence obtained from GA approach reduces the penalty amount largely compared to the schedules obtained in other methods using priority indices. The program may be modified to optimize the sequence for minimum cycle time. The program may also be modified for optimizing both penalty amount and cycle time.

Various combinations of values for GA parameters have been applied with the algorithm and the best combination of parameter values has been found and used to find the penalty amount. This indicated that GA parameter settings influence the quality of solution and the computational time. Besides the setting of GA parameters the excellence in solutions depends on the representation and schedule deduction schemes and the type of genetic operators also. A report on a proposed representation scheme for JSSP through GA and the effect of inversion operator on the performance of the algorithm are presented in the next chapter.