

# **1. Introduction**

## **1.1 Project Overview**

### **Flexible Manufacturing System**

FMS Scheduling problem is one of the most difficult NP-hard combinatorial optimization problems. Therefore, determining an optimal schedule and controlling an FMS is considered a difficult task.

Job scheduling on various machines of varied capabilities has been a tedious task for any organization. Since the ability and functions of modern machines have been widely extended, the scheduling plan of a part might not be unique. So on a real shop floor, many feasible schedules can be found. The increased flexibility of an FMS can require a more flexible scheduling system. The aim of the project is to provide a solution for the problem taking into consideration actual breakdowns in small and medium scale industries by building a scheduling tool.

Scheduling of FMS have received much attention of academicians and practitioners over several decades.

Flexibility is defined as the ability to deal with change by judiciously providing and exploiting Controllable options dynamically. Due to this flexibility, some decision- making problems have occurred in the system. In order to run the system efficiently, the decision points and their importance should be defined and assessed very carefully.[3]

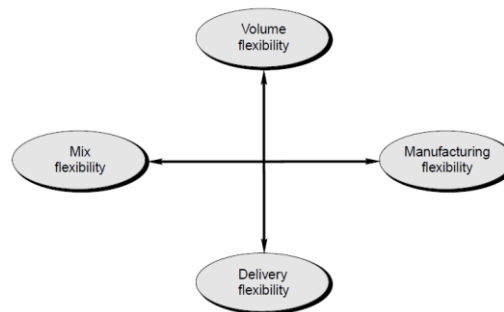
According to [3] there are four stages of decision problems for the FMS: designing, planning, scheduling, and control. The centre of attention of authors is on the scheduling problem after considering that designing and planning phase has been over. In the field of manufacturing control problems of an FMS, scheduling is an extensive area for research which is still alluring many a researchers. Scheduling of operations is one of the most critical issues in the planning and managing of manufacturing processes. Scheduling problem is an assignment problem, which can be defined as the assigning of available resources (machines) to the activities (operations) in such a manner that maximizes the profitability, flexibility, productivity, and performance of a production system.

Most of the analytical models neglect the breakdowns and the maintenance operations that disturb the smooth working of the factory. Scheduling of production in FMS has been extensively researched over the past years and it continues to attract the interest of both academic researchers and practitioners. Generation of new and modified production schedules is becoming a necessity in today's complex manufacturing environment. Uncertainties in the production environment and modelling limitations inevitably result in deviations from the generated schedules. This makes rescheduling or reactive scheduling essential. Since the aspect of breakdowns has not been much included in the researches done till date which motivates us to work on the problem of generating a schedule taking into consideration breakdowns that may happen on the job floor.

In scheduling each job is formed by a sequence of operations, each operation requires exactly one machine, and machines are continuously available and can process one operation at a time without interruption.

Scheduling problem in flexible manufacturing system is more difficult due to the possibility of allocation of operations on any among a set of available machines. Unlike traditional production systems, the manufacturing environment is much more complex and dynamic in a flexible manufacturing system (FMS)

In the 1960s the market became more competitive. Cost was the main concern in the period 1960-1970. After when quality became the priority. With the increasing complexity of the market the speed of delivery became important to the customer. The above required the companies to adapt to an environment which was more flexible than the usual to satisfy the different segments of the market. Therefore the effort to gain a competitive edge led to the innovation of FMS.



**Fig 1.1: Types Of flexibilities**

To sustain in today's competitive global market manufacturing organizations have to develop a manufacturing system that can fulfil the changing demands of customer for customised products. The system should be flexible, productive and should be able to meet the demands within time bounds at a reasonable cost. FMS belongs to a class halfway amidst job shop manufacturing system and batch manufacturing system. An FMS has an integrated and computer controlled configuration which is capable of automatically changing tools and parts. These machines are interconnected by automatic guided vehicles, pallets and storage buffers that have flexibility that allows to modify system behaviour on occurrence of changes whether predicted or unpredicted. It is modelled as a collection of workstations. The FMS should be designed to simultaneously manufacture different volumes of a varying variety of high quality products. The flexibility may be machine flexibility or routing flexibility. Machine flexibility refers to system's ability to produce new product types and change the sequence of operations executed on a part. Routing flexibility is the ability to absorb large scale changes such as in volume, capacity and capability.

The arrangement of machines in an FMS is connected by a transport system. The components are automatically governed using local area network.

### ***Basic components of FMS***

FMS basically composes of the following three parts:

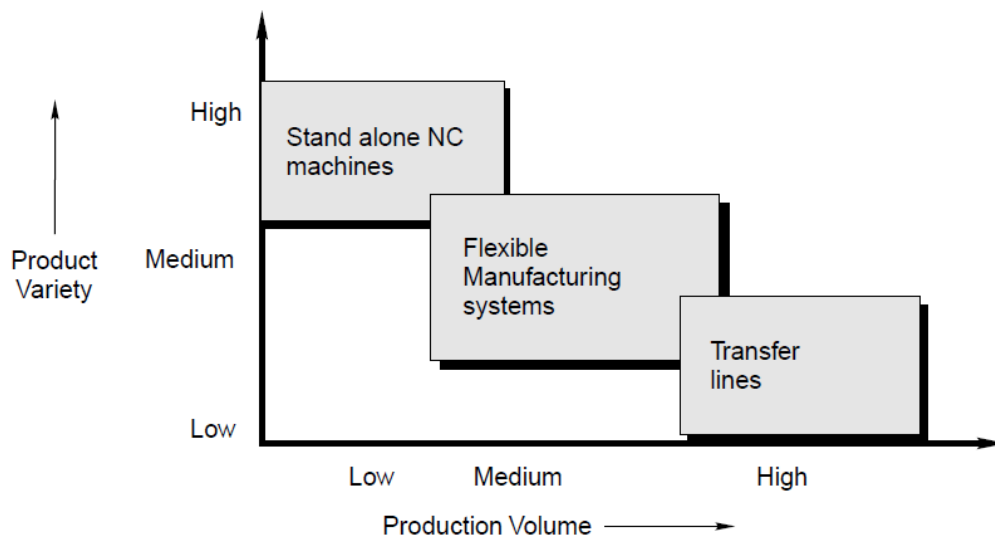
1. *Workstations*: A machine tool which is computer controlled is called a workstation. Machine centres, load/unload stations, assembly workstations, inspection stations, forging stations, sheet metal processing etc. are a few examples of workstations.

2. *Automated Storage stations and Material handling stations*: The movement of work parts and sub assembly parts between different workstations is done mechanically which is referred to as automated material handling and storage system. The functions performed are:

- (i) Random movement of work parts between stations independently
- (ii) Handling various work part configurations
- (iii) Temporary storage
- (iv) Loading and unloading of work parts for easy access
- (v) Computer control compatibility

3. *Computer controlled systems*: The functioning of the stated components is co-ordinated by a controlling Computer System. Its functions are:

- (i) Controlling work stations
- (ii) Control instruction distribution to the work stations
- (iii) Controlling production
- (vi) Controlling traffic
- (v) Monitoring and work handling
- (vi) Monitoring the performance of the system and reporting



**Fig1.2: Application characteristics of FMS**

**Table 1.2: Types of flexibilities**

<i>Approach</i>	<i>Flexibility meaning</i>
1. Manufacturing	<ul style="list-style-type: none"><li>■ The capability of producing different parts without major retooling</li><li>■ A measure of how fast the company converts its process from making an old line of products to produce a new product</li><li>■ The ability to change a production schedule, to modify a part, or to handle multiple parts</li></ul>
2. Operational	<ul style="list-style-type: none"><li>■ The ability to efficiently produce highly customized and unique products</li></ul>
3. Customer	<ul style="list-style-type: none"><li>■ The ability to exploit various dimension of speed of delivery</li></ul>
4. Strategic	<ul style="list-style-type: none"><li>■ The ability of a company to offer a wide variety of products to its customers</li></ul>
5. Capacity	<ul style="list-style-type: none"><li>■ The ability to rapidly increase or decrease production levels or to shift capacity quickly from one product or service to another</li></ul>

Various approaches have been proposed for the scheduling problem of FMS including branch and bound, priority rules, simulated annealing, tabu search and genetic algorithm. Among these approaches the genetic algorithm approach with variations has been extensively applied for scheduling in an FMS.

Genetic algorithm by Holland is a randomised search and optimisation technique which is governed by the principles of evolution and natural genetics. It has been studied that it is difficult for the traditional optimisation techniques to approach the best solution. It is observed that genetic algorithms have better capability than the traditional scheduling methods.

## **Genetic Algorithm**

GA belong to the class of evolutionary algorithms. John Holland based GA on the basic Darwinian Mechanism of “Survival of the fittest”

Genetic Algorithms are an adaptive heuristic search algorithms based on the evolutionary idea of natural selection and genetics. As such they represent an intelligent exploration of a random search used to solve optimization problems.

The principle of GA is simple: imitate genetics and natural selection by a computer program. The parameters of the problem are most naturally coded as DNA- like linear data structure, a vector, a string.

A set called population, of these problem-dependent parameter value vectors is processed by GA. To start there is a random population, the values of different parameters generated by a random number generator. Typically population size is from few dozens to thousands. To do optimization we need a cost function or fitness function. By fitness function we can select the best solution candidates from the population and delete the not so good specimens.

GAs differ from the conventional optimization techniques in the following ways:

- Gas operate with coded versions of the problem parameters rather than parameters themselves i.e. GA works with the coding of solution set and not with the solution itself.
- Almost all conventional optimization techniques search from a single point, but GAs always operate on a whole population of points (strings) i.e. GA uses a population of solutions rather than a single solution for searching.
- GA uses fitness function for evaluation rather than derivatives. As a result, they can be applied to any kind of continuous discrete optimization problem.
- GAs use probabilistic transition operators while conventional methods for continuous optimization apply deterministic transition operators.

Genetic algorithm are preferred over other approaches because of the following advantages

- Genetic Algorithm works smoothly with numerical experimental or analytical functions and data.
- Well suited for parallel computing.
- Genetic Algorithm optimizes both variables efficiently, continuous or discrete.
- It doesn't requires any derivative information.
- It searches from a large sampling of the cost surface.
- It is capable of handling a large number of variables at the same time.
- It can optimizes variables with highly complex cost surfaces.
- Gives a number of optimum solutions, not a single solution.
- It works on encoded variables

A GA typically consists of the following steps:

**Step 1:** Generate the initial population. Obtain the population size and the maximum number of iterations.

**Step 2:** Compute the fitness function for every member of the initial population.

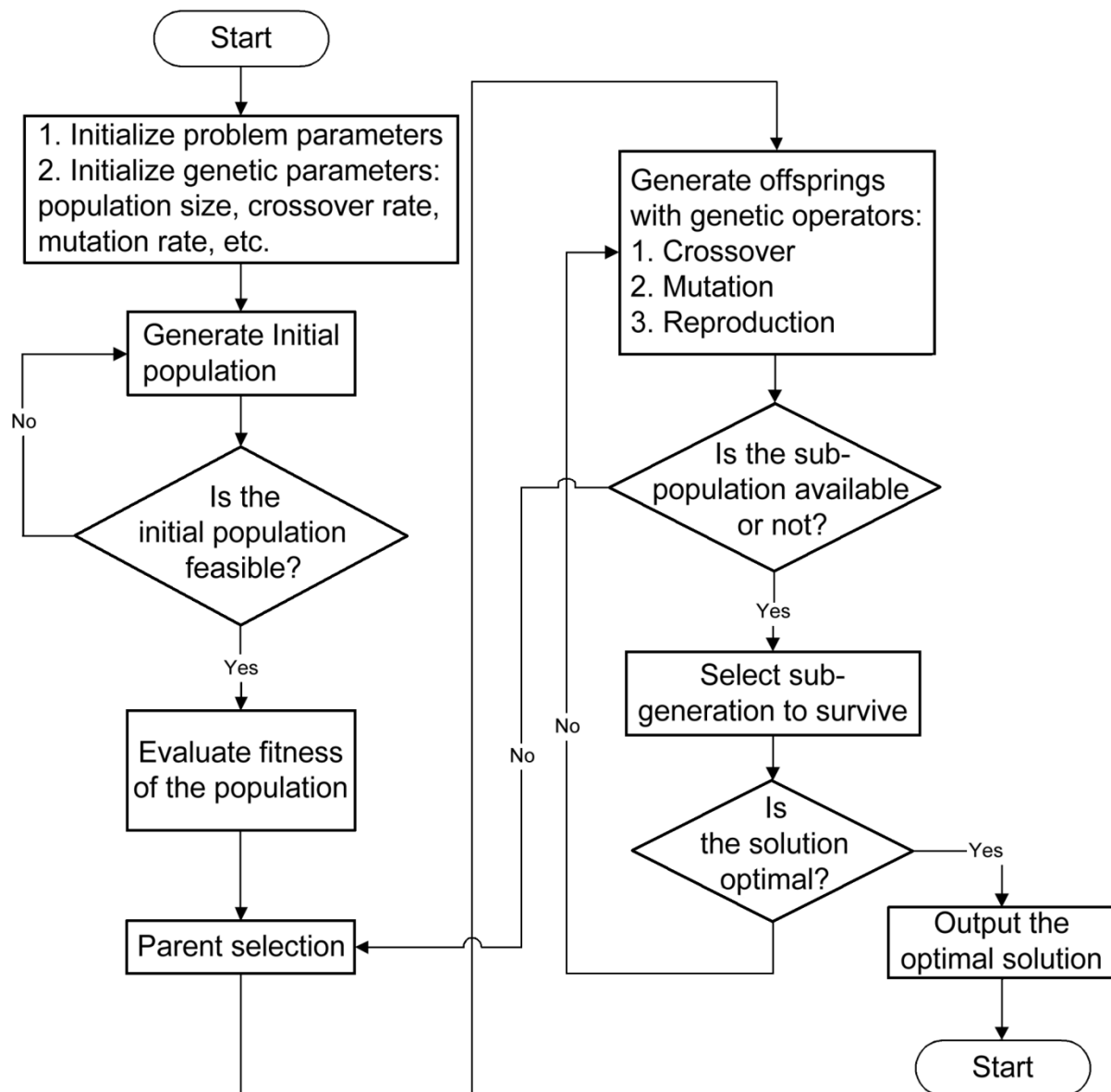
**Step 3:** Compute the selection probability for every member of the initial population by calculating the ratio of fitness value of that initial population to the sum of the fitness values of all the individuals.

**Step 4:** Select two members (parents) to be used for reproduction based on selection probability.

**Step 5:** Apply crossover, mutation, and inversion to the parents to obtain the offspring. Generated offspring form the new generation.

If the size of the new population is equal to the initial population size, go to step 6, else go to step 4.

**Step 6:** If the current generation is equal to the maximum number of the generation then stop, else move to step 2.



**Fig1.3: Steps of genetic algorithm**

## **Operators in Genetic Algorithm**

The basic operators that are to be discussed in this section are: encoding, selection, crossover and mutation operators.

### **Encoding**

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, list or any other objects. The encoding depends mainly on solving the problem. For example, one can encode directly real or integer numbers.

#### **1. Binary encoding**

The most common way of encoding is binary string. Each chromosome encodes a binary string. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. Another possibility is that the whole string can represent a number. The way bit strings can encode differs from problem to problem.

#### **2. Octal encoding**

This encoding uses string made up of octal numbers (0-7)

#### **3. Hexadecimal Encoding**

This encoding uses string made up of hexadecimal numbers (0-9.A-F).

#### **4. Permutation Encoding (Real number coding)**

Every chromosome is a string of numbers, represented in a sequence. Sometimes corrections have to be done after genetic operation is complete. In permutation encoding every chromosome is a string of integer/real values, which represent number in a sequence.

This encoding is only useful for ordering problems as in this project. Even for this problem some type of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it).

#### **5. Value Encoding**

Every chromosome is a string of values and the values can be anything connected to the problem. This encoding produces best results for some special problems. On the other hand, it is often necessary to develop new genetic operator's specific to the problem. Direct value encoding can be used in problems, where some complicated values, such as real numbers are used.

#### **6. Tree Encoding**

This encoding is mainly used for evolving program expressions for genetic programming.

## **Selection**

Selection is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection i.e. how to choose individuals in the population that will create the offspring for the next generation. The purpose of selection is to emphasise fitter individuals in the population in hope that their offspring have higher fitness. Chromosomes selected from the initial population to be parents for reproduction.

Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function the better chance that an individual will be selected. The selection pressure is defined as the degree to which the individuals are favoured. The selection pressure drives the GA to improve the population fitness over successive generations.

The convergence state of GA is largely determined by the magnitude of the selection pressure, with higher selection pressures resulting in higher convergence rate. GA should be able to identify optimal or nearly optimal solution under a wide range of selection scheme pressure. However if the selection pressure is too low, the convergence rate will be slow, and the GA will take unnecessarily longer to find the optimal solution. If the selection pressure is too high, there is an increased change of the GA prematurely converging to an incorrect solution. In addition to providing selection pressure, selection schemes should also preserve the population diversity as this helps to avoid premature convergence.

## **Roulette Wheel Selection**

Roulette selection is one of the traditional GA selection techniques. The principle of roulette selection is the linear search through a Roulette wheel with the slots in the wheel weighed in the proportion to the individual's fitness value. The expected value of an individual is individual's fitness divided by the actual fitness of the population. Each individual is assigned a slice of Roulette wheel, the size of the slice being proportional to the individual's fitness.

## **Random Selection**

This technique randomly selects a parent from the population. In terms of disruption of genetic code, random selection is a little more disruptive on average, than Roulette wheel selection.

## **Rank Selection**

The Roulette wheel will have a problem when the fitness values differ very much. If the best chromosome fitness is 90%, its circumference occupies 90% of the Roulette wheel and then other chromosomes have too few chances of getting selected. Rank selection ranks the population and every chromosome receives fitness from the ranking. The worst has fitness 1 and the best has fitness N.



## **Tournament Selection**

The best individual from the tournament is the one with the highest fitness, who is the winner of Nu.Tournament competitions and the winner are then inserted in the mating pool. The tournament competition is repeated until the mating pool for generating new offspring is filled.

## **Crossover (recombination)**

Crossover is the process of taking two parent solutions and producing from them a child. After the selection process, the population is enriched with better individuals. Crossover operator is applied to the meeting point with the hope that it creates a better offspring.

Crossover is a recombination operator that proceeds in three steps:

1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.

Finally, the position values are swapped between the two strings following the cross site.

## **Single point crossover**

The traditional GA uses single-point crossover, where the two mating chromosomes are cut once at corresponding points and sections after the cuts exchanged Here, a cross site or crossover point is selected randomly along the length of the mated strings and bits next to the cross sites are exchanged. If appropriate site is chosen, better children can be obtained by combining good parents, else it severely hampers string quality.

## **Two point crossover**

It should be noted that adding further crossover points reduces the performance of the GA. However, an advantage of having more cross over points is that the problem space maybe searched more thoroughly.

In two point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents.

## **Multi point crossover (N point crossover)**

There are two ways in this crossover. One is even number of cross sites and the other odd number of cross sites. In the case of even number of cross sites, the cross sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross sites, different cross point is always assumed at the string beginning.

### **Uniform crossover**

It is quite different from the N point crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length of the chromosomes. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a zero in the mask the gene is copied from the second parent.

### **Three parent crossover**

In this technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring, otherwise, the bit from the third parent is taken for the offspring.

### **Mutation**

After crossover, the string s are subjected to mutation. The mutation prevents the algorithm to be trapped in the local minimum. Mutation plays the role of recovering the lost genetic material as well as for randomly distributing genetic information. If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for the exploration of the whole search space. Mutation is viewed as a background operator to maintain genetic diversity in the population. It introduces new genetic structures in the population by randomly modifying some of its building blocks. It also keeps the gene pool well stocked, thus ensuring ergodicity. A search space is said to be ergodic if there is a non-zero probability of generating any solution from any population state.

It is also possible to implement kind of hill climbing mutation operators that do mutation only if it improves the quality of the solution. Such an operator can accelerate the search; however, care should be taken, because it might also reduce the diversity in the population and make the algorithm to converge towards some local optima.

### **Flipping**

Flipping of a bit involves changing 1 to 0 and 0 to 1 based on a mutation chromosome generated. A parent is considered and a mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in the parent chromosome is flipped and the child chromosome is produced.

## **Interchanging**

Two random positions of the strings are chosen and the bits corresponding to those positions are interchanged.

## **Reversing**

A random position is chosen and the bits next to that position are reversed and child is produced.

## **Other problem solving techniques**

With the rise of artificial life computing and the development of heuristic methods, other computerised problem solving techniques have emerged that are in some ways similar to genetic algorithms. This section explains some of these techniques, in what ways they resemble GAs and in what ways they differ.

## **Neural Network**

A neural network, or neural net for short, is a problem solving method based on a computer model of how neurons are connected to the brain. A neural network consists of layers of processing units called nodes joined by directional links: one input layer, one output layer, and zero or more hidden layers in between. An initial pattern of input is presented to the input of the neural network, and nodes that are stimulated then transmit a signal to the nodes of the next layer to which they are connected. If the sum of all inputs entering one of these virtual neurons is higher than that neuron's so called activation threshold, that neuron itself activates, and passes on its own signal to neuron in the next layer. The pattern of activation therefore spreads forward until it reaches the output layer and is there returned as a solution to the presented input. Just as in the nervous system of the biological organisms, neural networks learn and fine-tune their performance over time via repeated rounds of adjusting their thresholds until the actual output matches the desired output for any given input. This process can be supervised by a human experimenter or may run automatically using a learning algorithm. Genetic algorithm have been used both to build and to train neural networks.

## **Hill Climbing**

Similar to genetic algorithms, though more systematic and less random, a hill-climbing algorithm begins with one initial solution to the problem at hand, usually chosen at random. The string is then mutated, and if the mutation results in higher fitness for the new solution than for the previous one the new solution is kept otherwise, the current solution is retained. The algorithm is then repeated until no mutation can be found that causes an increase in the current solution's fitness, and this solution is returned as a result. (To understand where the name of this technique comes from, imagine that space of all possible solutions to a given problem is represented as

three-dimensional contour landscape. A given set of coordinates in the landscape represents one particular solution. Those solutions that are better are higher in altitude, forming hills and peaks, those that are worse are lower in altitude forming valleys. A “hill-climber” is then an algorithm that starts out at a given point on the landscape and moves inexorably uphill.) Hill-climbing is what is known as a greedy algorithm, meaning it always makes best choice available at each step in the hope that the overall best result can be achieved this way. By contrast, methods such as GA and simulated annealing, discussed below are not greedy; these methods sometimes make sub-optimal choices in the hope that they will lead to better solutions later on.

## **Simulated Annealing**

Another optimisation technique similar to evolutionary algorithms is known as simulated annealing. The idea borrows its name from the industrial process of annealing in which a material is heated to above a critical point to soften it, then gradually cool in order to erase defects in its crystalline structure, producing a more stable and regular lattice arrangement of atoms. Simulated annealing, as in GA, there is a fitness function that defines a fitness landscape, however, rather than a population of candidates as in GAs, there is only one candidate solution. Simulated annealing also adds the concept of “temperature”, a global numerical quantity which gradually decreases overtime. At each step of the algorithm, the solution mutates (which is equivalent to moving to an adjacent point of the fitness landscape). The fitness of the new solution is then compared to the fitness of the previous solution, if it is higher, the new solution is kept. Otherwise the algorithm makes a decision whether to keep or discard it based on temperature. If the temperature is high, as it is initially, even changes that cause significant increase in fitness may be kept and used as the basis for the next round of the algorithm, but as temperature decreases, the algorithm becomes more and more inclined to only accept fitness-increasing changes. Finally, the temperature reaches zero and the system “freezes”; whatever configuration it is in at that point becomes the solution. Simulated annealing is often used for engineering design applications such as determining the physical layout of components on a computer chip.

## **Tabu Search**

Tabu search is the meta heuristic search method which uses local search techniques used for mathematical optimisation. Local (neighbourhood) searches pick up a solution to a problem which is potential and check its immediate neighbours i.e. those solutions which are similar except for some minor details, with an intention of finding a better solution. Local search methods tend to get stuck in suboptimal regions or on plateaus where many solutions are equally fit. Tabu search uses memory structures to describe visited solutions or user provided rules in order to enhance the performance of these local techniques. If any solution which is potential has been previously visited within a short duration or if it doesn't satisfy a rule, then it is marked as “tabu”. By doing so the algorithm doesn't consider that possibility repeated.

Local search procedures often become stuck in poor scoring areas or areas where scores plateau. Tabu search carefully explores the neighbourhood of the potential solutions as the search progresses so that these pitfalls can be avoided and to explore those regions of the search space that would be left unexplored by other local search procedures.

Application areas such as resource planning, telecommunication, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis and many others employ tabu search. Tabu search is useful as the solutions obtained are significantly of higher quality than those obtained by methods previously applied.

### **Priority rules**

Priority rules contain information used to construct the list of activities that ranks all the activities of a project in some order such as to determine the priorities in which the activities are assigned in the project schedule. Such a list of the priority is constructed based on the project data in order to design priorities. The project data consists of:

- Activity information: time and cost estimate information of the activities determine the activity priorities.
- Network information: project network logic information determines the activity priorities.
- Scheduling information: activity priorities are obtained by the information of the critical path scheduling tools.

Resource information: the information about the resources of the project determines the project activity.

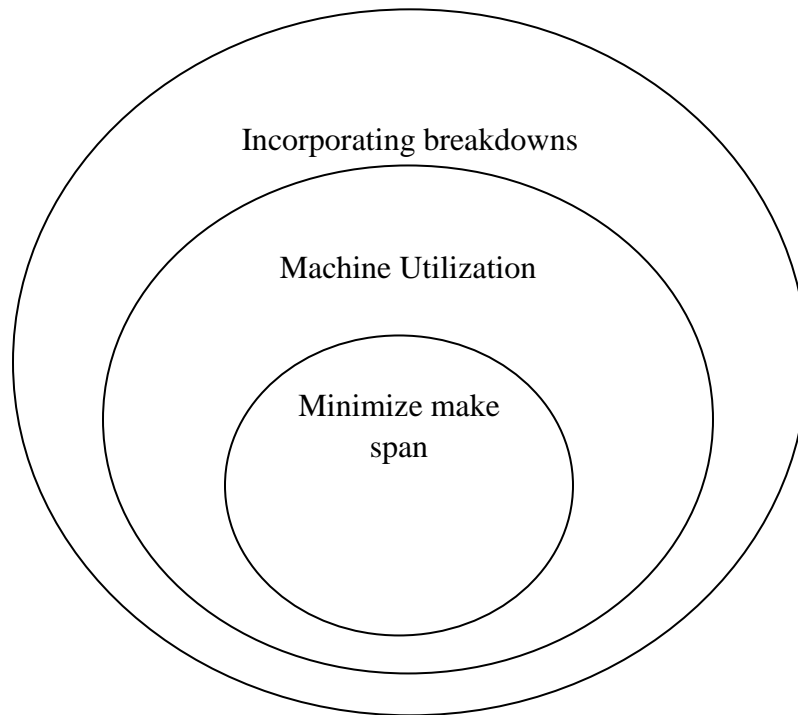
The constructed list is then used and activities are removed one by one from the list and are put in the schedule during the heuristic scheduling process. To that purpose, a schedule generation scheme is used to determine the exact start and finish time of the activity.

## **1.2 Objective**

Modern fms although seems very accurate theoretically but when applied to real life scenario fail considerably due to the fact that they have ignored the factors of DELAY involved in production process. Our objective through this project is to bridge this gap and make the solution suitable for real life application and show how total load distribution equally in each machine, minimize the makespan and machine utilization.

- Determination of a schedule with the objective of minimizing the total make span time with actual breakdown.
- To generate a schedule using Genetic Algorithm
- To reduce set up and queue times

- Improve efficiency
- Reduce time for product completion
- Utilize human workers better
- Produce more product more quickly
- To make the scheduling more robust and stable



**Fig1.4: Objectives**

## 2. Literature Review

To gain a better insight into the FMS scheduling problem solution space a comparative review of the various algorithms, techniques and approaches has been done. The comparison was carried out while keeping in mind various parameters such as selection, mutation, crossover, fitness criteria and solution space. After in- depth analysis a conclusion can be inferred that there is a trade-off between optimality of solutions and computation time required. Since JSSP is a NP complete problem, there exists no specific algorithm which would give the optimal solution for all cases.

[9] Presents a new GA based Discrete Dynamic Programming approach for generating static schedules in a FMS environment. It adopts a sequence dependent schedule generation strategies where to generate feasible job sequences, GA is employed and a series of DDP are constructed to generate schedules. The author stated that DDP is capable of identifying locally optimised partial schedules. The proposed algorithm does not suffer from state explosion problem. The author concluded that the algorithm was successfully implemented for generation of schedules.

In [5] three objectives i.e. minimise makespan, minimise maximal machine workload and minimum total workload are addressed and a hybrid genetic algorithm is proposed for a problem. They have used two vectors to represent the solution also two operators i.e. advance crossover and mutation operator are used to adapt special chromosome structure and varying characteristics of the problem. Authors have improved the genetic algorithm by variable neighbourhood descent in order to strengthen the search ability. The approach used involved two local search procedure; local search of moving one operation, local search of moving two operations. They have developed an efficient method for finding assignable time intervals for the deleted operations based on the earliest and latest event times. To unify the operation sequence in chromosome with the sequence in the decoded schedule or recorded procedure is used which facilitates genetic operators to pass from the good traits of the parents.

Liang Sun et al in their work [4] proposed a genetic algorithm with the penalty function for the FMS scheduling problem. For this, they used a clonal selection based hyper mutation and a lifespan extended strategy. During a search process, an adaptive penalty function is decided so that the algorithm can search in both feasible and infeasible solution of the solution space. They conducted experiments on 23 benchmark and instances of the OR-Library. The proposed algorithm effectively exploits the capabilities of distributed and parallel computing of swarm intelligence approaches and effectively makes use of the famous scheme theorem and the building block hypothesis of Holland. The results indicate the successful incorporation of the proposed operators.

In paper [3] M. Heydar solved the FMS scheduling problem considering two objectives i.e. maximum completion time (makespan) and maximum tardiness. This scheduling problem is stated belonging to the class of NP hard problems and hence no exact method is appropriate to solve the practical cases of scheduling problems. They proposed a hybrid genetic algorithm

combined with four priority dispatching rules. The proposed approach resulted in performing well in efficiency and quality of solutions.

In paper [6] Christian Bierwirth proposed a new representation technique mathematically known as permutation with repetition in order to sequence the operations of an FMS on a number of machines related to the technological machine order of jobs. A new crossover operator is designed with similar behaviour to the Order-Crossover for simple permutation schemes which reserves the initial scheme structure of permutations with repetitions. Together, the new representation and Generalisation of OX facilitates genetic search for scheduling problem. The author concluded that the representation demonstrates sustained performance on a platform of difficult benchmark problems and the algorithm attains a quality level of optimisation within the range of other GA approaches.

In this paper [7] Nidhish Mathew Nidhiry, stated that the main characteristics of FMS is simultaneous execution of processes and sharing a finite set of resources. In this paper, a genetic algorithm based scheduling of FMS is presented with the objectives; minimising idle time of the machine and minimising the total penalty cost for not meeting the deadline. A software is also developed for getting optimal sequence of operation. The FMS under study has 16 CNC machine tools for processing 43 varieties of product. Various meta heuristic method are used for solving the same scheduling problems. Authors concluded the successful implementation of the procedure developed.

In this paper [8] a complex scheduling problem in flexible manufacturing system (FMS) has been addressed

With a novel approach called knowledge based genetic algorithm (KBGA). It was stated that meta heuristics may be used for combinatorial decision making problems. This approach combines knowledge based which uses expert knowledge and genetic algorithm for searching optimal solution. The approach has been applied on four different stages of genetic algorithm. The authors tested KBGA on 10 different moderate size of data. It has been concluded that in the proposed algorithm the operators also used with knowledge based to reduce computational obstacles.

In [2] Vijai Singh et al stated that achieving high performance for an FMS require good scheduling system which makes a right decision at the right time considering the system conditions which is difficult using traditional optimising techniques. Author has made use of genetic algorithm with roulette wheel based selection process. They concluded that roulette wheel selection process is fast and easy to implement and GA based scheduling has considerable potential application to FMS.

Swati Singh et al in their work[1] stated job shop scheduling problem as one of the most difficult NP hard combinatorial optimisation problem. Hence, optimal schedule determination is considered difficult. Their paper focussed on the objective of generating a schedule, minimising total makespan time using genetic algorithm. They concluded that JSSP may have made a schedule depending upon the number of jobs and machine. Genetic algorithm is preferred over traditional scheduling methods



## Result

**Table 2: The comparative analysis of the work studied is presented in the table below:**

	Objective	Selection	Crossover	Mutation	Future work
JSSP with delay constraints[1]	Improving solution to JSSP problem(i.e. minimising makespan)	Random	Single point crossover at point ceil $\lceil p/2 \rceil$  P=no.of piece/job  N=total jobs	Transposition  Mutation probability =0.5	1. Assumed no breakdown.  2. Considered jobs atomic.  3. Considered transportation and setup time constant.
FMS Scheduling using Roulette wheel selection[2]	Minimising the total makespan time	Roulette wheel selection	Single point crossover with fixed crossover probability in range $[1, L-1]$	Flipping	1. Assumed no breakdown.  2. Transportation and setup time constant
Multiobjective Hybrid GA method[3]	Minimise makespan and maximise tardiness	Roulette wheel approach	Job based crossover(JOX)	Operation swap mutation	Can be further extended by applying heuristic methods separately or in conjunction with HGA>
Penalty method for constraints in JSSP[4]	Minimise completion time for processing all jobs	Clonal selection	Normal	Inverse, interchange, insert	Proposed algo to more practical and integrated problem
HGA & variable neighbourhood descent algo for FJSSP(FMS)[5]	1.minimise makespan  2.minimise maximal machine workload  3.minimise	Ranking method	1.Order crossover for operation sequence vector.  2.Extended order	Allele based and immigration mutation	Moving 3 or more operation, the balance between genetic search and total

	total workload		crossover and uniform crossover for machine assignment vector		search
Generalised permutation approach to FMS with GA[6]	How to represent the problem in the algorithm	Permutation	Generalised order crossover(GOX)	-	-
Evaluation of GA approach for scheduling optimisation of FMS[7]	1.Minimising idle time of machines 2.Minimising total penalty cost	Tournament selection	Single point crossover	Flipping	Will include availability & handling times of load/unload stations, robots, AGVs.
FMS Scheduling with knowledge base GA approach[8]	Throughput & mean flow time have been taken to measure the performance of fms	Knowledge based 1.directional 2.steady 3.unruly	Knowledge based	Knowledge Based	Can be stretched out to various problems of FMS that cover the balancing or allocation of resources.
Discrete Dynamic Approach[9]	Minimising makespan	Random	Subtour chunking crossover	-	Generation of global optimal schedule
Generalised order crossover[10]	Minimising makespan	Neighbourhood mating and local offspring acceptance	GOX i.e. permutation with repetitions	Position based mutation	-
Hybrid GA using random keys[11]	Minimising makespan and maximising tardiness	Random and 10% direct	Parameterized uniform crossover	Replacing bottom 20% by random	Breakdowns not considered.

				generation	
--	--	--	--	------------	--

## 3. System Analysis

### 3.1 Platform Used

The entire program is being done on MATLAB R2010a. **MATLAB** is a numerical computing environment and fourth-generation programming language. Developed by MathsWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java and FORTRAN.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi- domain simulation and Model- Based Design for dynamic and embedded systems.

The input files are in form of spreadsheet. They are used because

- Simple to design
- Any low level data operator can also design them
- Easy to modify
- Easy to understand
- Reduces the complexity

The output is shown in form of bar charts, pie charts and Gantt charts which can be directly made by Matlab.

### 3.2 Feasibility Study

Feasibility studies aim to analysis and evaluation of a proposed project to determine if it is (1) is technically feasible, (2) is feasible within the estimated cost and (3) will be profitable. Feasibility studies are almost always conducted where large sums are at stake. Also called feasibility analysis.

#### 3.3.1 Technical feasibility:

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating etc. in order to estimate whether the new system will perform adequately or not. Technological feasibility is carried out to determine whether the company has the capability of the project for proving our project technically feasible we run our project 1000 itr for some data machine utilization shown as in fig 1 but when machine utilization shown as in fig 2 and much better run 10000 times.

### **3.3.2 Economic Feasibility**

Economic feasibility determines the benefits and saving that are expected from the system and compare them with costs. Cost/Benefit analysis has been done on the basis of total cost of the system and direct and indirect benefits derived from the system.

The total cost for the proposed system comprises of hardware costs and software costs. The main aim of economic feasibility is to check whether the system is financially affordable or not. The cost for the proposed system can be divided into two parts given below:

#### **Hardware Costs**

The hardware cost for the proposed system can be calculated from cost of hardware needed for the development of the proposed system. The hardware specifications for the system are given below:

#### **Personal Computer**

The cost of the PC depends upon the configuration of the PC. The minimum specification assumed for the Pc is given below:

- Pentium IV processor
- 128 MB RAM
- 40 GB free disk space
- MS Windows

#### **Software Costs:**

The Software costs for the proposed system can be calculated from the cost of software tools needed from the development of the proposed system. The software tools needed for the development of the system are given below:-

- Matlab

### **3.3.3 Operational feasibility**

Behavioral feasibility estimates the reaction of the User staff towards the development of the computerized system. For the successful implementation of any system, the users must be impressed that the new system is for his benefit. So, the behavioural feasibility plays a very important role in the development of new system. It reveals that whether the system is acceptable by user or not. If the user does not read to use it, then it doesn't matter how best the system is or how much effort you are putting in its development.

## **4. System Specification**

### **4.1 Software requirements:**

- Operating system
  - Windows XP or higher version
- Graphical User Interface:
  - MATLAB version 10 and above

### **4.2 Hardware requirements:**

- Processor Dual Core
- 1 GB RAM
- Secondary storage
  - Hard Disk (320 GB)

## **5. Software Description**

### **5.1 Front End**

The front end comprises of a graphical user interface which has been designed to facilitate the user to select the values for the input parameters. It allows the user to browse and select the input file. The other parameter values- mutation rate, crossover rate and number of iterations-can be selected from a range of values provided in a drop-down list. It also provides an option to choose whether the user wishes to operate in a breakdown environment or in an environment without breakdowns.

#### **5.5.1 Features**

- User friendly
- Easy to use
- Provides clarity of purpose of inputs
- Easy to understand
- Allows user to input values completely of his choice
- Generic in purpose

### **5.2 Back End**

The backend comprises of the implementation of the proposed in MATLAB. A functional based approach has been used to implement various functionalities. The entire computation has been broken down into smaller components each coded into a separate module. The modules are independent at the same time there are various interlunation call procedures.

#### **5.2.1 Features**

- The code is readable
- Fragmentation of the functionality into sub processes makes it easy to understand
- Bottom up approach
- Modularity in operation is adopted
- Compact, portable across different versions of platform.

## 6. Project Description

### 6.1 Problem Definition

Job scheduling on various machines of varied capabilities has been a tedious task for any organization. Since the ability and functions of modern machines have been widely extended, the scheduling plan of a part might not be unique. So on a real shop floor, many feasible schedules can be found. The increased flexibility of an FMS can require a more flexible scheduling system. The aim of the project is to provide a solution for the problem taking into consideration actual breakdowns in small and medium scale industries by building a scheduling tool

### 6.2 Overview of the Project

#### Input Used

##### What is taken as input?

- Number of objects to be produced (n).
- Number of available machines (m).
- Number of pieces per object (p).
- A time matrix of size  $(n*m)[M]$ 
  - Depicting time each object takes to be completed on each machine.
  - a time '0' is assigned if a machine is incapable of producing an object.
- A 'Delay Matrix' of size  $(n*m)$  [D] depicting real time delays
  - Buffer load/unload time
  - Setup time

##### How to take input?

By a formatted spread sheet.

-since the input is quite bulky data, hence separate bones would be tedious.

GAs are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetic. As such they represent an intelligent exploitation of a random search used to solve optimisation problems. They exploit historical information to direct the search into the region of better performance within the search space.

GA belong to the class of evolutionary algorithms. John Holland based GA on the basic Darwinian Mechanism of "Survival of the fittest" [12].

*Principle:* The principle of GA is simple, imitate genetics and natural selection by a computer program. The parameters of the problem are coded most naturally as DNA. A set called population, of these problem dependent parameter value vector is processed by GA.



They are better than conventional algorithms in that they are more robust. They do not break easily even if the inputs are changed slightly or in the presence of reasonable noise.

A typical GA process consist of following steps

**Step 1:** Generate the initial population. Determine the size of the population and the maximum number of the generation.

**Step 2:** Calculate the fitness value of each member of the initial population.

**Step 3:** Calculate the selection probability of each member of the initial population using the ratio of fitness value of that initial population to the summation of the fitness values of the individual solutions.

**Step 4:** Select a pair of members (parents) that can be used for reproduction using selection probability.

**Step 5:** Apply the genetic operators such as crossover, mutation, and inversion to the parents. Replace the parents with the new offspring to form a new population.

Check the size of the new population. If it is equal to the initial population size, then go to step 6, otherwise go to step 4.

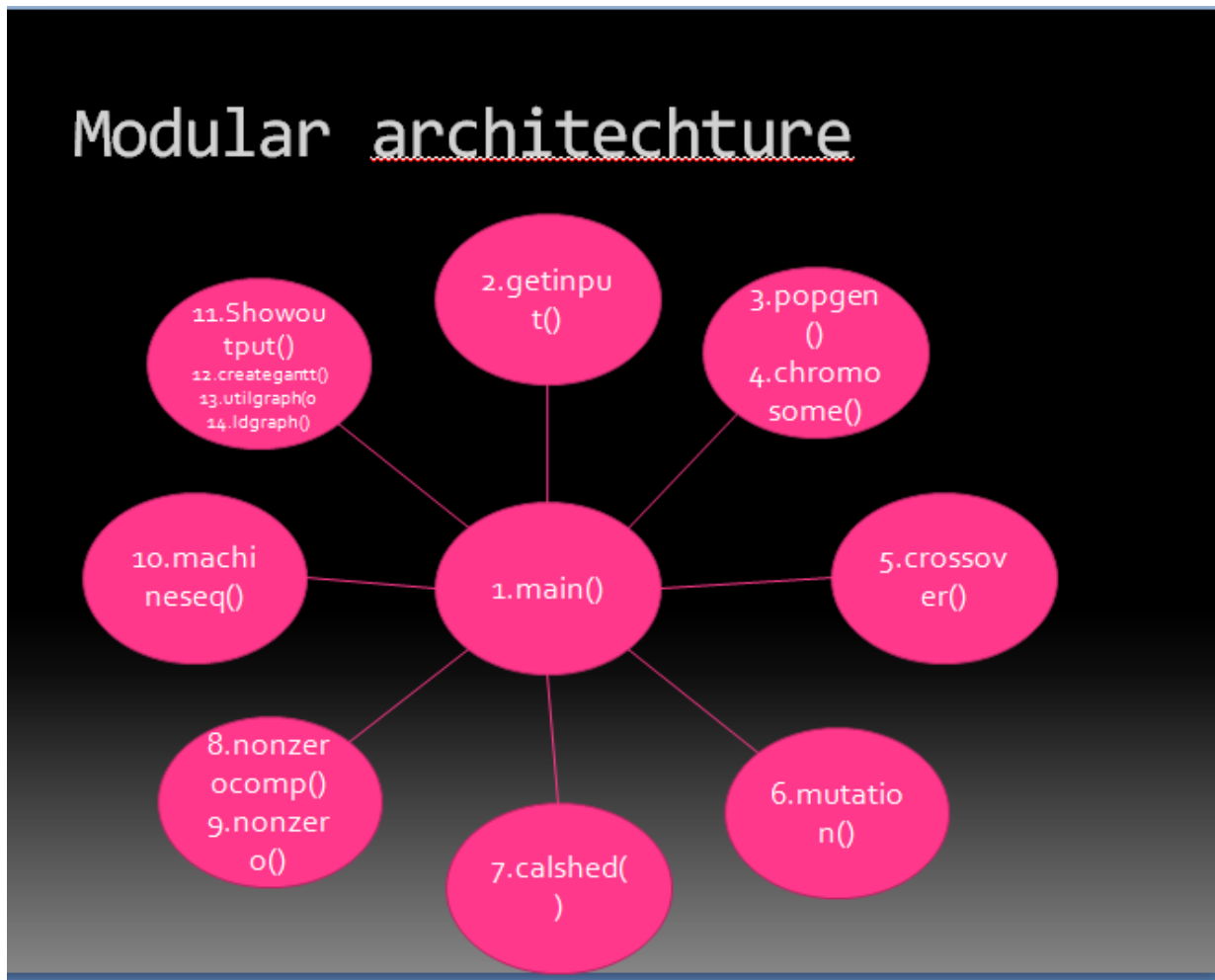
**Step 6:** If the current generation is equal to the maximum number of the generation then stop, else move to step 2.

## OUTPUT

- Optimal Gantt Chart
- Optimal Schedule
- Machine Utilization Graph
- Load Distribution Pie Chart

## 6.3 Module description

### Modular Architecture



**Fig6.3.1:Modular Architecture**

### 6.3.1 Modules

#### 1. Main Program

It is the driver or caller of the project which call all other function and responsible for taking input, processing and providing output.

#### 2. Getinput()

Getinput() take user input from spreadsheet which user has specified  
Input argument- none

Called by- main

Output/return- none

### 3. **Popgen()**

Used to generate random population of size 10.

Called by- main

Input argument- none

Return- random pop store in 'chrom'

Call- chromosome()

### 4. **Chromosome()**

Used to generate random chromosome of size  $n \times p$ .

Called by- popgen()

Input argument- none

Output/ return- a chromosome c of size  $n \times p$

Used built in foundation randperm(n)

### 5. **Crossover(chrom)**

Select two random chromosome and perform the crossover and return a child chromosome c.

Called by- main()

Input- population

Return- c

### 6. **mutation(c)**

Used for mutating a chromosome randomly

Input argument- chromosome c

Output/ return- mutated chromosome

Called by- main

### 7. **calcsched(c)**

Used to calculate the schedule in the matrix M1 with the help of chromosome c. The dimension of chromosome M1 is  $(m \times np)$

Input argument- chromosome c

Return- the matrix M1 which the schedule.

### 8. **Machineseq(machseq,matrix,chromo)**

Machineseq() is used to create job sequence as execute per machine

Called by- main()

Input argument- machseq{array filled with zeros of size  $(m \times (n \times p / m))$ }

Matrix- array sequence corresponding to M1

Chromo- final chromosome sequence.  
Output/ return- job sequence per machine.

**9. Create gantt(Matrix)**

Used to create gantt chart of final schedule.

**10. Utilgraph(mutil)**

Used to create bar graph according to % machine utilization.

**11. Nonzero(k,M1)**

Function used to calculate number of zero element i.e. kth row of matrix M1.

Called by- calcshcd(M1)

Input- k(row number)

M1(schedule matrix)

Return- number of non-zero element.

**12. Nonzerocomp(M1)**

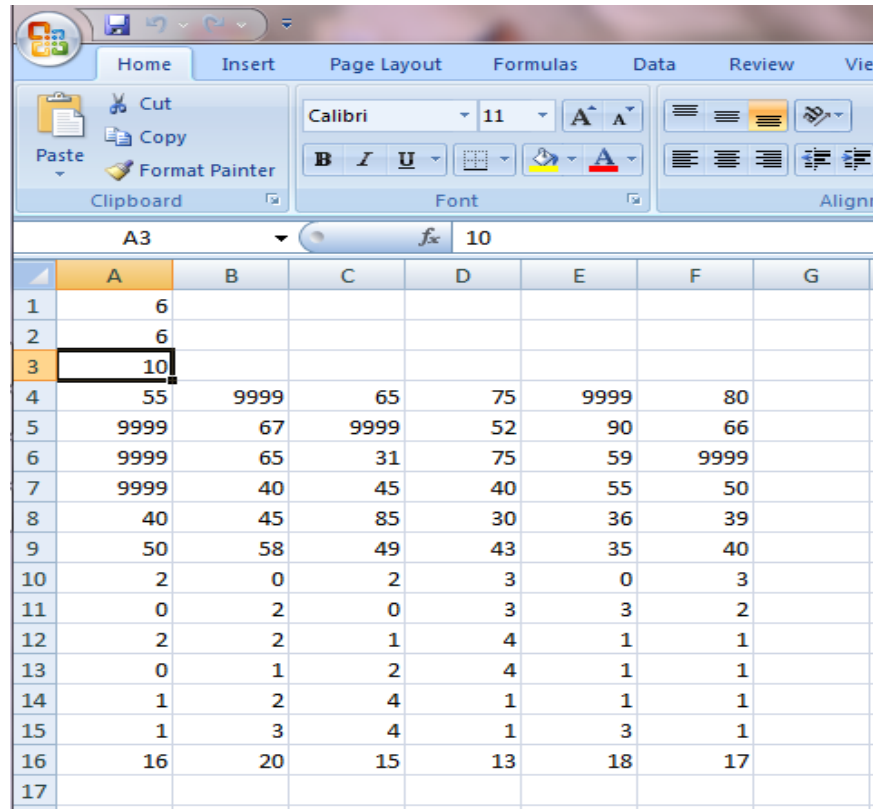
Function used to calculate number of non zero element in M1

Called by- main()

Input- matrix M1

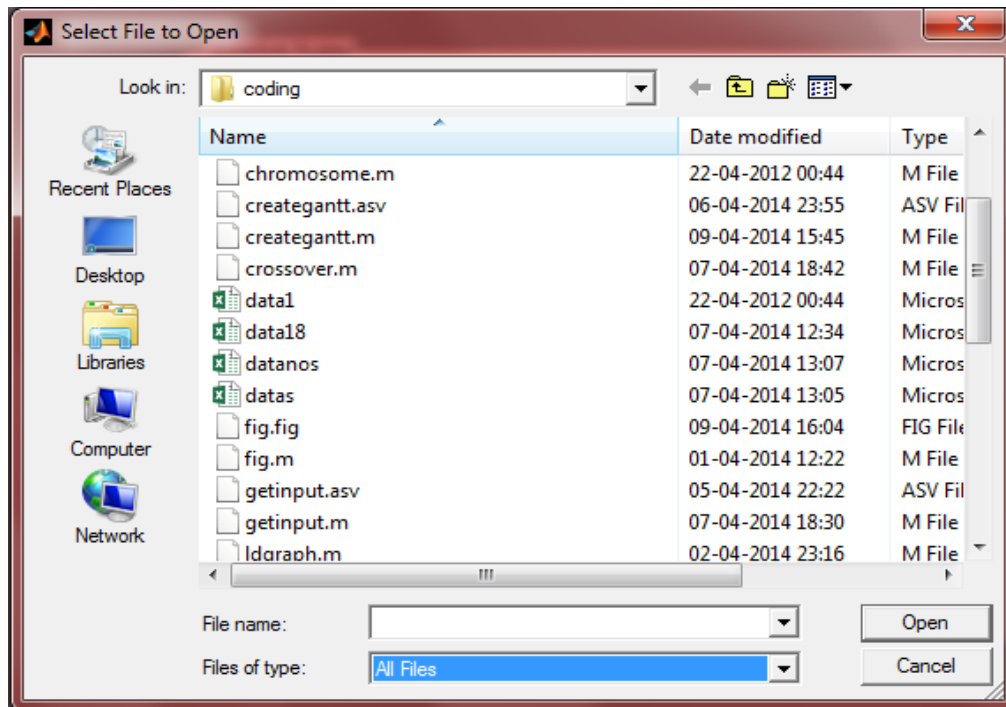
Return- number of non zero elements.

## 6.4 Input design



	A	B	C	D	E	F	G
1	6						
2	6						
3	10						
4	55	9999	65	75	9999	80	
5	9999	67	9999	52	90	66	
6	9999	65	31	75	59	9999	
7	9999	40	45	40	55	50	
8	40	45	85	30	36	39	
9	50	58	49	43	35	40	
10	2	0	2	3	0	3	
11	0	2	0	3	3	2	
12	2	2	1	4	1	1	
13	0	1	2	4	1	1	
14	1	2	4	1	1	1	
15	1	3	4	1	3	1	
16	16	20	15	13	18	17	
17							

**Fig6.4.1:Input File**



**Fig6.4.2:Selection of the input file**

The above screen appears as the user clicks on the browse button. It lets the user to browse all the way through the directory path to the input file.

### Processing time

<b>55</b>	<b>9999</b>	<b>65</b>	<b>75</b>	<b>9999</b>	<b>80</b>
<b>9999</b>	<b>67</b>	<b>9999</b>	<b>52</b>	<b>90</b>	<b>66</b>
<b>9999</b>	<b>65</b>	<b>31</b>	<b>75</b>	<b>59</b>	<b>9999</b>
<b>9999</b>	<b>40</b>	<b>45</b>	<b>40</b>	<b>55</b>	<b>50</b>
<b>40</b>	<b>45</b>	<b>85</b>	<b>30</b>	<b>36</b>	<b>39</b>
<b>50</b>	<b>58</b>	<b>49</b>	<b>43</b>	<b>35</b>	<b>40</b>

### Delay time

<b>2</b>	<b>0</b>	<b>2</b>	<b>3</b>	<b>0</b>	<b>3</b>
<b>0</b>	<b>2</b>	<b>0</b>	<b>3</b>	<b>3</b>	<b>2</b>

<b>2</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>1</b>	<b>1</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>4</b>	<b>1</b>	<b>1</b>
<b>1</b>	<b>2</b>	<b>4</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>1</b>	<b>3</b>	<b>4</b>	<b>1</b>	<b>3</b>	<b>1</b>

### **Setup time**

<b>16</b>	<b>20</b>	<b>15</b>	<b>13</b>	<b>18</b>	<b>17</b>
-----------	-----------	-----------	-----------	-----------	-----------

A(1,1)= number of machines

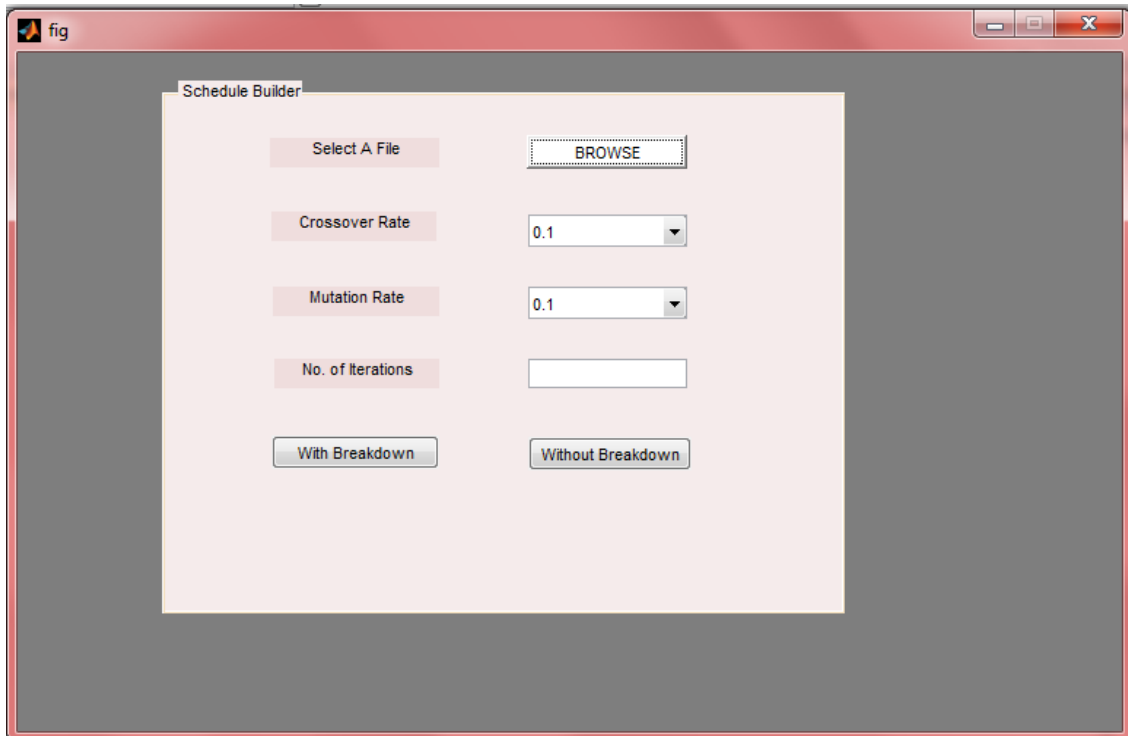
A(2,1)= number of jobs

A(3,1)= number of pieces per job

For rows 4 to 9, it is part processing time per minute.

For rows 10 to 15, it is delay time.

For row 11, it is setup time.



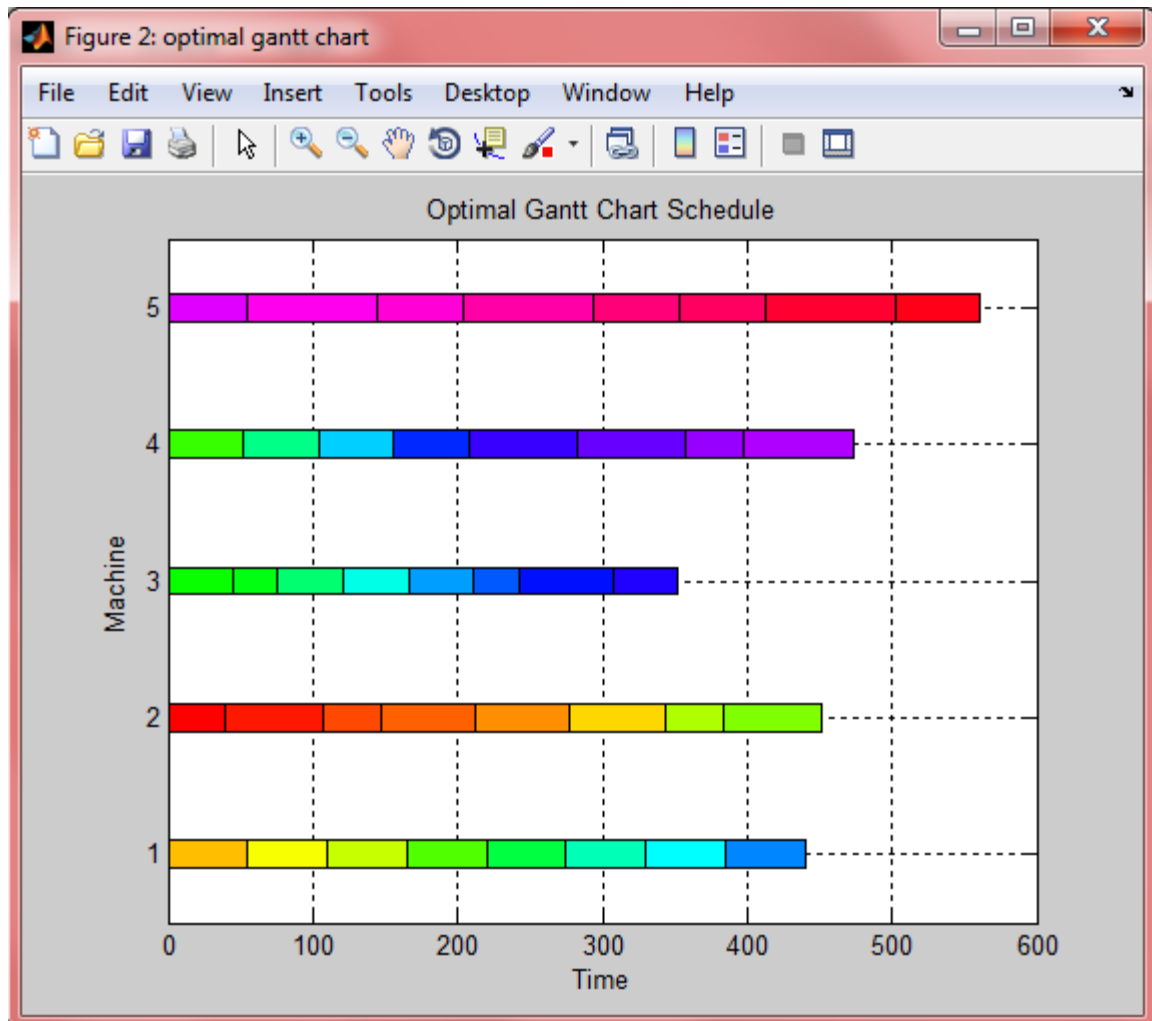
**Fig6.4.3: GUI**

Crossover Rate and Mutation Rate provides the number of choices from which the user can take any one depending upon the scenario.

The GUI provides two buttons with breakdown and without breakdown. Depending upon the scenario, user can choose any of them.

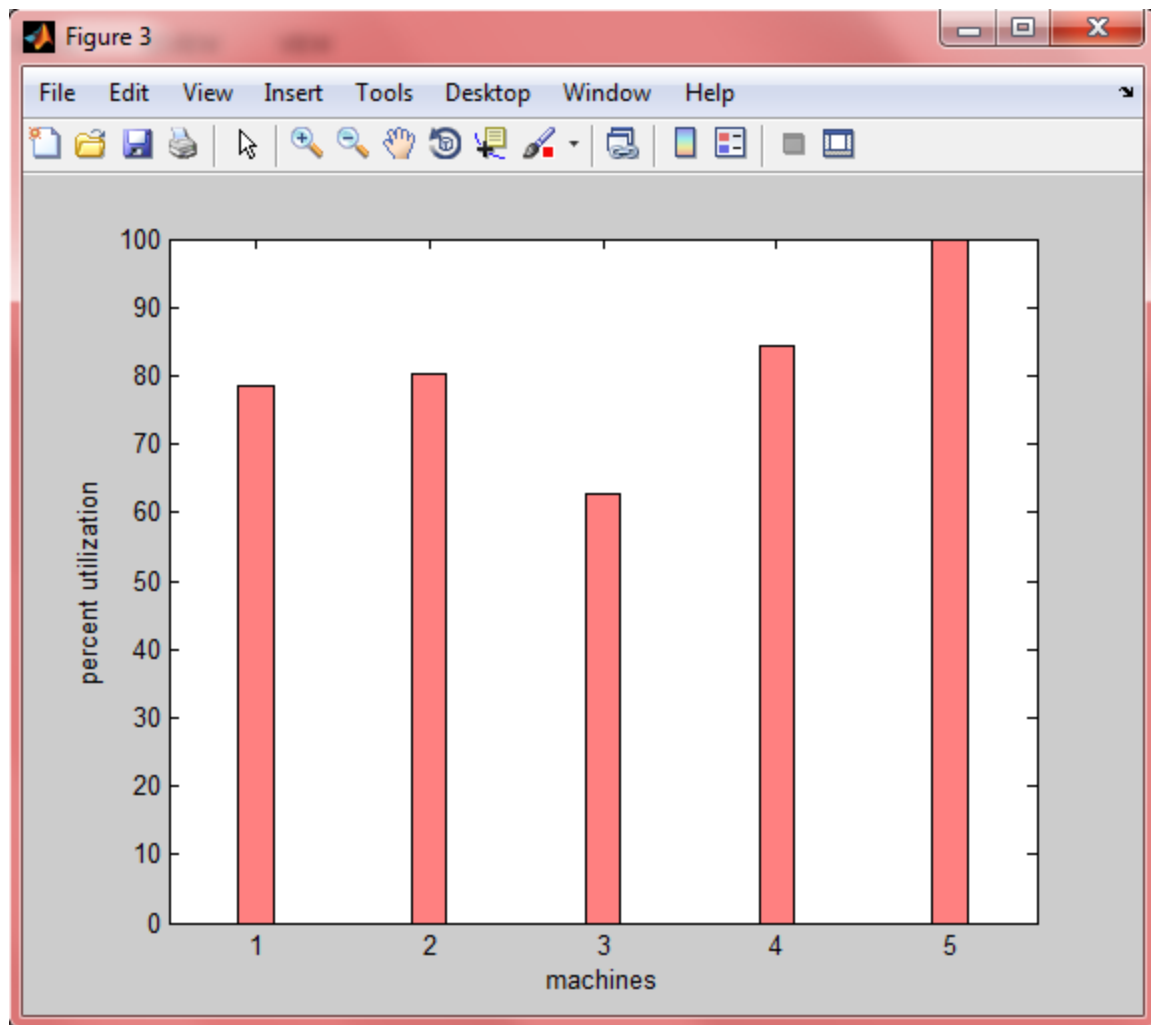


## 6.5 Output Design



**Fig6.5.1: Optimal Gantt chart**

**Function used: Creategantt(MATRIX)**

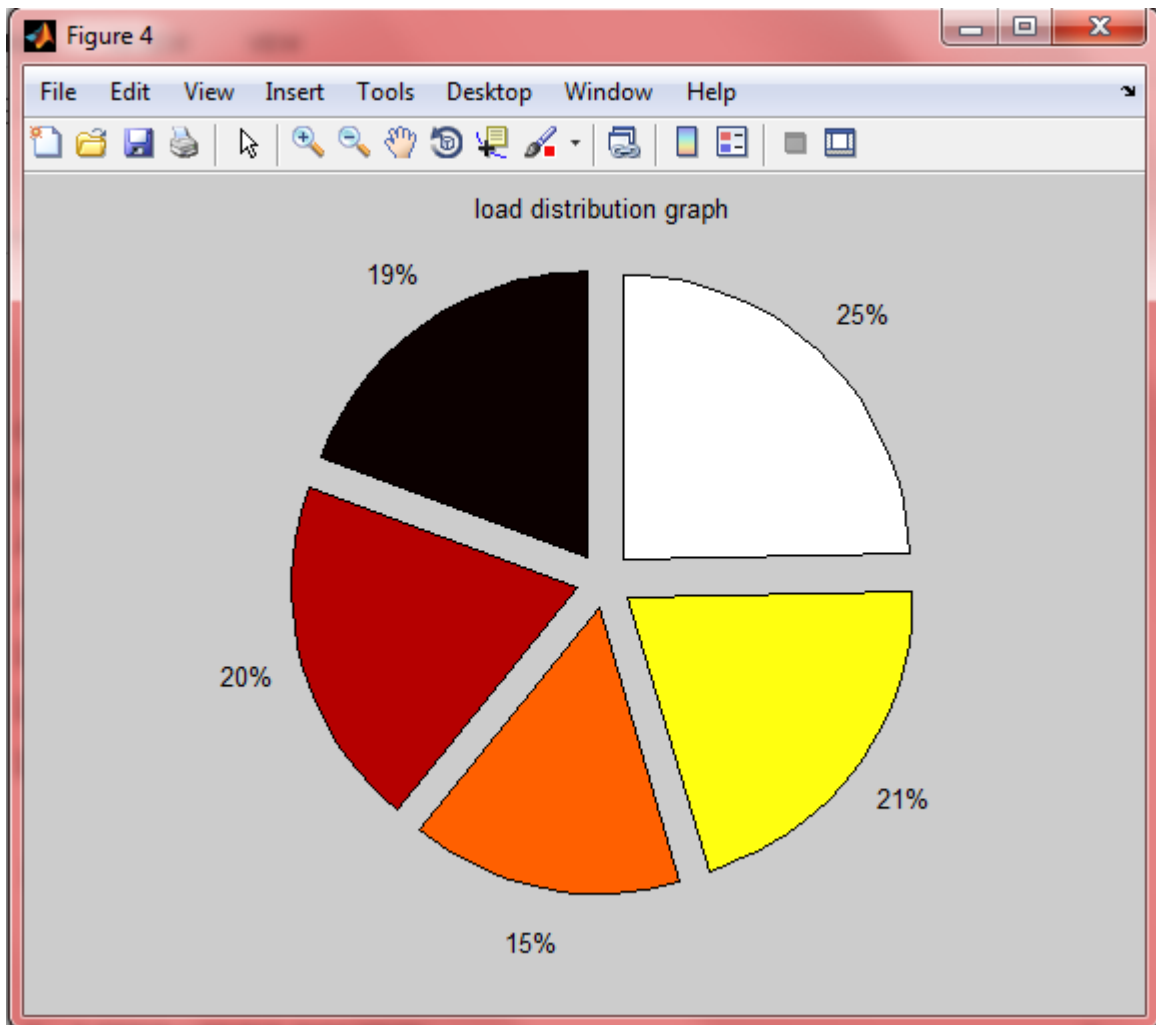


**Fig6.5.2: Load distribution graph**

**Formula Used**

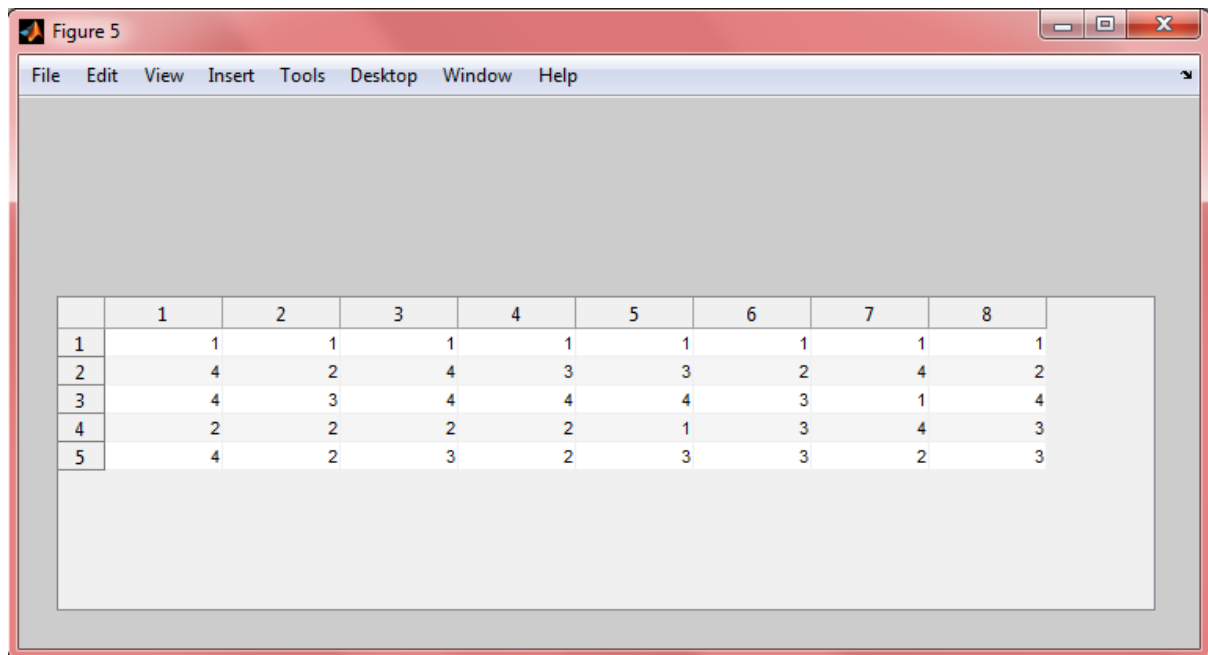
$$\text{mutil} = s * 100 / \text{makespan}$$

**Function used:** `ldgraph(mutil)`



**Fig6.5.3:Load distribution graph**

**Function used:utilgraph(mutil)**



**Fig6.5.4:Machine sequence**

**Function used:showoutput(MATRIX,mutil,machseq)**

## 7. System Implementation

Algorithm of implemented functions

Flexible manufacturing system can be clearly defined as:

- Combinatorial optimization problem
- NP complete in nature
- An nxmfms deals with scheduling of n jobs on m available machines
- Aim of the fms scheduler is to minimize the total '**makespan time**' i.e the total length of the schedule.

Since we have used genetic algorithms, we have implemented it on Matlab version 2010 platform.

**Given:**

- nxm matrix, where
  - n=number of jobs(which are atomic in nature)
  - m=number of available machines
  - $M[I\ j]$  is the matrix which contains the time required to complete [i] job on machine [j].
- P=contains the number of samples to be manufactured for each job

**Assumptions:**

- Equal number of products of each job is to be manufactured.
- Delay and setup time is considered to be constant.
- The time that a machine takes to process a job cannot be changed during a schedule generation process.

**Algorithm:**

Begin:

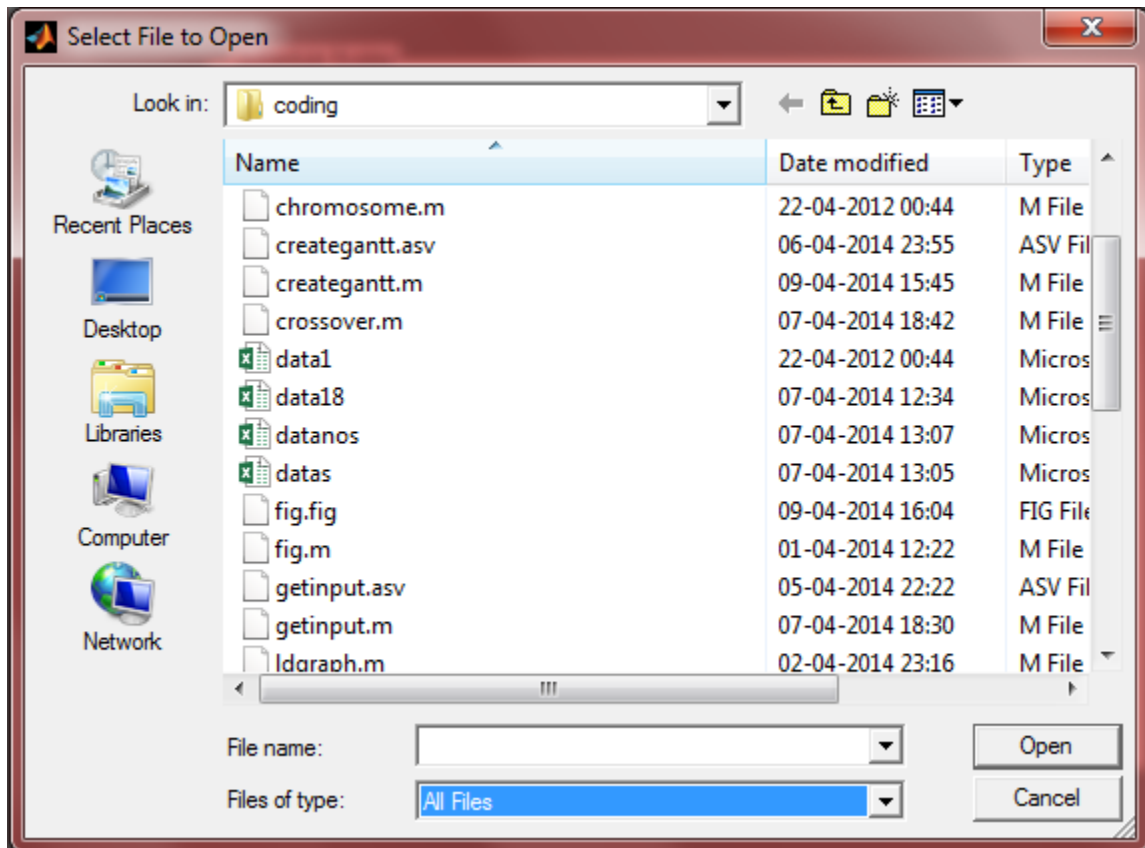
1. Initial population size =10  
Chromosome[C] of length =n x p, with random sequence of equal number of distributions.
2.  $M'[I\ j]$  of size  $m \times (n \times p)$  which would store the job sequence of each machine.
3. Ceil  $((n \times p)/m)$  number of jobs per machine can be scheduled
4.  $M'[I\ j]$  contains the feasible time of processing C jth element of chromosome i.e. the job number on the ith machine.
5.  $Sum[m]$ =sum of total processing time of each row of  $M'$ , i.e. for each machine.
6. Select the largest from  $sum[m]$ .

7. Select the largest value from  $\text{sum}[m]$  and compare with the MS assumed and copy the smaller value in MS (MS is the assumed makespan).
8. The sequence of machine operations corresponding to the makespan value is the optimised machine sequence.

The screenshot displays a graphical user interface titled "Schedule Builder". It features a light pink background with several input elements arranged in a structured layout. At the top left, there is a label "Select A File" next to a "BROWSE" button. Below this, the "Crossover Rate" is set to "0.1" in a dropdown menu, and the "Mutation Rate" is also set to "0.1" in a dropdown menu. The "No. of Iterations" is represented by an empty text input field. At the bottom, there are two buttons: "With Breakdown" and "Without Breakdown", both of which are currently selected or highlighted.

**Fig7.1: Front End**

The front end is in the form of a graphical user interface that lets the user browse and select the input file. It also allows the user to input the values for various parameters such as the mutation rate, crossover rate and the number of iterations. The input GUI also allows the user to choose from the two alternatives i.e. whether the breakdown has to be considered or not.



**Fig7.2: Browsing the input file**

The above screen appears as the user clicks on the browse button. It lets the user to browse all the way through the directory path to the input file.

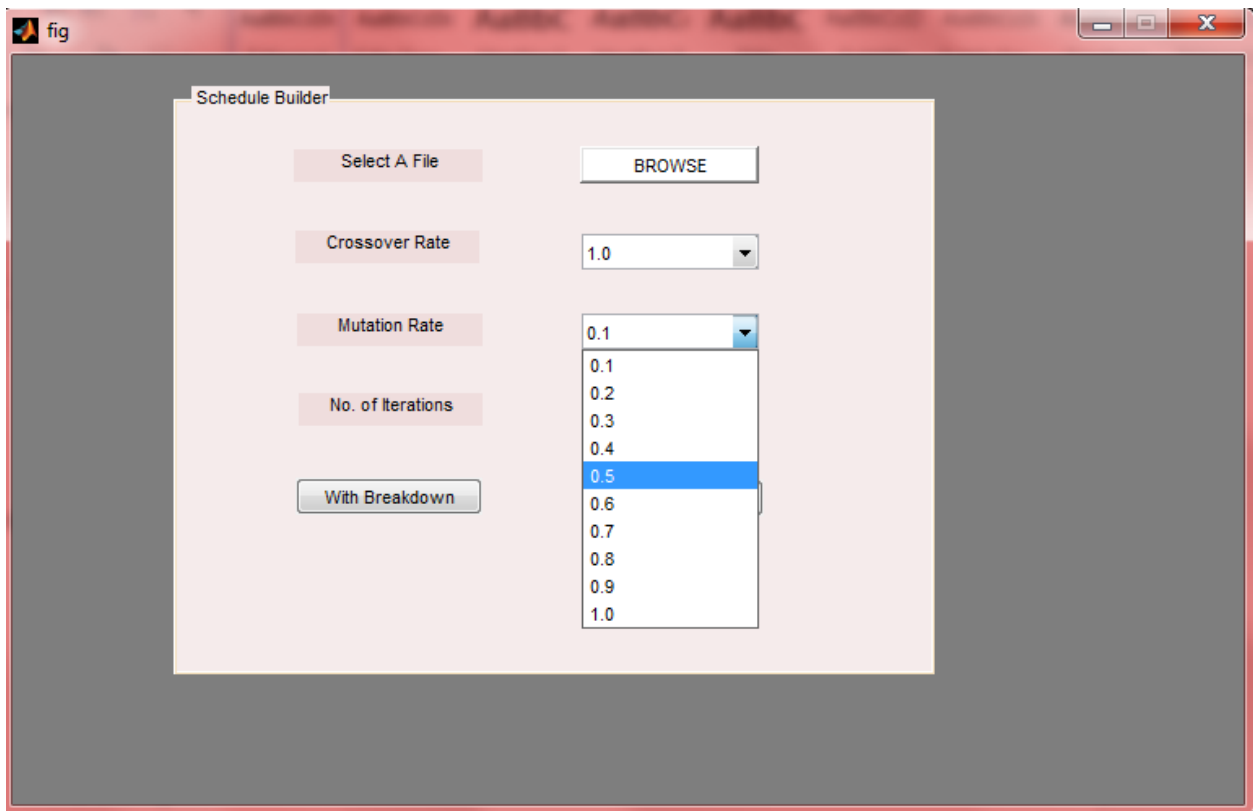
	A	B	C	D	E	F	G
1	6						
2	6						
3	10						
4	55	9999	65	75	9999	80	
5	9999	67	9999	52	90	66	
6	9999	65	31	75	59	9999	
7	9999	40	45	40	55	50	
8	40	45	85	30	36	39	
9	50	58	49	43	35	40	
10	2	0	2	3	0	3	
11	0	2	0	3	3	2	
12	2	2	1	4	1	1	
13	0	1	2	4	1	1	
14	1	2	4	1	1	1	
15	1	3	4	1	3	1	
16	16	20	15	13	18	17	
17							

**Fig7.3: Input file**

The scheduler tool accepts the input in a tabulated form. The input is given as a spreadsheet file. The backend extracts the required information from the input file such as the number of machines, number of jobs, number of pieces per job, the time required by each object to process on a particular machine, setup time for each machine and the delay time.

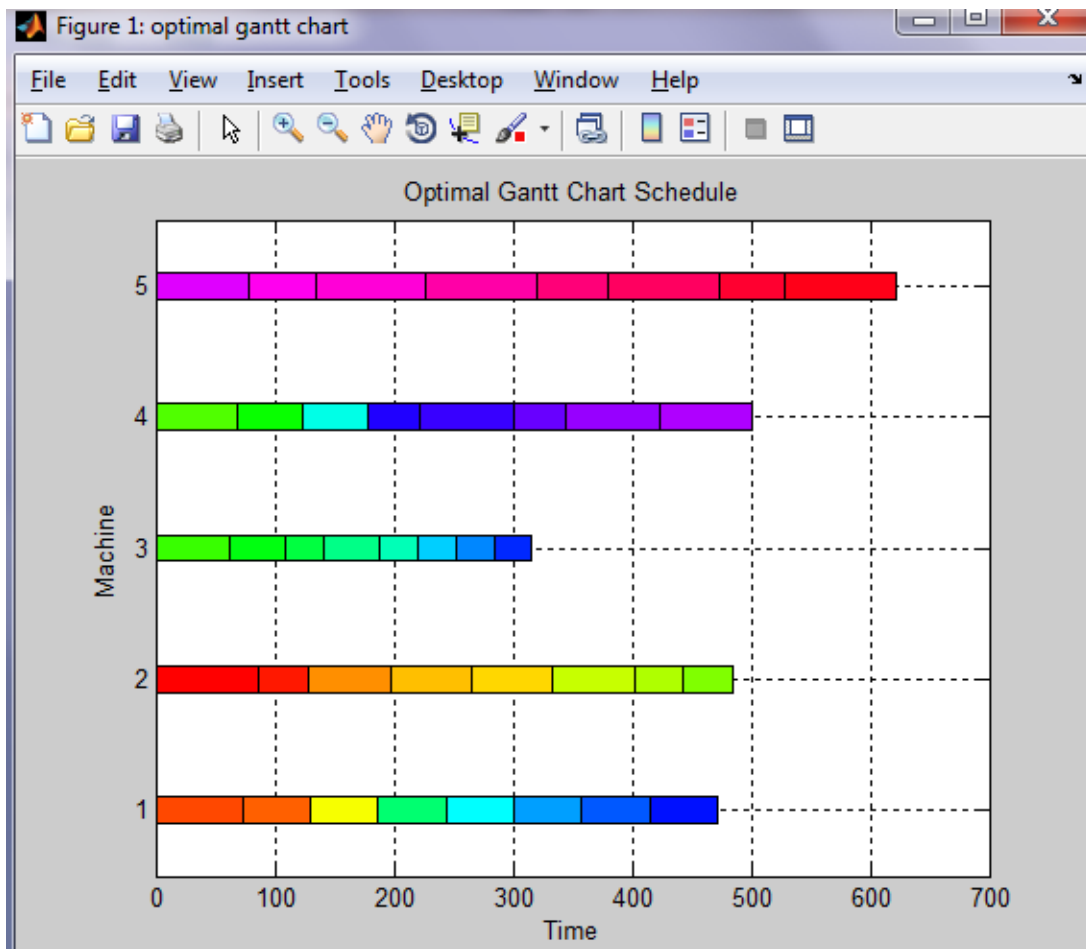
If a particular job doesn't go to a machine then the corresponding processing time is taken to be a garbage value indicating 'no processing'.





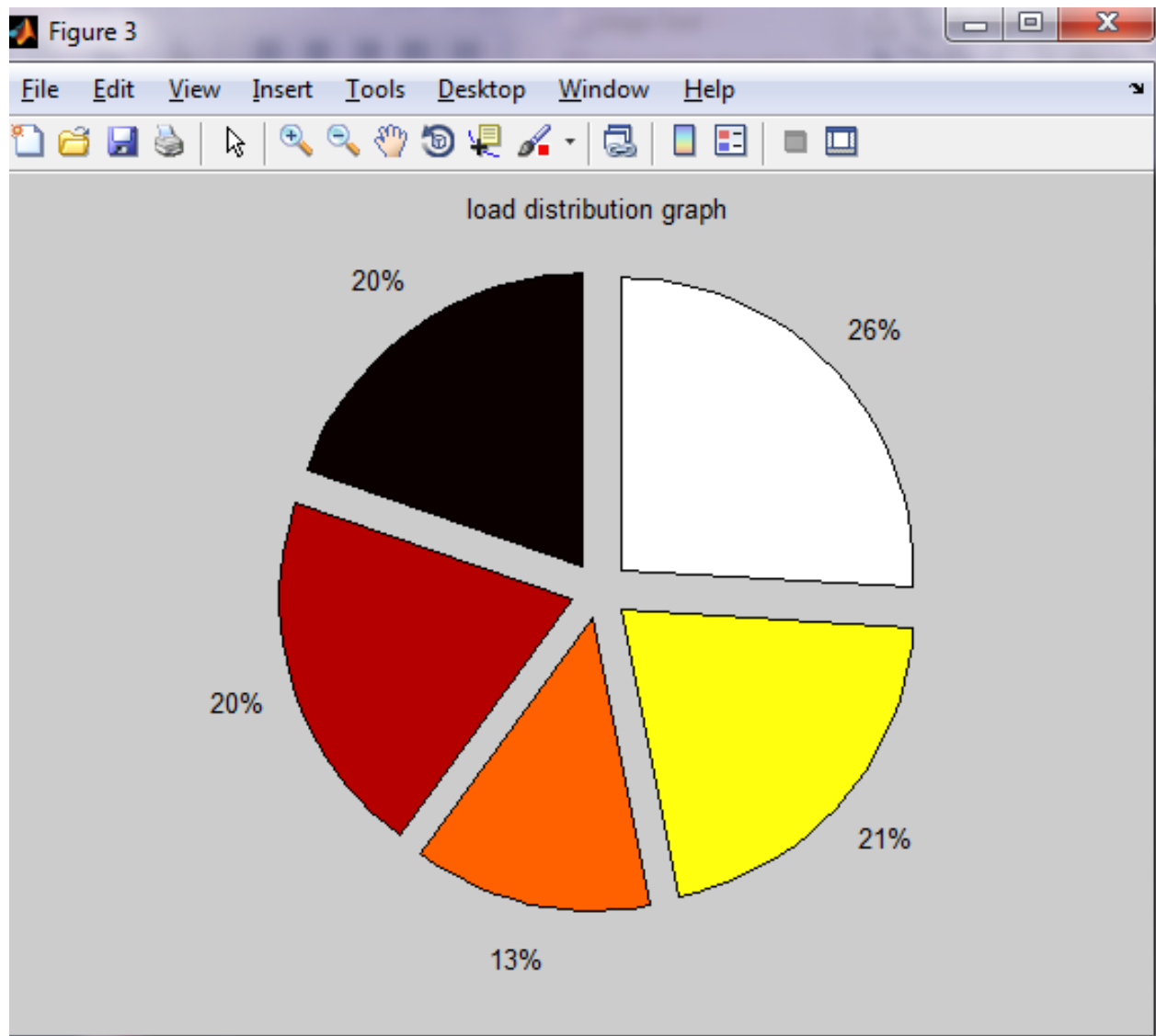
**Fig7.4: Selecting values for various parameters.**

The above screen depicts how the user gets to choose the crossover rate, mutation rate and the number of iterations. A dropdown list appears on clicking, displaying the range of values from which the parameter value is to be selected



**Fig7.5: Optimal Gantt Chart**

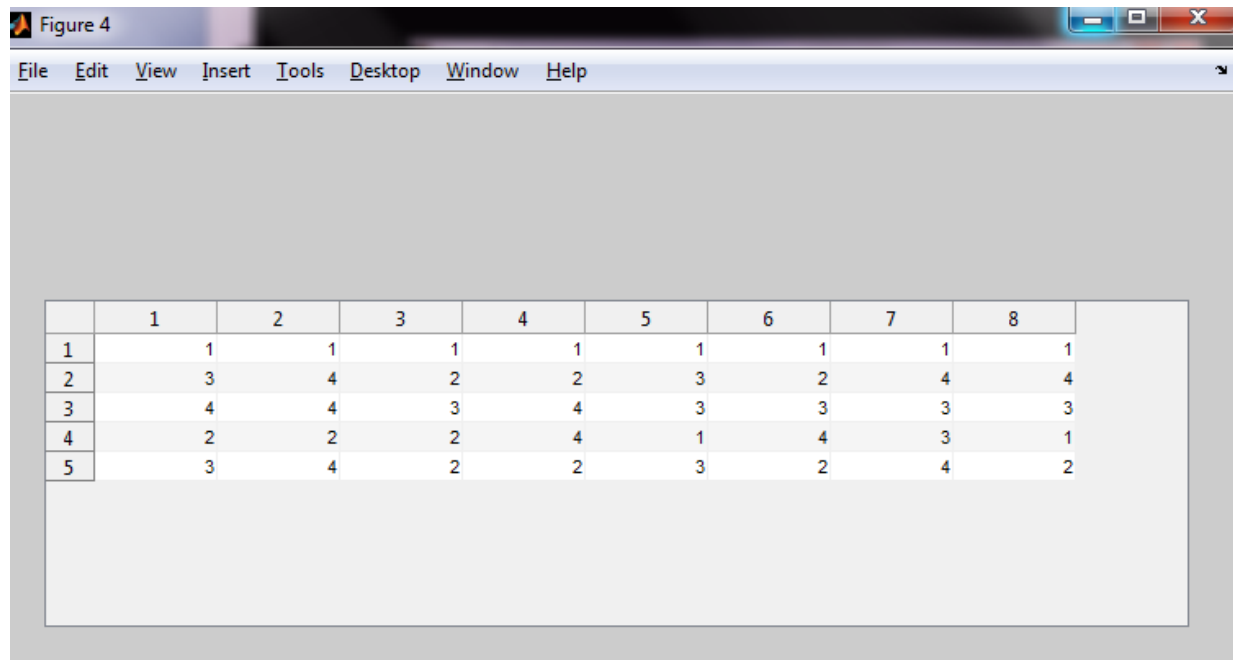
In the above figure , each operation that is to be done on a particular machine is shown in Gantt chart form. Thus the duration of each job and the total time for which each machine is working can be easily see. The function used for generating the Gantt chart is `creategantt(MATRIX)`;it creates the stacked bar graph of each machine usage.



**Fig7.6: Load Distribution graph**

The above figure depicts the load distribution graph which shows the percentage of work done by each machine. This pie chart is useful as it can help to detect which machine is becoming worn out and should be replaced. Thus the load bearing capacity of each machine can be seen

The function used to generate the above is: `utilgraph(mutil)`.



	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	3	4	2	2	3	2	4	4
3	4	4	3	4	3	3	3	3
4	2	2	2	4	1	4	3	1
5	3	4	2	2	3	2	4	2

**Fig7.7: Machine sequence**

This screen is the matrix representation of operation number on each machine. This output can be easily pasted on job boards or near machines to get the schedule of the entire batch to be produced.

## 8. Results and Conclusion

### Improved Solution to Job Shop Scheduling Problem with Delay Constraints using Genetic Algorithm by Shelly Chikara, Shruti Kapoor and Swati Singh.

They concluded that:

The jssp many type of schedules depending upon the number of jobs and machines. Since the ability and functions of modern machines have been widely extended, the scheduling plan of a part might not be unique. So on a real shop floor, many feasible schedules can be found. It can be seen that Genetic Algorithm gives better result than traditional scheduling methods. The applicability of the GA based methodology has considerable potential application to manufacturing with further refinement in certain aspects, as outlined below.

1. We have considered transportation time and setup time as constant, further work can be done using setup time and transportation time.
2. We have assumed no break down, further work can be done taking into account actual breakdown.
3. The jobs are considered to be atomic. Further work can be done by creating schedules with various parts manufactured on different machines.

	<b>M1</b>	<b>M2</b>	<b>M3</b>	<b>M4</b>
<b>J1</b>	40	50	60	70
<b>J2</b>	40	57	34	27
<b>J3</b>	60	54	80	36
<b>J4</b>	40	34	66	34
<b>J5</b>	49	23	34	20
<b>J6</b>	35	45	55	50

**Table 2: Delay time in minutes in processing**

	<b>M1</b>	<b>M2</b>	<b>M3</b>	<b>M4</b>
<b>J1</b>	10	12	8	7
<b>J2</b>	3	7	13	10
<b>J3</b>	6	4	10	12
<b>J4</b>	7	4	10	6
<b>J5</b>	1	4	8	10
<b>J6</b>	2	3	1	6

The above table shows the part processing time of each job per machine and the sequence of machines to be used in the production of each operation.

**Table 8.1: Result comparison of previous work done**

<b>Shruti Singh et al.</b>		<b>Our Results</b>	
<b>No. of generation</b>	<b>Minimum Makespan</b>	<b>No.of Generations</b>	<b>Minimum Makespan</b>
10	728	10	722
100	716	100	720
500	706	500	678
1000	698	1000	654
5000	698	5000	675
10000	682	10000	669

Thus the results show the makespan of the schedule produced for various iterations for the data mention in the paper **Improved Solution to Job Shop Scheduling Problem with Delay Constraints using Genetic Algorithm** by Shelly Chikara, Shruti Kapoor and Swati Singh.

## 9. Conclusions and future scope of the study

Scheduling as one of the most challenging of all decision making problems in production, the level of scheduling knowledge available in manufacturing industries is very poor. Practical aspects of scheduling is not formally taught at academia and the education on scheduling among industries is almost non-existent. People are learning production scheduling by common sense, experience and software tools. The understanding of the dynamic nature of the workflow on the shop floor and the perception of scheduling complexity are fairly low in job shops. The most interesting aspect of production scheduling is that a scheduler can somehow ensure workflow by making real-time decisions with common sense (pushing each job from one work centre to the next one based on work progress) or adopting any of the methods described above while the researchers consider job shop scheduling as highly complex. It is the simultaneous production of diverse jobs (with small quantities) using shared resources that makes job shop scheduling quite complex. The jobs have different routings, due dates, priorities, quantities and materials and resource requirements.

In general, a job shop is a production unit where order quantities are usually small, process requirements vary with customer order, and processing starts for an order only after receiving the order from the customer and many work orders are simultaneously produced using shared resources. Even the resource and material requirements often vary with the order in job shops. There are some production systems where each order is a project that involves numerous tasks with dependency relations and requires many finite capacity resource for performing the task. Although most of the job shops are relatively smaller in size and revenue, from production management viewpoint, they are more complex than a large repetitive production system. Many job shops with capital also have a difficulty to maximize raw material inventories in order to achieve short order-to-delivery lead times.

There are many schedules in an FMS depending upon the number of jobs and machines. Since the ability and functions of the modern machines have been widely extended, the scheduling plan of a part might not be unique. So on a real shop floor, many feasible solutions can be found. It can be seen that Genetic Algorithm gives better results than the traditional scheduling methods. GAs use basic Darwinian mechanism “Survival of the fittest”, and repeatedly utilize the information contained in the solution population to generate new solutions with better performance.

Classical GAs use binary strings to represent potential solutions. One main problem in classical GA is that binary strings are not naturally suited for FMS. From the viewpoint of FMS itself, it is a hard combinatorial problem with constraint. The goal of the scheduling methods is to find a solution that satisfies the constraints. In the proposed genetic algorithm, the potential solutions are generated without considering the constraints. The applicability of GA based methodology has considerable potential application to manufacturing with further refinements in certain aspects, as outlined below:

1. We have considered setup time and delay (transportation time) as constant, further work can be done using setup time and transportation time.

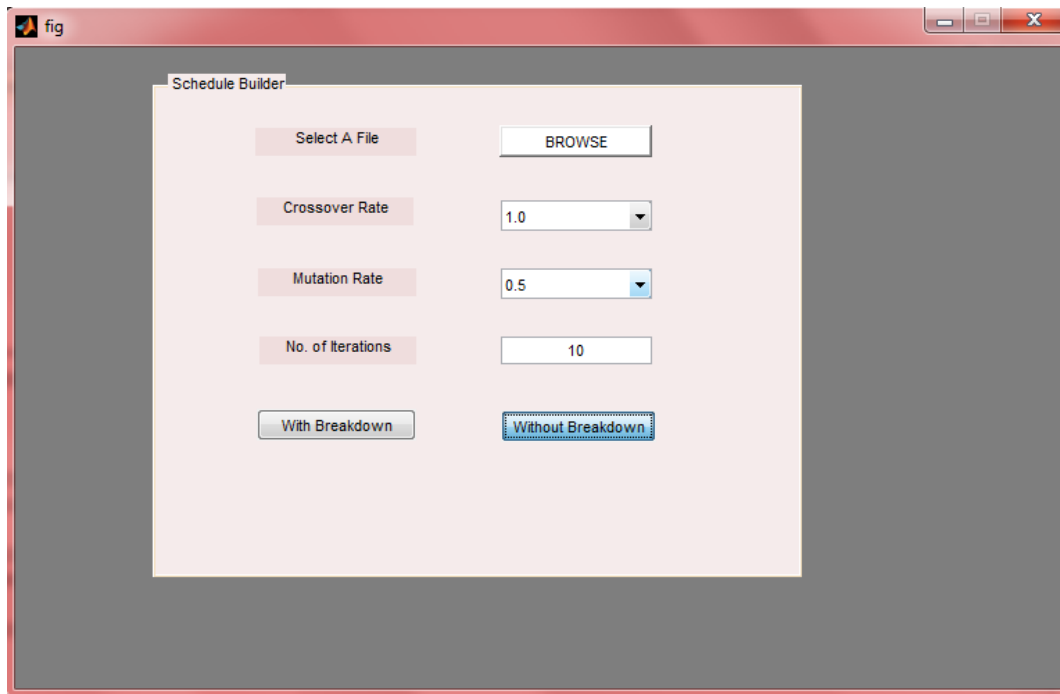
2. The jobs are considered to be atomic. Further work can be done by creating schedules with various parts manufactured on different machines.
3. No preemption has been done, further makespan can be reduced by the use of preemption.



## 10. Appendix

### 10.1 Screenshots

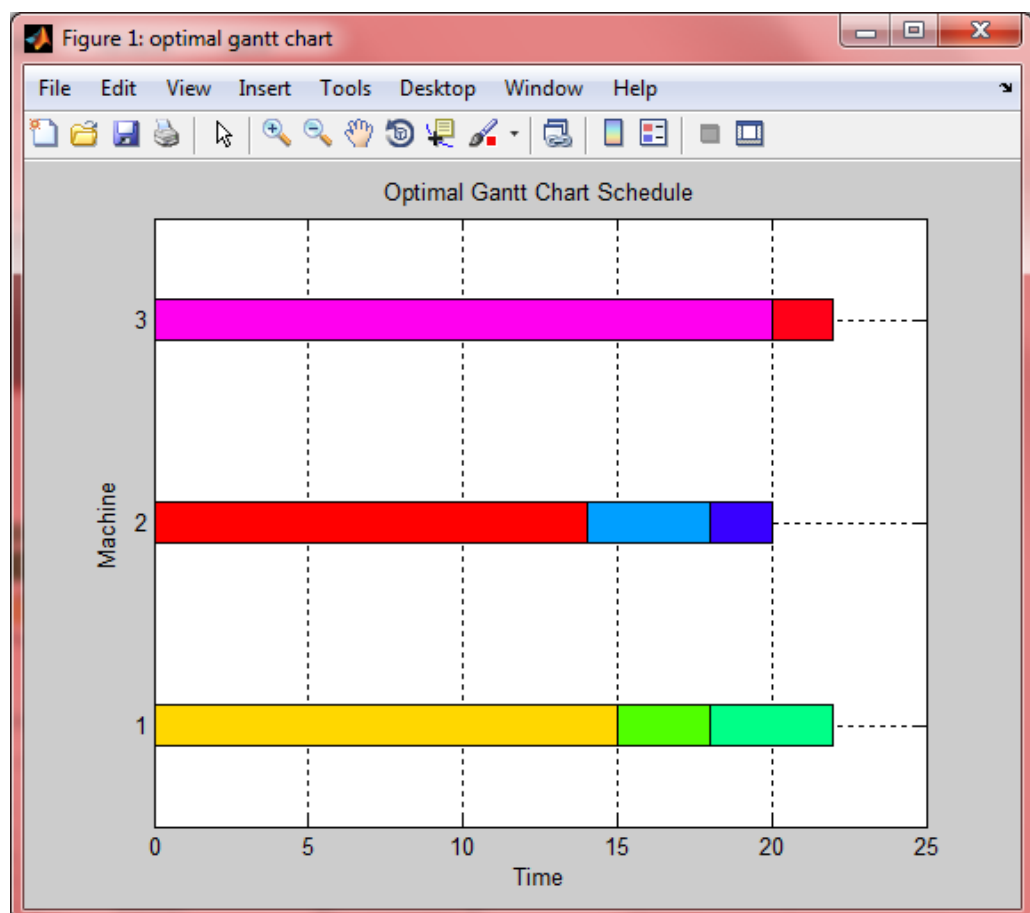
For Data 1 considering setup and delay time



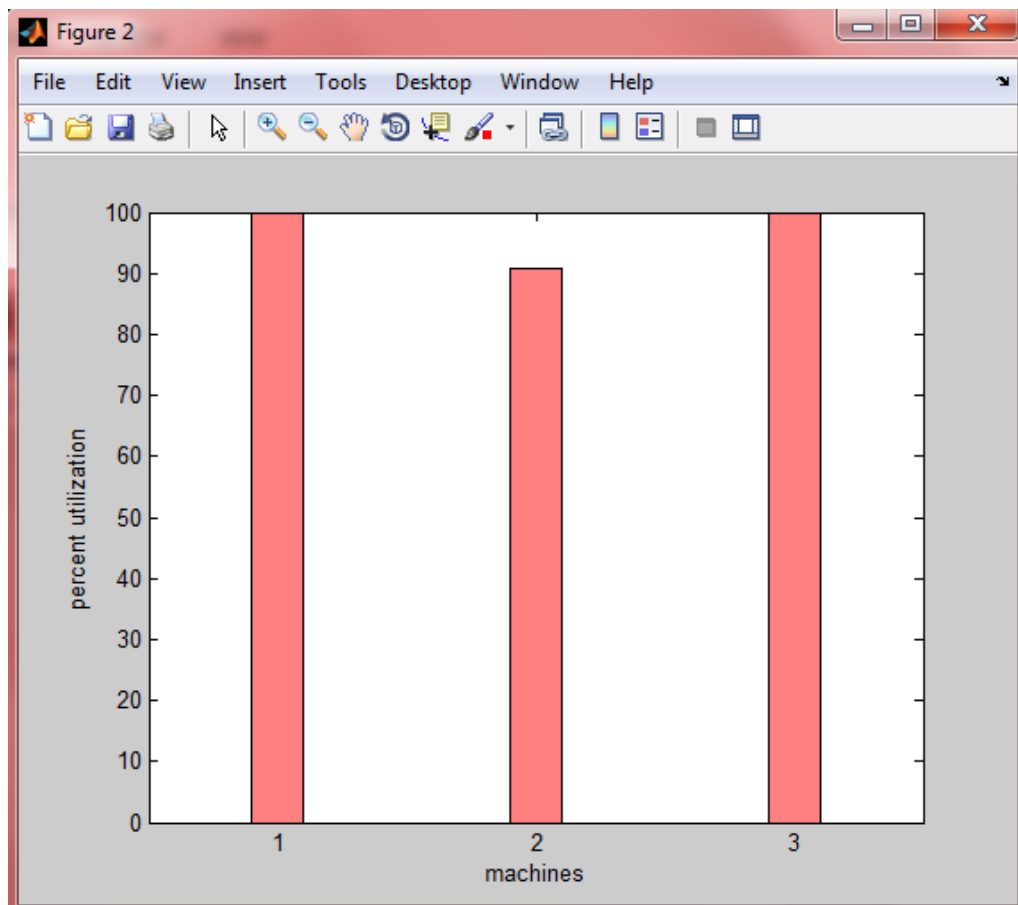
**Fig10.1:GUI of the project with Without Breakdown button clicked.**

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		3	no. of machines										
2		4	no. of jobs										
3		2	no. of pieces per job										
4		3	2	9999									
5		4	9999	6									
6		9999	2	2	part processing time								
7		1	4	8									
8		1	0	0									
9		1	0	2									
10		0	1	0	delay								
11		2	0	2									
12		10	11	12	setup time								
13													
14													
15													
16													
17													

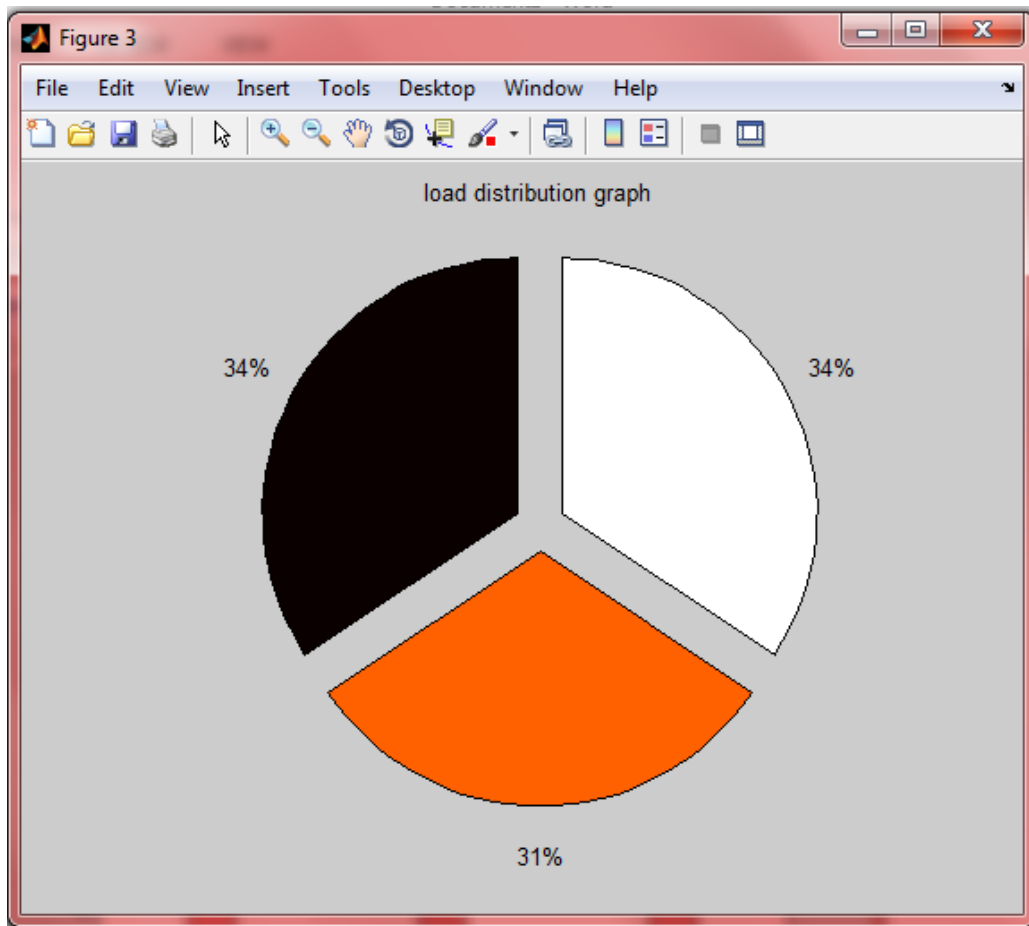
**Fig10.2:Input spreadsheet file**



**Fig10.3: Optimal Gantt chart**



**Fig10.4:Machine utilization graph**



**Fig10.5: Load distribution graph**

Figure 4 is a software window titled 'Figure 4' with a menu bar (File, Edit, View, Insert, Tools, Desktop, Window, Help) and a toolbar. The main area displays a table with the following data:

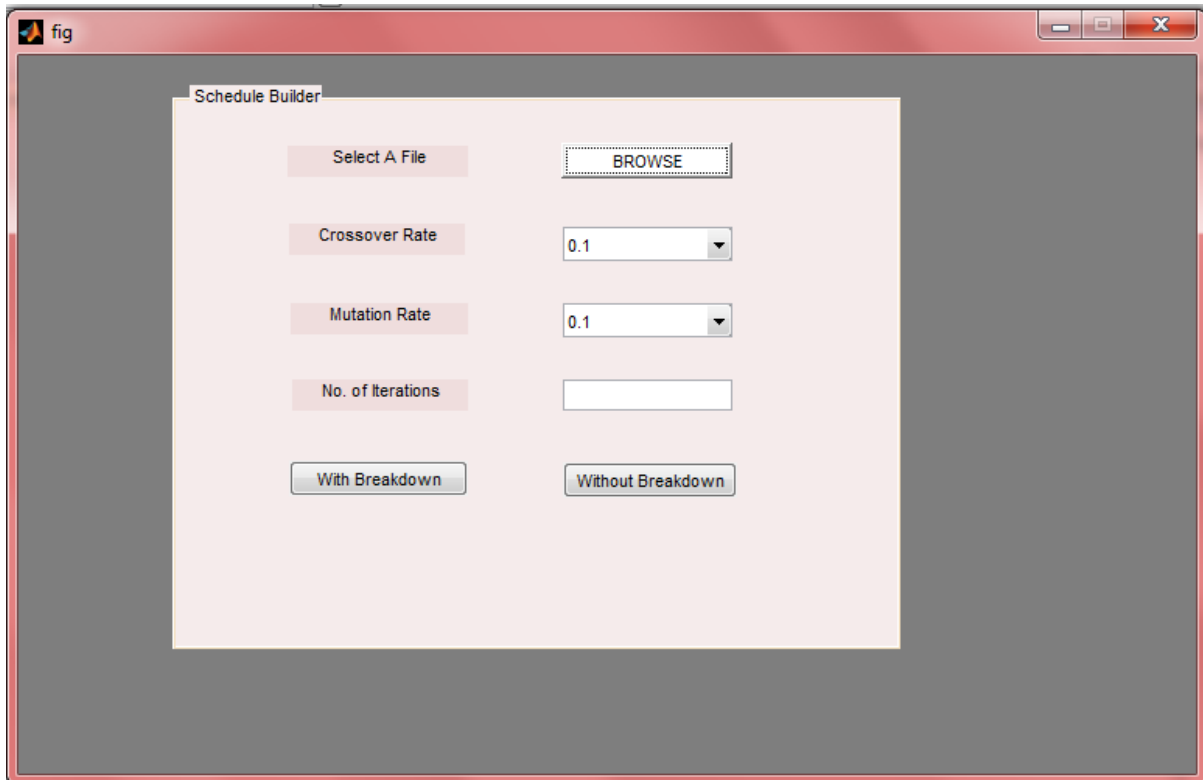
	1	2	3	
1	2	4	1	
2	3	4	1	
3	2	3	0	

**Fig10.6:Machine sequence**

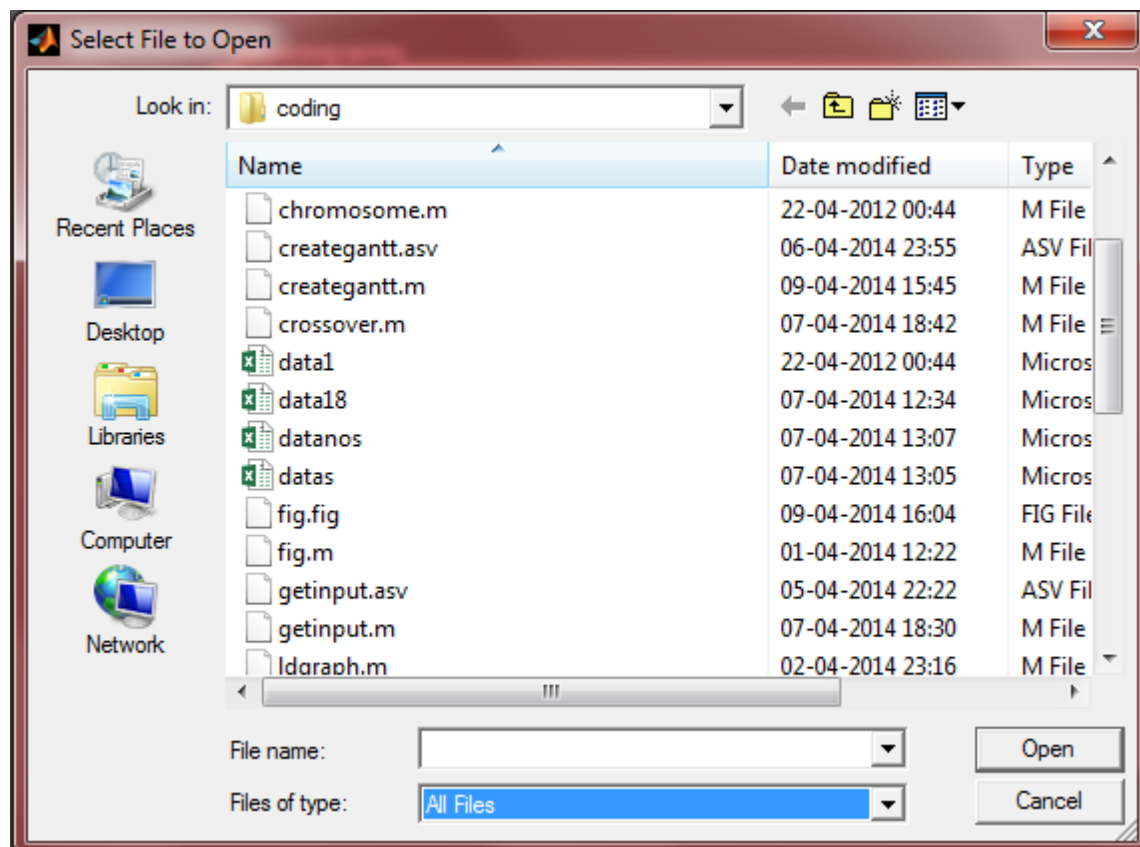
**For Data with no setup time named as Datanos**

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		5	number of machines										
2		4	number of operations										
3		10	number of pieces per job										
4	55	9999	65	75	9999								
5	9999	67	9999	52	90								
6	9999	65	31	75	59		part processing times						
7	9999	40	45	40	55								
8	0	0	0	0	0								
9	0	0	0	0	0		no delay						
10	0	0	0	0	0								
11	0	0	0	0	0								
12	0	0	0	0	0		no setup time						
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													

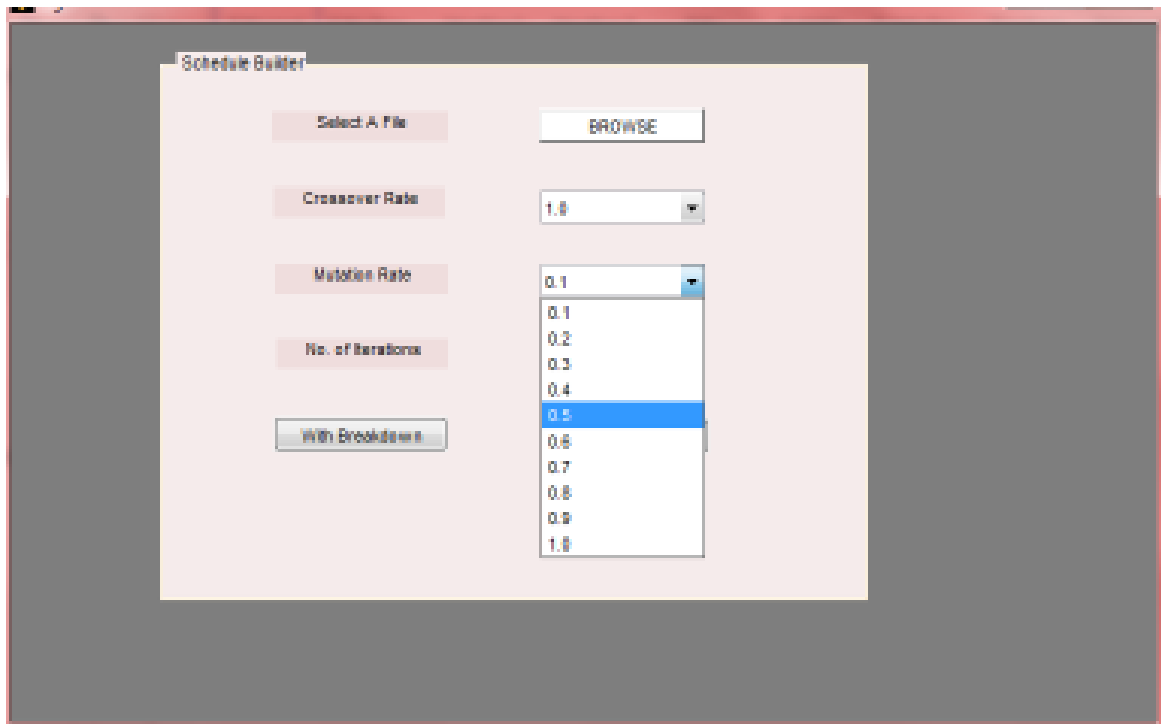
**Fig10.7:Input spreadsheet file with no setup and delay time**



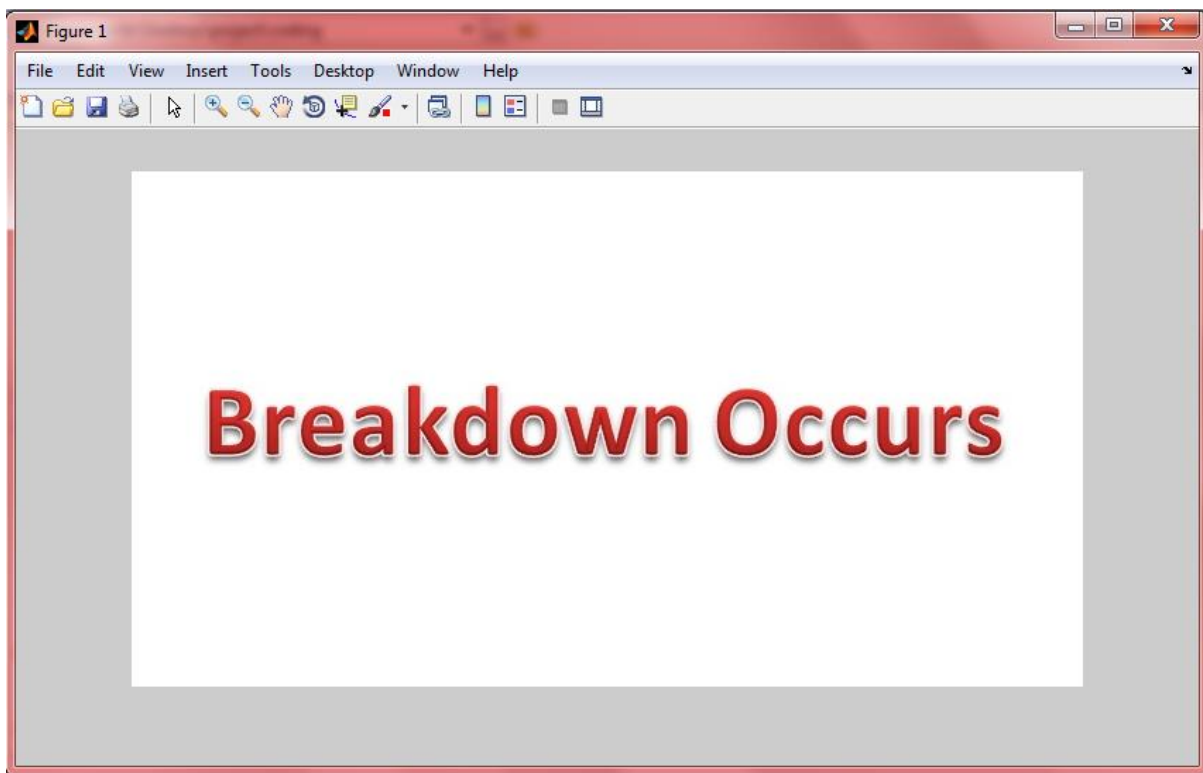
**Fig10.8:GUI of the project where ‘With Breakdown’ button is pressed**



**Fig10.9:On clicking on browse button**

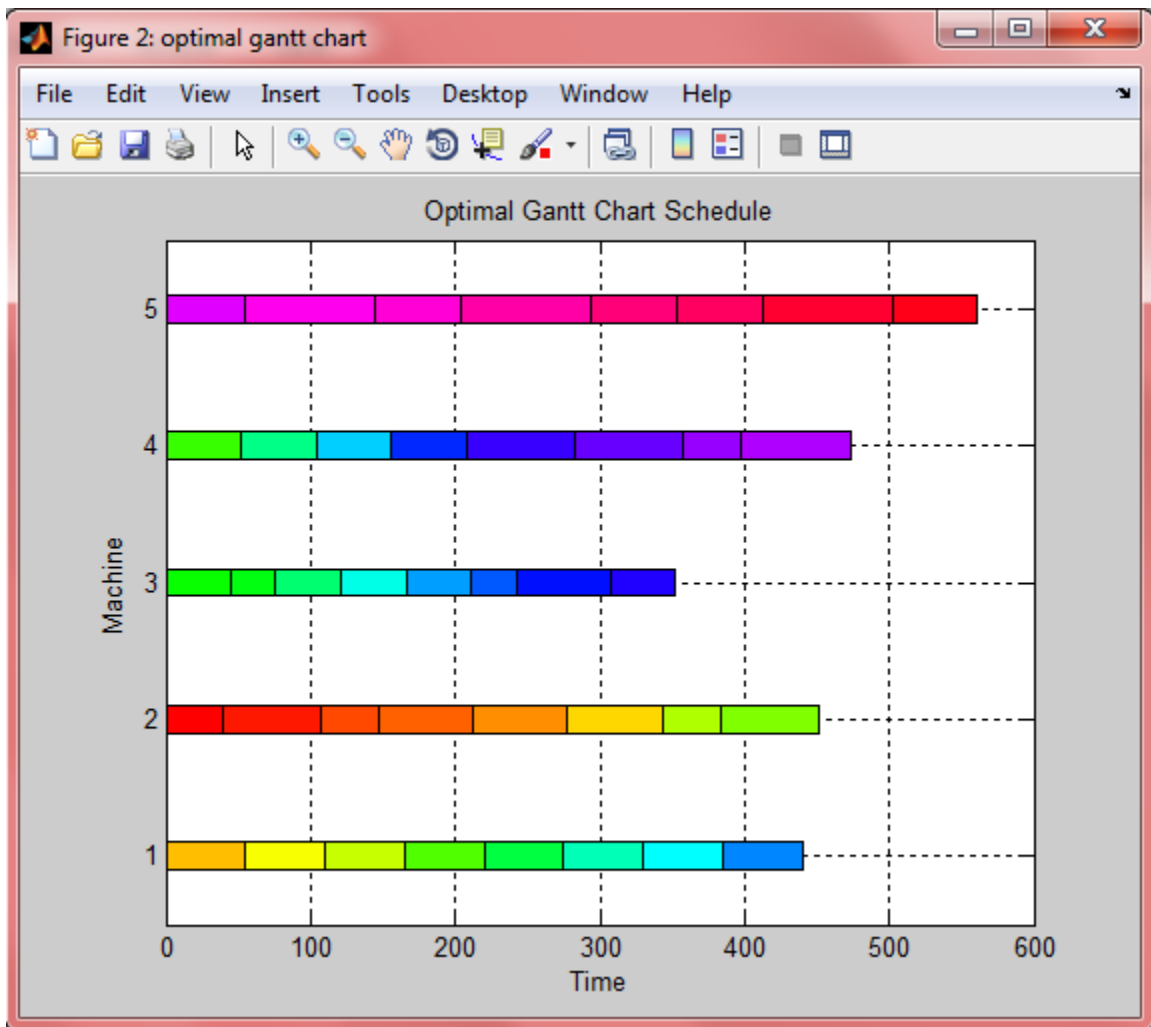


**Fig10.10:Select the parameter values for crossover and mutation rate**

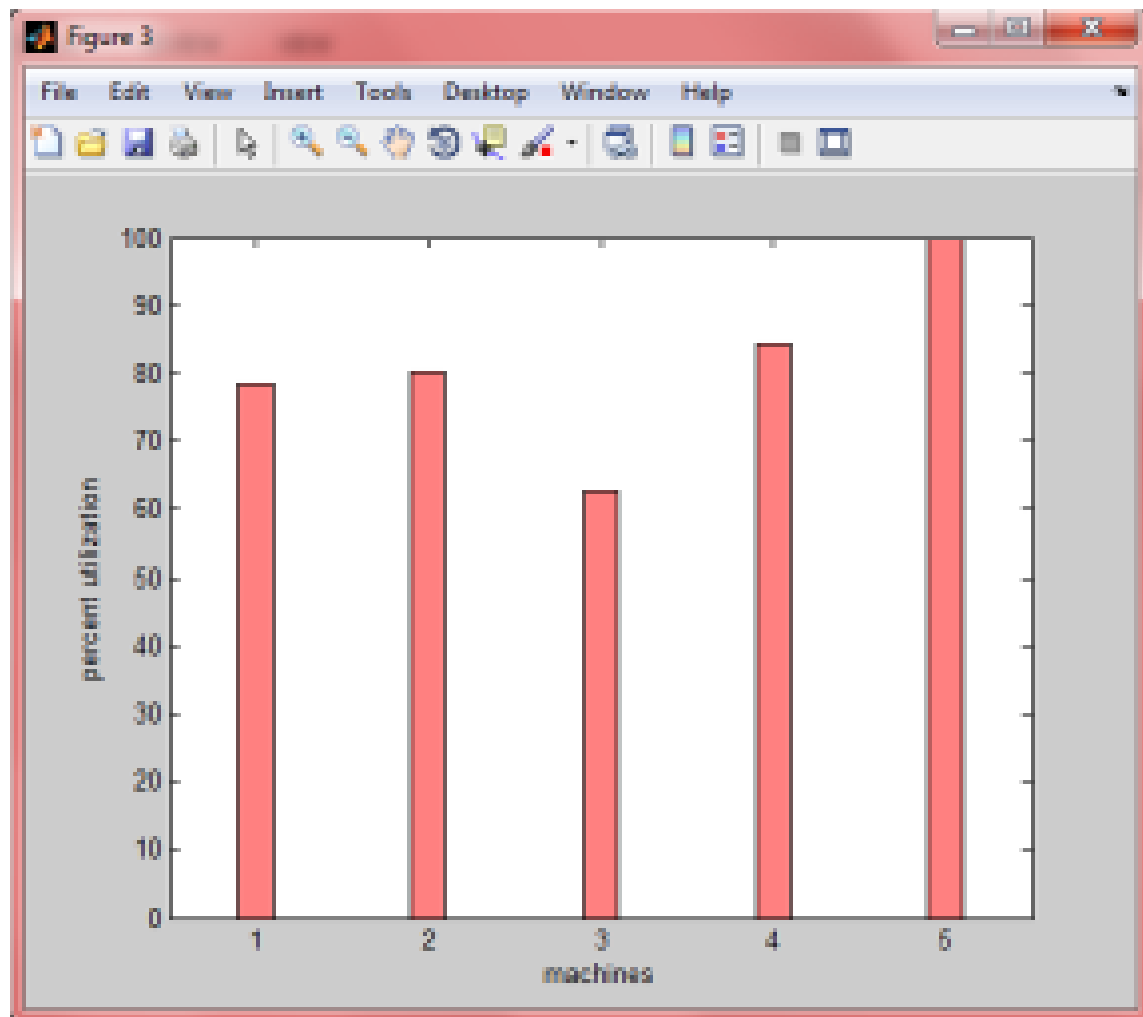


**Fig10.11:figure depicting that randomly breakdown occurred.**

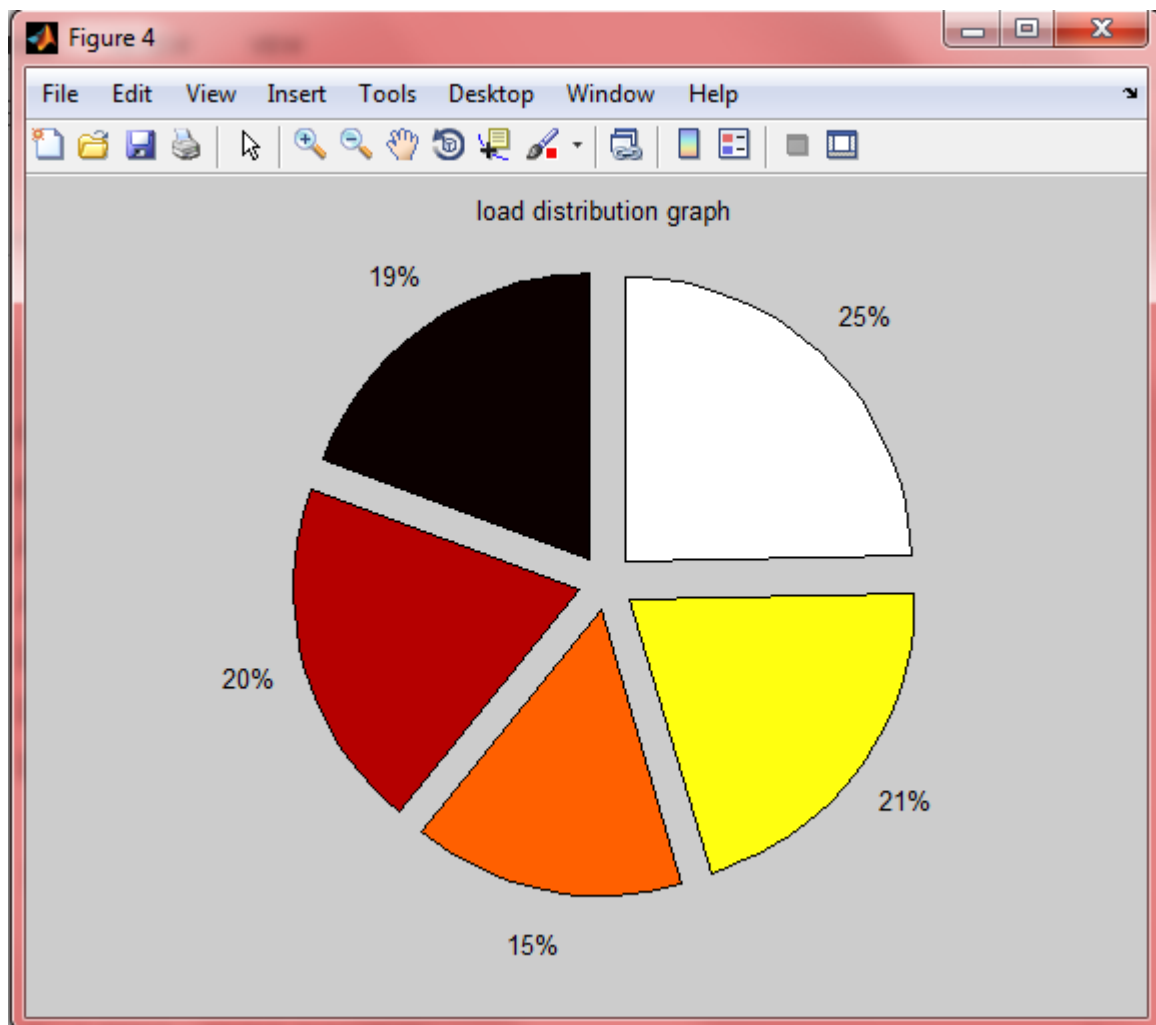




**Fig10.12:Optimal Gantt chart**

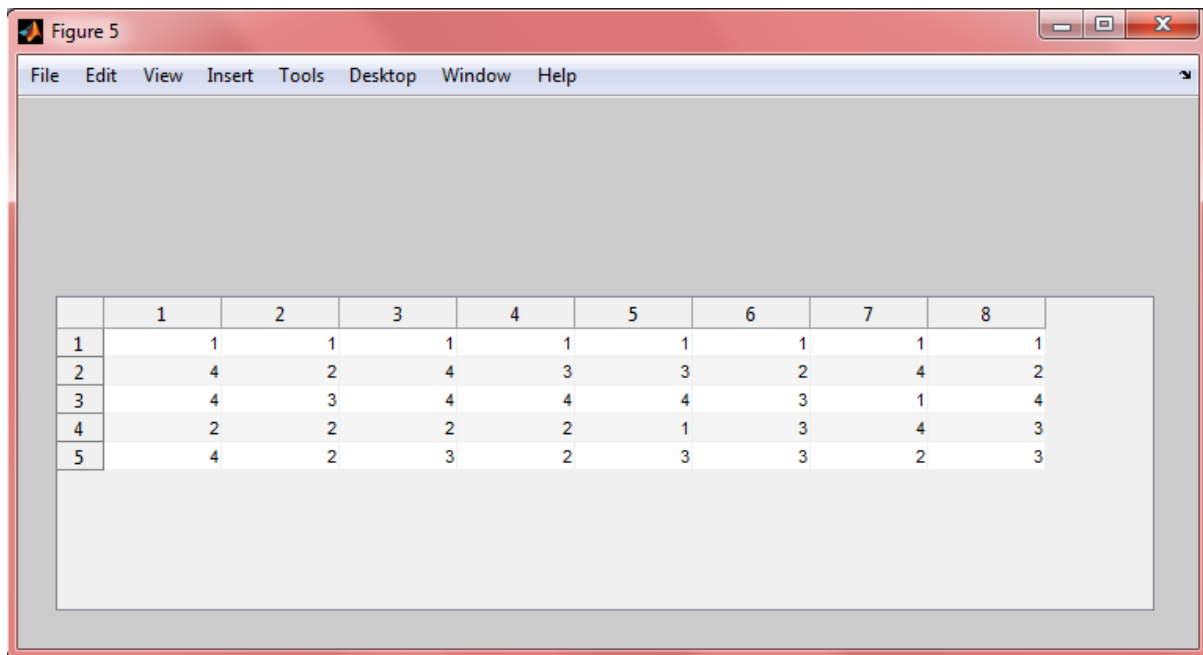


**Fig10.13:Machine utilisation graph**



Fig

**Fig10.14:Load distribution graph**



**Fig10.15:Machine sequence**

## 10.2 Research publications

# Scheduling of flexible manufacturing system using Genetic algorithm(multiobjective):A Review

Navnikaa Rajan

CSE Dept.

IMS Engineering

College

Srishti Jaiswal

CSE Dept

IMS Engineering

College

Tanya Kalsi

CSE Dept

IMS Engineering

College

Vijai Singh

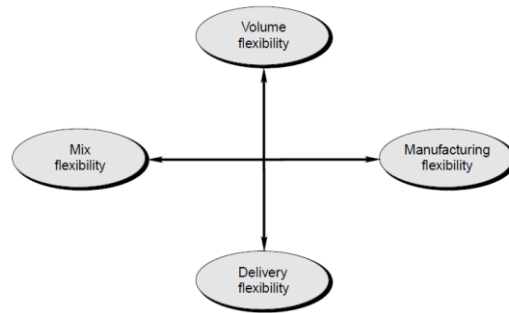
Asst.Professor,CS

IMS Engineering

College

**Abstract** A flexible, integrated, computer-controlled environment allows the system to react on occurrence of changes ,whether predicted or unpredicted. Scheduling machines of varying capabilities in such an environment has always been a difficult task. This work reviews the various approaches applied to the scheduling problem in an FMS. Various genetic algorithm based approaches considering varied objectives and constraints have been studied and analysed to result in a comparative study. For achieving the desired performance in an FMS it is required that a good scheduling system, taking into account the system conditions should generate an optimal schedule at the right time. Genetic algorithm is capable of finding near to optimal solution in a short time although it doesn't guarantee to find an optimal solution.

**Introduction** To sustain in today's competitive global market manufacturing organizations have to develop a manufacturing system that can fulfil the changing demands of customer for customised products. The system should be flexible, productive and should be able to meet the demands within time bounds at a reasonable cost. FMS belongs to a class halfway amidst job shop manufacturing system and batch manufacturing system. An FMS has an integrated and computer controlled configuration which is capable of automatically changing tools and parts. These machines are interconnected by automatic guided vehicles,pallets and storage buffers that have flexibility that allows to modify system behaviour on occurrence of changes whether predicted or unpredicted. It is modelled as a collection of workstations. The FMS should be designed to simultaneously manufacture different volumes of a varying variety of high quality products.The flexibility may be machine flexibility or routing flexibility.Machine flexibility refers to system's ability to produce new product types and change the sequence of operations executed on a part. Routing flexibility is the ability to absorb large scale changes such as in volume,capacity and capability.



**Fig 1:Types Of flexibilities**

The arrangement of machines in an FMS is connected by a transport system. The components are automatically governed using local area network.

#### *Basic components of FMS*

FMS basically composes of the following three parts:

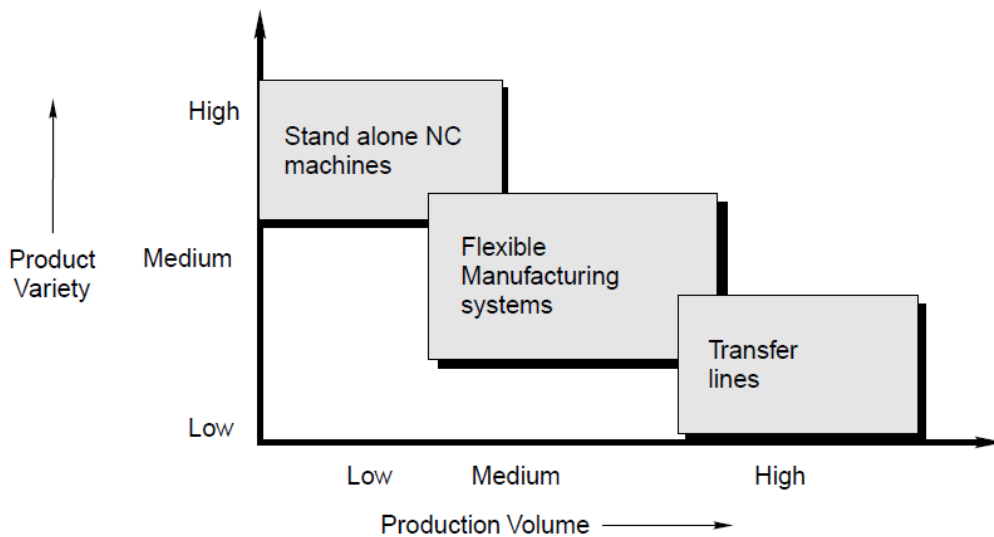
1.*Workstations:* A machine tool which is computer controlled is called a workstation. Machine centers,load/unload stations,assembly workstations,inspection stations,forging stations,sheet metal processing etc are a few examples of workstations.

2.*Automated Storage stations and Material handling stations:* The movement of workparts and sub assembly parts between different workstations is done mechanically which is referred to as automated material handling and storage system.The functions performed are:

- (i) Random movement of workparts between stations independently
- (ii) Handling various work part configurations
- (iii) Temporary storage
- (iv) Loading and unloading of work parts for easy access
- (v) Computer control compatibility

3.*Computer controlled systems:*The functioning of the stated components is co-ordinated by a controlling Computer System. Its functions are:

- (i) Controlling work stations
- (ii) control instruction distribution to the work stations
- (iii) Controlling production
- (vi) Controlling traffic
- (v) Monitoring and work handling
- (vi) Monitoring the performance of the system and reporting



**Fig 2:Application characteristics of FMS**

**Table 1.1** Different approaches to flexibility and their meanings

<i>Approach</i>	<i>Flexibility meaning</i>
1. Manufacturing	<ul style="list-style-type: none"> <li>■ The capability of producing different parts without major retooling</li> <li>■ A measure of how fast the company converts its process from making an old line of products to produce a new product</li> <li>■ The ability to change a production schedule, to modify a part, or to handle multiple parts</li> </ul>
2. Operational	<ul style="list-style-type: none"> <li>■ The ability to efficiently produce highly customized and unique products</li> </ul>
3. Customer	<ul style="list-style-type: none"> <li>■ The ability to exploit various dimension of speed of delivery</li> </ul>
4. Strategic	<ul style="list-style-type: none"> <li>■ The ability of a company to offer a wide variety of products to its customers</li> </ul>
5. Capacity	<ul style="list-style-type: none"> <li>■ The ability to rapidly increase or decrease production levels or to shift capacity quickly from one product or service to another</li> </ul>

Various approaches have been proposed for the scheduling problem of FMS including branch and bound, priority rules, simulated annealing, tabu search and genetic algorithm. Among these approaches the genetic algorithm approach with variations has been extensively applied for scheduling in an FMS.

Genetic algorithm by J.Holland is a randomised search and optimisation technique which is governed by the principles of evolution and natural genetics. It has been studied that it is difficult for the traditional optimisation techniques to approach the best solution. It is observed that genetic algorithms have better capability than the traditional scheduling methods.

## Genetic Algorithm

GAs are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetic. As such they represent an intelligent exploitation of a random search used to solve optimisation problems. They exploit historical information to direct the search into the region of better performance within the search space.

GA belong to the class of evolutionary algorithms. John Holland based GA on the basic Darwinian Mechanism of “Survival of the fittest”[12].

*Principle:* The principle of GA is simple, it imitates genetics and natural selection by a computer approach. The parameters of the problem are coded most naturally as DNA. A set called population, of these problem dependent parameter value vector is processed by GA.

They are more robust and hence better than the traditional algorithms. They do not break easily even if the inputs are varied slightly or in the presence of reasonable amount of noise.

Genetic algorithm are preferred over other approaches because of the following advantages

- Genetic Algorithm works smoothly with numerical experimental or analytical functions and data.
- Well suited for parallel computing.
- Genetic Algorithm optimizes both variables efficiently, continuous or discrete.
- It doesn't require any derivative information.
- It searches from a large sampling of the cost surface.
- It is capable of handling a large number of variables at the same time.
- It can optimize variables with highly complex cost surfaces.
- Gives a number of optimum solutions, not a single solution.
- It works on encoded variables

A GA typically consists of the following steps:

**Step 1:** Generate the initial population. Obtain the population size and the maximum number of iterations.

**Step 2:** Compute the fitness function for every member of the initial population.

**Step 3:** Compute the selection probability for every member of the initial population by calculating the ratio of fitness value of that initial population to the sum of the fitness values of all the individuals.

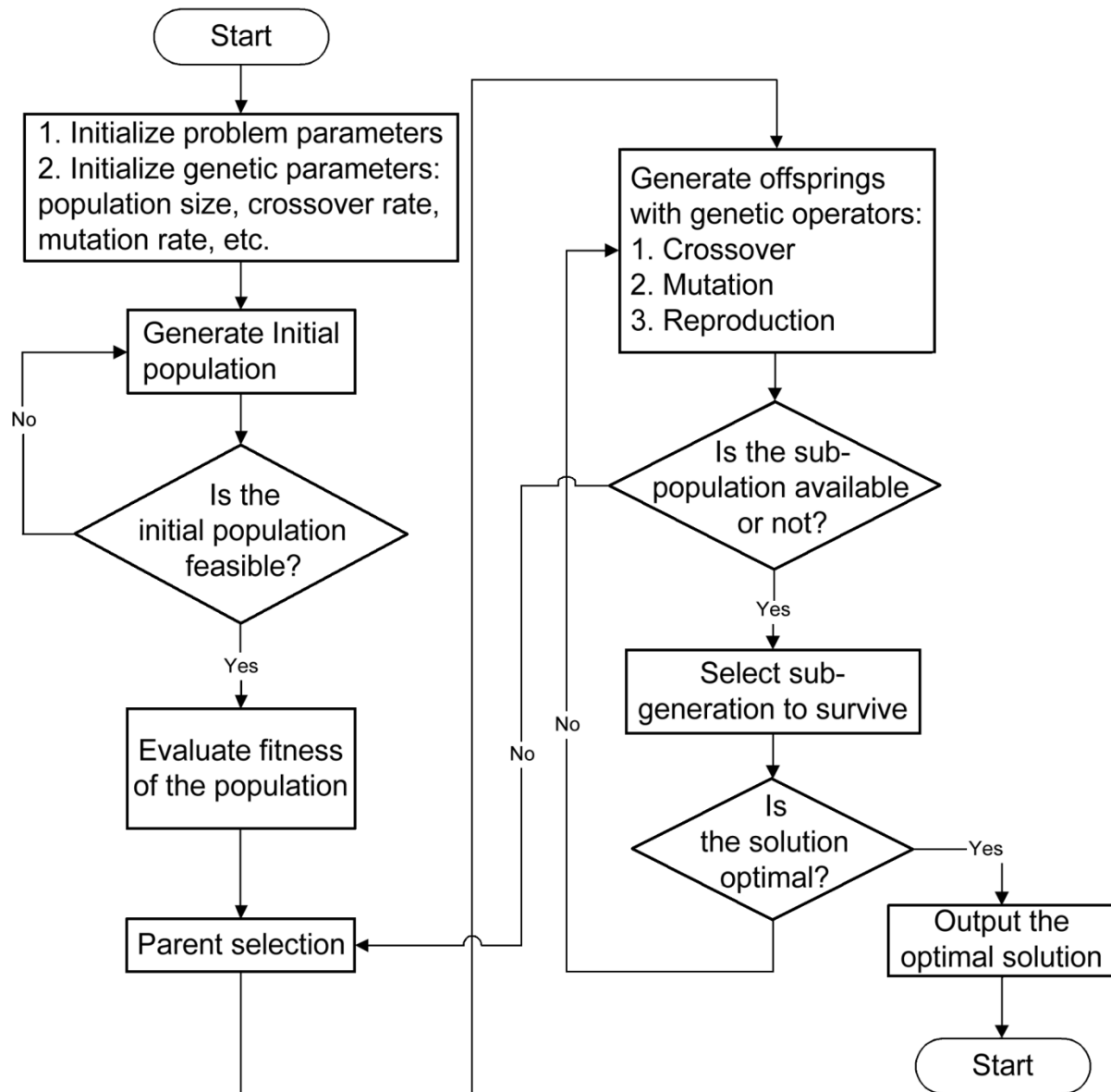
**Step 4:** Select two members (parents) to be used for reproduction based on selection probability.

**Step 5:** Apply crossover, mutation, and inversion to the parents to obtain the offsprings. Generated offsprings form the new generation.

If the size of the new population is equal to the initial population size, go to step 6, else go to step 4.

**Step 6:** If the current generation is equal to the maximum number of the generation then stop, else move to step 2.





**Fig 3: Steps of genetic algorithm**

**Selection:** The process of choosing a pair of parents from the population for crossing in order to generate offsprings is called crossover. The aim of selection is to choose fitter individuals so that the population has offsprings having higher fitness value. Selection process randomly chooses chromosomes out of population based on the evaluation of their fitness function value. The higher fitness function value, more are the chances for the individual of being selected.

**Crossover:** The process of taking two parent solutions and generating a child is called crossover. The selection process, enriches the population with fitter individuals. Crossover is a genetic operator used in genetic algorithm to vary the programming of a chromosome or chromosomes from one generation to the next. It is similar to reproduction in biological crossover, on which genetic algorithms is modelled.

**Mutation:** Mutation helps in avoiding the algorithm from being trapped in a local minimum. Mutation is responsible for recovering the lost genetic information. As crossover is responsible for exploiting the current population for finding better parents, mutation is designated for the exploration of the entire search space. It is an operator used for maintaining genetic diversity in the population.

## Literature Survey

To gain a better insight into the FMS scheduling problem solution space a comparative review of the various algorithms, techniques and approaches has been done. The comparison was carried out while keeping in mind various parameters such as selection, mutation, crossover, fitness criteria and solution space. After in-depth analysis a conclusion can be inferred that there is a trade-off between optimality of solutions and computation time required. Since JSSP is a NP complete problem, there exists no specific algorithm which would give the optimal solution for all cases.

[9] presents a new GA based Discrete Dynamic Programming approach for generating static schedules in a FMS environment. It adopts a sequence dependent schedule generation strategies where to generate feasible job sequences, GA is employed and a series of DDP are constructed to generate schedules. The author stated that DDP is capable of identifying locally optimised partial schedules. The proposed algorithm does not suffer from state explosion problem. The author concluded that the algorithm was successfully implemented for generation of schedules.

In [5] three objectives i.e. minimise makespan, minimise maximal machine workload and minimum total workload are addressed and a hybrid genetic algorithm is proposed for a problem. They have used two vectors to represent the solution also two operators i.e. advance crossover and mutation operator are used to adapt special chromosome structure and varying characteristics of the problem. Authors have improved the genetic algorithm by variable neighbourhood descent in order to strengthen the search ability. The approach used involved two local search procedure; local search of moving one operation, local search of moving two operations. They have developed an efficient method for finding assignable time intervals for the deleted operations based on the earliest and latest event times. To unify the operation sequence in chromosome with the sequence in the decoded schedule or recorded procedure is used which facilitates genetic operators to pass from the good traits of the parents.

Liang Sun et al in their work [4] proposed a genetic algorithm with the penalty function for the FMS scheduling problem. For this, they used a clonal selection based hyper mutation and a lifespan extended strategy. During a search process, an adaptive penalty function is decided so that the algorithm can search in both feasible and infeasible solution of the solution space. They conducted experiments on 23 benchmark and instances of the OR-Library. The proposed algorithm effectively exploits the capabilities of distributed and parallel computing of swarm intelligence approaches and effectively makes use of the famous scheme theorem and the building block hypothesis of Holland. The results indicate the successful incorporation of the proposed operators.

In paper [3] M. Heydar solved the FMS scheduling problem considering two objectives i.e. maximum completion time (makespan) and maximum tardiness. This scheduling problem is stated belonging to the class of NP hard problems and hence no exact method is appropriate to solve the practical cases of scheduling problems. They proposed a hybrid genetic algorithm combined with four priority dispatching rules. The proposed approach resulted in performing well in efficiency and quality of solutions.

In paper [6] Christian Bierwirth proposed a new representation technique mathematically known as permutation with repetition in order to sequence the operations of an FMS on a number of machines related to the technological machine order of jobs. A new crossover operator is designed with similar behaviour to the Order-Crossover for simple permutation schemes which reserves the initial scheme structure of permutations with repetitions. Together, the new representation and Generalisation of OX facilitates genetic search for scheduling problem. The author concluded that the representation demonstrates sustained performance on a platform of difficult benchmark problems and the algorithm attains a quality level of optimisation within the range of other GA approaches.

In this paper[7] Nidhish Mathew Nidhiry, stated that the main characteristics of FMS is simultaneous execution of processes and sharing a finite set of resources. In this paper, a genetic algorithm based scheduling of FMS is presented with the objectives; minimising idle time of the machine and minimising the total penalty cost for not meeting the deadline. A software is also developed for getting optimal sequence of operation. The FMS under study has 16 CNC machine tools for processing 43 varieties of product. Various meta heuristic method are used for solving the same scheduling problems. Authors concluded the successful implementation of the procedure developed.

In this paper[8] a complex scheduling problem in flexible manufacturing system (FMS) has been addressed with a novel approach called knowledge based genetic algorithm (KBGA). It was stated that meta heuristics may be used for combinatorial decision making problems. This approach combines knowledge based which uses expert knowledge and genetic algorithm for searching optimal solution. The approach has been applied on four different stages of genetic algorithm. The authors tested KBGA on 10 different moderate size of data. It has been concluded that in the proposed algorithm the operators also used with knowledge based to reduce computational obstacles.

In[2] Vijai Singh et al stated that achieving high performance for an FMS require good scheduling system which makes a right decision at the right time considering the system conditions which is difficult using traditional optimising techniques. Author has made use of genetic algorithm with roulette wheel based selection process. They concluded that roulette wheel selection process is fast and easy to implement and GA based scheduling has considerable potential application to FMS.

Swati Singh et al in their work[1] stated job shop scheduling problem as one of the most difficult NP hard combinatorial optimisation problem. Hence, optimal schedule determination is considered difficult. Their paper focussed on the objective of generating a schedule, minimising total makespan time using genetic algorithm. They concluded that JSSP may have made a schedule depending upon the number of jobs and machine. Genetic algorithm is preferred over traditional scheduling methods

## Result

The comparative analysis of the work studied is presented in the table below:

	Objective	Selection	Crossover	Mutation	Future work
JSSP with delay constraints[1]	Improving solution to JSSP problem(i.e minimising makeapan)	Random	Single point crossover at point ceil $[p/2]$  $P = \text{no.of piece/job}$  $N = \text{total jobs}$	Transposition  Mutation probability=0.5	1.Assumed no breakdown.  2.Considered jobs atomic.  3.Considered transportation and setup time constant.
FMS Scheduling using Roulette wheel selection[2]	Minimising the total makespan time	Roulette wheel selection	Single point crossover with fixed crossover probability in range $[1, L-1]$	Flipping	1.Assumned no breakdown.  2.Transportation and setup time constant
Multiobjective Hybrid GA method[3]	Minimise makespan and maximise tardiness	Roulette wheel approach	Job based crossover(JOX)	Operation swap mutation	Can be further extended by applying heuristic methods separately or in

					conjunction with HGA>
Penalty method for constraints in JSSP[4]	Minimise completion time for processing all jobs	Clonal selection	Normal	Inverse, interchange, Insert	Proposed algo to more practical and integrated problem
HGA & variable neighbourhood descent algo for FJSSP(FMS)[5]	1.minimise makespan 2.minimise maximal machine workload 3.minimise total workload	Ranking method	1.Order crossover for operation sequence vector. 2.Extended order crossover and uniform crossover for machine assignment vector	Allele based and immigration mutation	Moving 3 or more operation,the balance between genetic search and total search
Generalised permutation approach to FMS with GA[6]	How to represent the problem in the algorithm	Permutation	Generalised order crossover(GOX)	-	-
Evaluation of GA approach for scheduling optimisation of FMS[7]	1.Minimising idle time of machines 2.Minimising total penalty cost	Tournament selection	Single point crossover	Flipping	Will include availability & handling times of load/unload stations,robots,AGVs.
FMS Scheduling with knowledge base GA approach[8]	Throughput & mean flow time have been taken to measure the performance of fms	Knowledge based 1.directional 2.steady 3.unruly	Knowledge based	Knowledge Based	Can be stretched out to various problems of FMS that cover the balancing or allocation of resources.
Discrete Dynamic Approach[9]	Minimising makespan	Random	Subtour chunking crossover	-	Generation of global optimal schedule
Generalised order crossover[10]	Minimising makespan	Neighbourhood mating and local offspring acceptance	GOX i.e. permutation with repetitions	Position based mutation	-
Hybrid GA using random keys[11]	Minimising makespan and maximising	Random and 10% direct	Parametrized uniform crossover	Replacing bottom 20% by random	Breakdowns not considered.

	tardiness			generation	
--	-----------	--	--	------------	--

**Table 1.2:Comparative summarization of the reviewed papers**

## Conclusion

The ability and functions of the machines now a days have been extended on large scales, the scheduling plan of a manufacturing system might not be unique. Hence, on a practical shop floor, many feasible schedules can be found. It can be seen that Genetic Algorithm gives better result than traditional scheduling methods because of which the applicability of the GA based methodology has considerable potential application to manufacturing with further refinement in certain aspects[2] This work, and other current investigations, demonstrates that the genetic algorithm method is broad, approximate search procedure with applications in diverse problem areas.

The future scope can be spanned on the following two facets:

The transportation time and setup time are considered as constant, further work can be done using setup time and transportation time. The jobs are considered to be atomic. Further work can be done by creating schedules with various parts manufactured on different machines.

## References

- [1] S.Kapoor et al Improved Solution to Job Shop Scheduling Problem with Delay constraints using Genetic Algorithm International Journal of Computer Applications (0975 – 8887) Volume 45– No.13, May 2012
- [2] V.Singh et al GA based Scheduling of FMS using Roulette Wheel Selection Process
- [3] M.Hyder et al A hybrid GA for simultaneously scheduling an FMC under multiple objectives
- [4] Liang Sun et al Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function
- [5] Jie Gao et al A hybrid genetic and variable neighbourhood descent algorithm for flexible job shop scheduling problems
- [6] Christian Bierwirth A Generalised Permutation Approach Job Shop Scheduling with Genetic Algorithm
- [7] Nidhish Mathew Nidhiry Evaluation of Genetic Algorithm Approach for Scheduling Optimization of Flexible Manufacturing Systems
- [8] A. Prakash et al FMS scheduling with knowledge based genetic algorithm Approach
- [9] Jian-Bo Yang GA-Based Discrete Dynamic Programming Approach for Scheduling in FMS Environments

- [10] José Fernando Gonçalves, Jorge José de Magalhães Mendes, Maurício G.C. Resende in their paper “A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem”
- [11] Hameshbabu Nanvala Use of Genetic algorithm based approaches in scheduling of FMS:A review
- [12] I.Sriramka Scheduling of Flexible Manufacturing System using Genetic Algorithm
- [13] Kwan Woo Kim et al Network-based hybrid genetic algorithm for scheduling in FMS environments
- [14] Wei He et al Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies
- [15] Vijay Kumar et al Scheduling of flexible manufacturing system using genetic algorithm: A heuristic approach
- [16] Chinnusamy.T.R et al Flexible Manufacturing System Scheduling with Dynamic Environment
- [17] Chuda Basnet et al Scheduling and Control of Flexible Manufacturing Systems: A Critical Review
- [18] Pankaj Upadhyay et al Improving a Flexible Manufacturing Scheduling using Genetic Algorithm
- [19] A. Motaghedi-larijani et al Solving Flexible Job Shop Scheduling with Multi Objective Approach
- [20] Farzad Khorsandi Shahrestani et al Optimization of scheduling flexible manufacturing systems by using multi-objective Genetic algorithm
- [21] Jian Xiong et al Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns
- [22] Nasr Al-Hinai et al Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm
- [23] Liping Zhang et al Dynamic rescheduling in FMS that is simultaneously considering energy consumption and schedule efficiency
- [24] Udhayakumar et al Task Scheduling of AVG in FMS using non-traditional optimisation techniques
- [25] S.N. Sivanandam & S.N. Deepa (2008). Principles of Soft Computing

## References

- [1] S.Kapoor et al Improved Solution to Job Shop Scheduling Problem with Delay constraints using Genetic Algorithm International Journal of Computer Applications (0975 – 8887) Volume 45– No.13, May 2012
- [2] V.Singh et al GA based Scheduling of FMS using Roulette Wheel Selection Process
- [3] M.Hyder et al A hybrid GA for simultaneously scheduling an FMC under multiple objectives
- [4] Liang Sun et al Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function
- [5] Jie Gao et al A hybrid genetic and variable neighbourhood descent algorithm for flexible job shop scheduling problems
- [6] Christian Bierwirth A Generalised Permutation Approach Job Shop Scheduling with Genetic Algorithm
- [7] Nidhish Mathew Nidhiry Evaluation of Genetic Algorithm Approach for Scheduling Optimization of Flexible Manufacturing Systems
- [8] A. Prakash et al FMS scheduling with knowledge based genetic algorithm Approach
- [9] Jian-Bo Yang GA-Based Discrete Dynamic Programming Approach for Scheduling in FMS Environments
- [10] José Fernando Gonçalves, Jorge José de Magalhães Mendes, Maurício G.C. Resende in their paper “A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem”
- [11] Hameshbabu Nanvala Use of Genetic algorithm based approaches in scheduling of FMS:A review
- [12] I.Sriramka Scheduling of Flexible Manufacturing System using Genetic Algorithm
- [13] Kwan Woo Kim et al Network-based hybrid genetic algorithm for scheduling in FMS environments
- [14] Wei He et al Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies
- [15] Vijay Kumar et al Scheduling of flexible manufacturing system using genetic algorithm: A heuristic approach
- [16] Chinnusamy.T.R et al Flexible Manufacturing System Scheduling with Dynamic Environment

- [17] Chuda Basnet et al Scheduling and Control of Flexible Manufacturing Systems: A Critical Review
- [18] Pankaj Upadhyay et al Improving a Flexible Manufacturing Scheduling using Genetic Algorithm
- [19] A. Motaghedi-larijani et al Solving Flexible Job Shop Scheduling with Multi Objective Approach
- [20] Farzad Khorsandi Shahrestani et al Optimization of scheduling flexible manufacturing systems by using multi-objective Genetic algorithm
- [21] Jian Xiong et al Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns
- [22] Nasr Al-Hinai et al Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm
- [23] Liping Zhang et al Dynamic rescheduling in FMS that is simultaneously considering energy consumption and schedule efficiency
- [24] Udhayakumar et al Task Scheduling of AVG in FMS using non-traditional optimisation techniques
- [25] S.N. Sivanandam & S.N. Deepa (2008). Principles of Soft Computing